Tampere University

XINGYANG NI

Towards Better Image Embeddings Using Neural Networks

Tampere University Dissertations 606

Tampere University Dissertations 606

XINGYANG NI

Towards Better Image Embeddings Using Neural Networks

ACADEMIC DISSERTATION
To be presented, with the permission of
the Faculty of Information Technology and Communication Sciences
of Tampere University,
for public discussion in the auditorium TB109
of the Tietotalo, Korkeakoulunkatu 1, Tampere,
on 20 May 2022, at 12 o'clock.

ACADEMIC DISSERTATION
Tampere University, Faculty of Information Technology and Communication Sciences
Finland

Responsible supervisor and Custos
Associate Professor
Esa Rahtu
Tampere University
Finland

Supervisor
Dr. Heikki Huttunen
Visy Oy
Finland

Pre-examiners
Associate Professor
Tuomas Eerola
LUT University
Finland

Associate Professor
Christian Micheloni
University of Udine
Italy

Opponent
Professor
Guoying Zhao
University of Oulu
Finland

Cover design: Roihu Inc.

ISBN 978-952-03-2415-5 (print)
ISBN 978-952-03-2416-2 (pdf)
ISSN 2489-9860 (print)
ISSN 2490-0028 (pdf)
http://urn.fi/URN:ISBN:978-952-03-2416-2

PunaMusta Oy – Yliopistopaino
Joensuu 2022

PREFACE/ACKNOWLEDGEMENTS

Studies presented in this dissertation were conducted at Tampere University and Nokia Technologies between 2018 to 2021. It has been a rewarding experience, and I have developed essential skills in performing independent research.

First of all, I would like to express my profound appreciation to my supervisors, *i.e.*, Heikki Huttunen and Esa Rahtu. This dissertation would not have been possible without their insightful guidance. The knowledge, skills, and abilities that I learned from them would be a great asset throughout my professional career.

Furthermore, I sincerely value the contributions of the co-authors, especially Caglar Aytekin and Francesco Cricri. The dedication and commitment they put to work enable reaching the success of projects at a fast pace.

In addition, I am grateful for the accompany of many colleagues and friends, including Bishwo Adhikari, Jingui Li, Lei Xu, Lingyu Zhu, Saeed Bakhshi Germi, Song Yan, and Wenyan Yang. Having a fabulous time surrounded by those awesome people is delightful.

Moreover, I hit the jackpot with a compassionate family that I can always count on. In particular, my mom Yaner Xia and my dad Dehai Ni have supported me every step of the way during my journey.

Last but not least, special thanks go to DW Documentary. I always learn something new from each one of its top-notch programs on YouTube, and it is enjoyable to discover the world while staying at home.

Xingyang Ni
Tampere, August 2021

ABSTRACT

The primary focus of this dissertation is to study image embeddings extracted by neural networks. Deep Learning (DL) is preferred over traditional Machine Learning (ML) for the reason that feature representations can be automatically constructed from data without human involvement. On account of the effectiveness of deep features, the last decade has witnessed unprecedented advances in Computer Vision (CV), and more real-world applications are expected to be introduced in the coming years.

A diverse collection of studies has been included, covering areas such as person re-identification, vehicle attribute recognition, neural image compression, clustering and unsupervised anomaly detection. More specifically, three aspects of feature representations have been thoroughly analyzed. Firstly, features should be distinctive, *i.e.*, features of samples from distinct categories ought to differ significantly. Extracting distinctive features is essential for image retrieval systems, in which an algorithm finds the gallery sample that is closest to a query sample. Secondly, features should be privacy-preserving, *i.e.*, inferring sensitive information from features must be infeasible. With the widespread adoption of Machine Learning as a Service (MLaaS), utilizing privacy-preserving features prevents privacy violations even if the server has been compromised. Thirdly, features should be compressible, *i.e.*, compact features are preferable as they require less storage space. Obtaining compressible features plays a vital role in data compression.

Towards the goal of deriving distinctive, privacy-preserving and compressible feature representations, research articles included in this dissertation reveal different approaches to improving image embeddings learned by neural networks. This topic remains a fundamental challenge in Machine Learning, and further research is needed to gain a deeper understanding.

CONTENTS

ORIGINAL PUBLICATIONS

Publication I X. Ni, L. Fang and H. Huttunen. Adaptive L2 Regularization in Person Re-Identification. *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, 9601–9607. DOI: 10.1109/ICPR48806.2021.9412481.

Publication II X. Ni and E. Rahtu. FlipReID: Closing the Gap Between Training and Inference in Person Re-Identification. *2021 9th European Workshop on Visual Information Processing (EUVIP)*. 2021, 1–6. DOI: 10.1109/EUVIP50544.2021.9484010.

Publication III X. Ni, H. Huttunen and E. Rahtu. On the Importance of Encrypting Deep Features. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2021, 4125–4132. DOI: 10.1109/ICCVW54120.2021.00460.

Publication IV X. Ni and H. Huttunen. Vehicle Attribute Recognition by Appearance: Computer Vision Methods for Vehicle Type, Make and Model Classification. *Journal of Signal Processing Systems* 93.4 (2021), 357–368. DOI: 10.1007/s11265-020-01567-6.

Publication V C. Aytekin, X. Ni, F. Cricri, J. Lainema, E. Aksu and M. Hannuksela. Block-optimized Variable Bit Rate Neural Image Compression. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018, 2551–2554.

Publication VI C. Aytekin, X. Ni, F. Cricri and E. Aksu. Clustering and Unsupervised Anomaly Detection with L2 Normalized Deep Auto-Encoder Representations. *2018 International Joint Conference on*

Neural Networks (IJCNN). 2018, 1–6. DOI: 10.1109/IJCNN.2018.8489068.

Author's contribution

Publication I	As the first author, Xingyang Ni devised the adaptive L_2 regularization mechanism, implemented the source code, conducted the experiments, drafted the paper, and maintained the repository.
Publication II	As the first author, Xingyang Ni devised the FlipReID structure with the flipped loss, implemented the source code, conducted the experiments, drafted the paper, and maintained the repository.
Publication III	As the first author, Xingyang Ni devised the attack scenarios and the ShuffleBits mechanism, implemented the source code, conducted the experiments, drafted the paper, and maintained the repository.
Publication IV	As the first author, Xingyang Ni carried out a literature review, devised the approach of analyzing vehicle attribute recognition from the perspective of image retrieval, implemented the source code, conducted the experiments, and drafted the paper.
Publication V	As the second author, Xingyang Ni devised inference-stage code optimization and the deblocking network, assisted in implementing the source code, conducting the experiments, and drafting the paper. On the other hand, the first author Caglar Aytekin initiated the ideas of binarization simulation, variable bit rates with multiple networks, and entropy-friendly representations.
Publication VI	As the second author, Xingyang Ni devised the approach of extending the clustering algorithm for unsupervised anomaly detection, assisted in implementing the source code, conducting the experiments, and drafting the paper. On the other hand, the first author Caglar Aytekin initiated the idea of imposing the L_2 normalization constraint on features during training.

1 INTRODUCTION

1.1 Background

Studies in Machine Learning (ML) aim to build a model that learns to solve problems without being explicitly programmed [101, 160]. Unlike explicit programming that consists of specifically defined instructions, machine learning makes it feasible to learn the knowledge from data, and the same algorithm might be reusable on a different problem after replacing the dataset. The last decade has witnessed rapid advancement in the field of machine learning. Significant progress has been made in academic research, *e.g.*, face recognition [35, 165], speech synthesis [140, 193], action recognition [90, 190], and language understanding [36, 153]. In addition, practical applications are ubiquitous in everyday life, *e.g.*, Google Lens[1], Siri[2], Facebook ads[3], Alexa[4], and Cortana[5].

There are two critical enablers behind the tremendous success of machine learning techniques. On the one hand, large-scale and high-quality datasets have been collected. Different types of datasets are available, including image data [33, 107], audio data [133, 142], video data [105, 172], and text data [126, 130]. The importance of data collection should not be overlooked since having large amounts of data for training contributes to better generalization during inference. On the other hand, modern computing hardware provides high processing power at an affordable price. Apart from general-purpose Central Processing Unit (CPU), specialized processors have been developed to achieve faster speed and improved efficiency. Notable hardware units include Graphics Processing Unit (GPU), Tensor Processing Unit (TPU), and Field-Programmable Gate Array (FPGA).

[1] https://lens.google/
[2] https://www.apple.com/siri/
[3] https://www.facebook.com/business/ads
[4] https://developer.amazon.com/en-US/alexa
[5] https://www.microsoft.com/en-us/cortana

Feature extraction signifies the process of converting data from one feature space to another [38]. Regardless of the type of machine learning algorithm used, deriving a compact yet discriminative representation has a significant impact on the overall performance. Given the supervised image classification task, features are extracted from raw pixels, and class probabilities are obtained afterward [104]. In the case of solving dimensionality reduction which is unsupervised, an autoencoder learns the low-dimensional codes which will be further used to reconstruct the high-dimensional input vectors [76]. To master the game of Go using reinforcement learning, both the value network and the policy network construct a representation of the board position [168]. Depending on whether human intervention is required, there are two types of approaches for feature extraction, *i.e.*, manual and automated.

Being manual means that a human expert builds specific strategies on feature extraction, and no learning is involved. Firstly, one could extract image features by analyzing edges [150, 156], corners [45, 66], blobs [45, 117], and ridges [27, 65]. Secondly, audio features include spectral features [67, 91, 99] and rhythm features [59]. Thirdly, studying motion estimation [53, 225] plays an important role when processing video data. Last but not least, word embedding on text data can be obtained from vector space model [159], latent semantic analysis [32, 96], and locally linear embedding [158]. By contrast, feature engineering differs from the aforementioned methods because transformations are applied to the vanilla features and the synthetic features may improve the predictive performance [134]. It is a time-consuming process, and domain knowledge of the problem at hand is required.

Being automated means that an algorithm processes the raw data directly and a suitable representation is automatically learned [14]. Also known as feature learning, it obviates the need for adopting an explicit feature extractor and provides superior performance than hand-crafted features. A Neural Network (NN) consists of multiple non-linear layers, while each layer transforms the data at one level and more abstract representation is learned at higher layers [110]. With the speedy development of large-scale datasets and modern computing hardware, neural network techniques have become the de facto standard in many fields of machine learning. Remarkable progresses have been accomplished in analyzing image data [71, 104], audio data [75, 112], video data [97, 208], and text data [36, 186].

1.2 Research Questions

The scope of this dissertation is limited to learning image embeddings with neural networks. The primary objective is to devise a feature extraction pipeline that is suitable for the task at hand. In particular, there are three main research questions to investigate:

- How to extract distinctive features so that samples from the same category are closer to each other than those from different categories? A typical scenario is finding the gallery sample which is most similar to a query sample.
- How to make features privacy-preserving so that an adversary can not infer sensitive information? It is especially relevant for a machine learning model deployed in production as deep features have to be transferred and stored.
- How to obtain compressible features so that the storage requirements are lowered? Note that a tradeoff between fidelity and compression ratio has to be made.

1.3 Summary of Publications

To investigate the aforementioned research questions, extensive studies have been conducted in multiple areas, including person re-identification [135, 137, 138], vehicle attribute recognition [136], neural image compression [7], clustering and unsupervised anomaly detection [6]. In the following, the main contributions of each publication are outlined separately.

Firstly, we propose an adaptive L_2 regularization mechanism that applies to any neural network in [135]. The regularization factors in existing studies are handpicked via hyperparameter optimization, and these values remain constant during training. Such practice would result in suboptimal performance because evaluating a set of hyperparameters is time-consuming while the search space is extremely large. We address this challenge by directly learning the regularization factors through backpropagation. More specifically, a trainable scalar variable is defined for each parameter, and a scaled hard sigmoid function is applied afterward. The resulting tensors control the regularization strength, and no human intervention is required to adjust their values manually.

Secondly, we introduce a novel network architecture termed FlipReID in [138]. Adopting data augmentation throughout training is a popular technique to suppress overfitting and improve generalization. In the inference procedure, feature vectors are obtained for both the original and flipped images, while the mean feature vectors are adopted for evaluation. Although such test-time augmentation scheme results in a decent performance boost, it leads to a mismatch between training and inference because the model is not explicitly optimized on the mean feature vectors. We present a systematic study and explore various options to investigate this problem.

Thirdly, we perform model inversion attacks under a black-box setting in [137]. An adversary can send query requests to a server that runs a machine learning model, and feature vectors are extracted from samples in a local dataset. Moreover, the adversary has gained unauthorized access to feature vectors of user data. We carry out experiments on two attack scenarios, namely recognizing auxiliary attributes and reconstructing user data. Given unencrypted feature vectors, it is viable for the adversary to infer sensitive information. Therefore, we present an algorithm for encrypting the deep features via shuffling the binary sequence of each floating-point number. Compared with conventional encryption methods, the proposed method can be seamlessly integrated as part of any neural network.

Fourthly, we search and evaluate related literature on vehicle attribute recognition in [136]. For each vehicle attribute (*i.e.*, type, make, and model), we compare commonly used datasets and summarize notable methods that use either hand-crafted features or automatically discovered features. Under a scenario in which the prediction of a query sample is the label of its nearest neighbor in a gallery set, we analyze the recognition task from the perspective of image retrieval. Experimental results of different loss functions are reported on a large-scale dataset. In addition, recommendations for future research directions are made.

Fifthly, we present an algorithm for neural image compression in [7]. An autoencoder is utilized to compress an image block by block, *i.e.*, the encoder generates a low-dimensional representation and the decoder reconstructs the input. Since the binarization operation is non-differentiable, noises are added to the codes to simulate it. Afterward, entropy encoding is applied to compress the binary sequence. When encoding a specific block, the weights of the decoder remain unchanged while we continue to update the encoder so that the codes are further optimized. In an effort to remove the blocking artifacts, a separate model is appended at the end.

Lastly, we analyze two problems in [6], *i.e.*, clustering and unsupervised anomaly detection. In previous works on unsupervised learning, opting for autoencoders has been a popular choice. While the encoder extracts the feature representations, we propose to impose the l_2 normalization constraint on such embeddings so that the extracted features lie on a unit sphere. We have conducted experiments on both problem settings, and adding the l_2 normalization constraint brings significant performance improvements.

1.4 Structure

The rest of this dissertation is organized as follows. Chapter 2 highlights three separate works in person re-identification. Afterward, Chapter 3 provides insights on vehicle attribute recognition. Chapter 4 discusses a methodology on neural image compression, while Chapter 5 shifts the focus to clustering and unsupervised anomaly detection. Last but not least, Chapter 6 contains the concluding remarks, followed by the original publications.

2 PERSON RE-IDENTIFICATION

In this chapter, we analyze image embeddings in person re-identification, and it is organized as follows. Firstly, the essential components of the background are explained in Section 2.1. Secondly, Section 2.2 introduces the adaptive L_2 regularization mechanism [135]. Thirdly, Section 2.3 analyzes the gap between the training and inference procedures [138]. Lastly, Section 2.4 presents a study on model inversion attacks in a black-box setting [137].

2.1 Background

While comprehensive surveys on person re-identification can be found in [12, 207, 220], we provide a succinct introduction to the research topic. Person re-identification is a scientific discipline that attempts to associate images/videos (of the same person) captured using multiple cameras. Figure 2.1 illustrates a common scenario: given a query sample (*i.e.*, image or video), an algorithm finds the most similar samples in a gallery set. More specifically, feature representations are extracted from those samples, and a metric measures the distance of two feature vectors. It is challenging for the reason that significant differences can be observed among samples, *e.g.*, occlusion, illumination, and viewpoint.

Surveillance cameras have been deployed in public places on a large scale, even though complaints have been raised due to concerns about privacy and security[1]. In resemblance to face recognition in which a system detects and identifies the faces, much attention has been focused on applying person re-identification in video surveillance. On the one hand, surveillance cameras are ubiquitous, and vast amounts of data are already available. On the other hand, taking images/videos is a non-invasive procedure, and it does not require explicit consent from the pedestrians in question.

[1] https://www.dw.com/en/in-germany-controversy-still-surrounds-video-surveillance/a-50976630

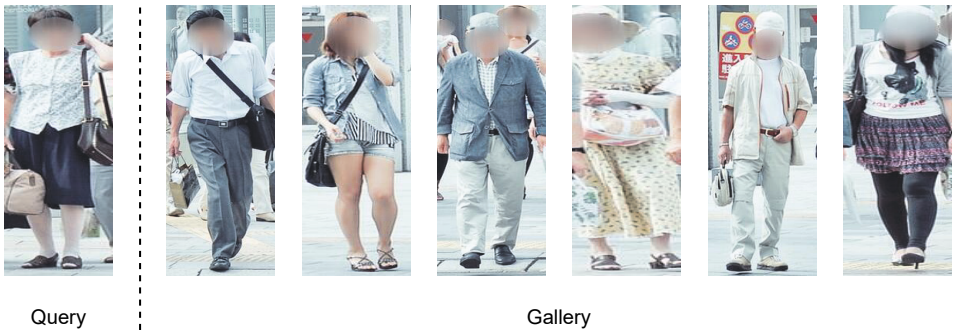


Figure 2.1 Samples in a gallery set are sorted based on their similarities to the query sample. Photo by JJ Ying.

2.1.1 Dataset

Three datasets are included, namely, Market-1501 [219], DukeMTMC-reID [155] and MSMT17 [194]. Table 2.1 shows a comparison in terms of the number of samples, identities, and cameras. Each dataset has three non-overlapping partitions: training set, query set, and gallery set.

- The Market-1501 dataset comprises 32,217 images from 1,501 pedestrians. In total, there are six cameras, while each pedestrian is captured by at least two cameras. Images assigned to ID "-1" are junks that would be neglected during testing. By contrast, images with ID "0" are false positives produced by a detector, and those samples contain only the background.
- The DukeMTMC-reID dataset contains 702 and 1,110 identities in the training and test sets, respectively. Note that 408 identities appear in only one camera, while the remaining 1,404 identities show up in no less than two cameras. In total, 36,411 images have been collected on campus with eight outdoor cameras.
- The MSMT17 dataset is gathered using 3 indoor cameras and 12 outdoor cameras. It consists of 126,441 images of 4,101 pedestrians, making it the largest dataset for image-based person re-identification at present. Unlike other datasets, the testing set (*i.e.*, query and gallery) has approximately two times more samples than the training set. Such practice encourages practitioners to devise algorithms that efficiently utilize a finite amount of training samples.

Table 2.1 Comparisons among the datasets for person re-identification, including Market-1501 [219], DukeMTMC-reID [155] and MSMT17 [194].

Dataset	Market-1501	DukeMTMC-reID	MSMT17
Training Samples	12,936	16,522	32,621
Training Identities	751	702	1,041
Test Query Samples	3,368	2,228	11,659
Test Gallery Samples	15,913	17,661	82,161
Test Identities	751	1,110	3,060
Cameras	6	8	15

2.1.2 Evaluation Metric

After the training procedure is finalized, feature representations of the test samples are extracted. Either Euclidean or cosine distance function is applied to each query-gallery pair, and a distance matrix is obtained. Subsequently, two evaluation metrics are utilized to measure the overall performance of the person re-identification model, *i.e.*, mean Average Precision (mAP) [219] and Cumulative Matching Characteristic (CMC) rank-k accuracy [192].

The CMC rank-k accuracy represents the fraction of cases in which a correct match for the query sample is found among the top-k gallery samples. In the literature, the hyperparameter k is typically set to 1, 5, or 10 [176]. As clarified in [219], the CMC rank-k accuracy is a suitable metric under the condition that there is only one ground truth match for each query sample. Unfortunately, this hypothesis does not hold for the datasets that are collected during the recent years [155, 194, 219]. Since there could be multiple gallery samples that match a query sample, a model might retrieve the easy matches but fail to identify the hard matches.

To address the aforementioned limitation, the mAP score is proposed as an alternative metric in [219]. For each query sample, the precision-recall curve is drawn, and it shows the trade-off between precision and recall for different thresholds. Precision is the percentage of relevant samples among the retrieved samples, while recall is the percentage of relevant samples that get retrieved. Also known as Average Precision (AP), the area under the precision-recall curve is measured. Afterward, the mean value over the query set is calculated as the final score. Compared with the CMC rank-k accuracy, the mAP score provides a more reliable measurement.

2.2 Adaptive L_2 Regularization

2.2.1 Related Work

Given a training dataset with finite samples, a model might remember it too well, and even the noise is learned. It leads to the overfitting issue, *i.e.*, the model works excellently on the training dataset while the performance on a test dataset is poor [25]. By contrast, generalization refers to the model's capability of processing an unseen dataset [131]. As a countermeasure, imposing an effective regularization strategy can suppress overfitting and improve generalization [55]. In the following, we provide a non-exhaustive list of common regularization strategies.

- Norm penalties can be included: L_1 regularization promotes sparsity [214], while L_2 regularization drives parameters toward zero [121].
- Augmented samples can be synthesized from existing samples by introducing label-preserving transformations, *e.g.*, zero padding, random cropping, and horizontal flipping [167].
- Labeling errors are unavoidable, especially when the dataset gets large. The label smoothing [177] method injects noise into the ground truth labels.
- Also known as dropout [77], a portion of the units is set to zero, and the subsequent layers rely on the remaining non-zero units to make predictions.
- In the case of Multi-Task Learning (MTL), several related tasks are solved in parallel [16]. Features learned by one task might benefit other tasks as well.
- Parameter sharing forces sets of parameters to be equal [109]. In a Convolutional Neural Network (CNN), the same convolution kernel is applied on different patches, regardless of the patch's position.
- While the error on the training dataset decreases steadily as training proceeds, the validation error might decrease first but rise later [149]. Early stopping shines a light on when does overfitting occur so that training can be stopped.
- Adversarial examples can be generated by adding small perturbations to vanilla examples, and a model might misclassify those with high confidence [57]. Training on adversarial examples introduces a local constancy prior, *i.e.*, a model should be locally constant in the neighborhood of the training data [55].

2.2.2 Proposed Method

A neural network comprises successive layers to learn a complex mapping between the inputs and outputs. In general, each layer has trainable parameters that would be updated with respect to the gradients. For example, a dense layer has a 2-D kernel and a 1-D bias, while a 2-D convolutional layer has a 4-D kernel and a 1-D bias. Thus, the set of trainable parameters in a neural network can be written as

$$P = \{\boldsymbol{w}_n \mid n = 1, \ldots, N\}, \tag{2.1}$$

where N is the number of distinct parameters. Note that $\boldsymbol{w}_n$ can be a scalar, a vector, a matrix, or a tensor of higher order.

In existing studies on L_2 regularization, a separate loss term is added to the objective function:

$$L_\lambda(P) = L(P) + \lambda \sum_{n=1}^{N} ||\boldsymbol{w}_n||_2^2, \tag{2.2}$$

where $L(P)$ denotes the original objective function, and $L_\lambda(P)$ symbolizes the updated objective function. Besides, $||\boldsymbol{w}_n||_2^2$ stands for the sum of the squares of all elements in $\boldsymbol{w}_n$, while the constant coefficient $\lambda \in \mathbb{R}_+$ corresponds to the strength level of regularization. Naturally, a unique coefficient can be assigned to each parameter so that L_2 regularization is applied at different strength levels:

$$L_\lambda(P) = L(P) + \sum_{n=1}^{N} \left(\lambda_n ||\boldsymbol{w}_n||_2^2\right), \tag{2.3}$$

where $\lambda_n \in \mathbb{R}_+$.

Like any other hyperparameter, a set of optimal regularization factors has to be chosen via tuning. Given the ResNet50 [71] backbone, there are more than 100 distinct parameters. On the one hand, the search spaces of those continuous regularization factors are large. On the other hand, evaluating the performance of a specific set is time-consuming. As a consequence, it is computationally impractical to tune the regularization factors one by one. In practice, the same value is used for regularization factors that are associated with the same type of parameters, *e.g.*, kernels in the dense layers.

To alleviate the computational complexity of hyperparameter optimization, one may find suitable values for regularization factors based on learning. A straightforward extension is to replace the manually assigned floating numbers with trainable scalar variables that can be optimized during backpropagation. More specifically, Equation 2.3 stays the same while $\lambda_n \in \mathbb{R}$. Without a constraint on the feasible region of regularization factors, the aforementioned method leads to model collapse. Given a $\lambda_n \in \mathbb{R}\_$, naively increasing $||w_n||_2^2$ guarantees that value of the objective function would decrease sharply. In other words, the penalty term would dominate the optimization process, and the model would not extract meaningful features of the inputs.

In order to fix the problem of model collapse, the hard sigmoid function is applied on the scalar variables, and it is formulated as

$$f(x) = \begin{cases} 0, & \text{if } x < -c \\ 1, & \text{if } x > c \\ x/(2c) + 0.5, & \text{otherwise.} \end{cases} \tag{2.4}$$

where the constant number c is set to 2.5. Additionally, we have

$$\lambda_n = f(\theta_n), \tag{2.5}$$

where $\theta_n \in \mathbb{R}$ $(n = 1, \ldots, N)$ refers to the trainable scalar variables. It is apparent that the regularization factor λ_n can never be negative, and model collapse should not happen.

Moreover, we introduce the amplitude $A \in \mathbb{R}_+$ which sets the maximum value for the regularization strength:

$$\lambda_n = A f(\theta_n). \tag{2.6}$$

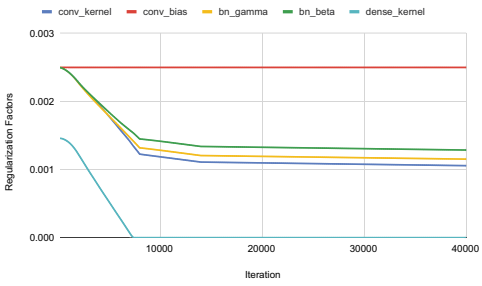
With a proper value for the amplitude A, the regularization strength would not become excessively strong.

Last but not least, combining Equation (2.3) and (2.6) concludes our proposed method that allows learning regularization factors adaptively:

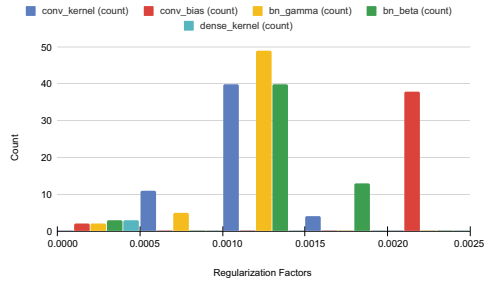
$$L_\lambda(P) = L(P) + \sum_{n=1}^{N} \left(A f(\theta_n) ||w_n||_2^2 \right). \tag{2.7}$$

Table 2.2 Comparisons with notable methods on Market-1501 [219], DukeMTMC-reID [155] and MSMT17 [194]. mAP: mean Average Precision. R1: CMC rank-1 accuracy. -: result is unavailable. §: the re-ranking [222] approach is utilized. Reproduced with permission from [135].

Method	Backbone	Market-1501		DukeMTMC		MSMT17	
		mAP	R1	mAP	R1	mAP	R1
Annotators [213]	-	-	93.5	-	-	-	-
PCB [176]	ResNet50	81.6	93.8	69.2	83.3	-	-
IANet [78]	ResNet50	83.1	94.4	73.4	87.1	46.8	75.5
AANet [179]	ResNet50	82.5	93.9	72.6	86.4	-	-
CAMA [204]	ResNet50	84.5	94.7	72.9	85.8	-	-
DGNet [221]	ResNet50	86.0	94.8	74.8	86.6	52.3	77.2
OSNet [224]	OSNet	84.9	94.8	73.5	88.6	52.9	78.7
MHN [22]	ResNet50	85.0	95.1	77.2	89.1	-	-
BDB [30]	ResNet50	86.7	95.3	76.0	89.0	-	-
BAT-net [43]	GoogLeNet	87.4	95.1	77.3	87.7	56.8	79.5
SNR [92]	ResNet50	84.7	94.4	72.9	84.4	-	-
HOReID [188]	ResNet50	84.9	94.2	75.6	86.9	-	-
PyrAttNet [129]	ResNet50	87.6	95.8	78.3	88.4	-	-
RGA-SC [217]	ResNet50	88.4	96.1	-	-	57.5	80.3
SCSN [23]	ResNet50	88.5	95.7	79.0	91.0	58.5	83.8
Baseline (Ours)	ResNet50	87.2	94.6	78.9	88.0	57.7	79.1
Adaptive L_2 Regularization (Ours)	ResNet50	88.3	95.3	79.9	88.9	59.4	79.6
	ResNet101	88.6	94.8	80.6	89.2	61.9	81.3
	ResNet152	88.9	95.6	81.0	90.2	62.2	81.7
	ResNet152§	94.4	96.0	90.7	92.2	76.7	84.9



(a) The median value of regularization factors against the number of iterations.



(b) The histogram of regularization factors in the last epoch.

Figure 2.2 Inspection of regularization factors in each category, using the adaptive L_2 regularization method. Reproduced with permission from [135].

2.2.3 Experimental Results

Experiments have been conducted on three popular datasets in person re-identification: Market-1501 [219], DukeMTMC-reID [155] and MSMT17 [194]. The performance of a model is measured by calculating the scores of mean Average Precision (mAP) and Cumulative Matching Characteristic (CMC) rank-k accuracy on the query and gallery sets.

Table 2.2 provides comparisons among baseline, adaptive L_2 regularization, and other notable methods from literature. Conventional L_2 regularization is utilized in the baseline method, *i.e.*, the regularization factors stay constant during training. A set of optimal values is found via hyperparameter optimization. The performance of the baseline method is comparable with other works. After switching to the adaptive L_2 regularization method, substantial improvements can be observed. Most significantly, we obtain the highest mAP score on MSMT17 among methods that use the ResNet50 [71] backbone. Furthermore, using deeper backbones and utilizing re-ranking [222] result in superior performance.

Figure 2.2 gives insights on the regularization factors that are learned through backpropagation. Parameters are grouped into five categories, namely, the kernel and the bias in convolutional layers, the gamma and the beta in batch normalization [85] layers, as well as the kernel in dense layers. Since the bias terms in dense layers are disabled, no regularization is applied to those parameters.

On the one hand, Figure 2.2a illustrates how the median value of regularization factors in each category changes as the training procedure proceeds. The regularization factors of *conv_bias* reach a plateau from the beginning. By contrast, the curves of the other four groups decrease gradually at first and remain largely unchanged afterward. In addition, the regularization factors of *dense_kernel* converge to 0 at the early stage of training.

On the other hand, Figure 2.2b visualizes the histogram of regularization factors in each category after training has finished. The interval $[0, 0.0025]$ is evenly divided into five bins, and the number of observations that fall into each subinterval is shown. Each regularization factor is optimized independently. Considering the regularization factors of *conv_bias*, there are 2 entries in the interval $[0, 0.0005]$ and 38 entries in the interval $[0.0020, 0.0025]$.

2.2.4 Summary

We proposed a novel regularization technique that can be easily applied when optimizing a neural network. In the conventional L_2 regularization method, the regularization factors stay constant during training, and those hyperparameters have to be optimized by tuning. Because it is tedious to perform hyperparameter optimization, we introduce the adaptive L_2 regularization method in which the regularization factors are learned through backpropagation. For each distinct parameter, we define a trainable scalar variable that is further processed by a scaled hard sigmoid function, and the resulting tensors determine the regularization strength. We validate the effectiveness of our proposed method in the setting of person re-identification. Better image embeddings are acquired, and substantial score improvements can be observed. A natural extension is to conduct experiments in other domains, such as image classification, object detection, and semantic segmentation.

2.3 Closing the Gap between Training and Inference

2.3.1 Related Work

Machine learning algorithms rely on a dataset to optimize its internal parameters. The importance of datasets could not be overemphasized, especially for modern neural networks. Under an extreme scenario that the ground truth labels in an image classification dataset are replaced by entirely unstructured random noise, a model could still fit reasonably well on the training dataset while it would not generate meaningful predictions on the test dataset [210]. Such overfitting issue poses a severe challenge, and adopting data augmentation during the training procedure is a prevalent option to improve generalization [167]. While the size of a dataset is finite, label-preserving transformations can be applied to vanilla samples. Thus, the dataset is artificially enlarged without the need for collecting and annotating new samples.

More advanced augmentation policies have been developed, apart from the basic methods such as zero padding, random cropping, and horizontal flipping. Given a pair of samples, convex combinations of the inputs and labels are utilized to optimize a model in the mixup [212] work. It imposes the prior knowledge that linear interpolations of the inputs are expected to generate linear interpolations of the labels. In both Cutout [37] and Random Erasing [223], a rectangular region is randomly masked out, and feeding images with various levels of occlusion improves the robustness of a model. Last but not least, a method based on Reinforcement Learning (RL) is introduced in [28], and it automatically searches for the optimal policies for the problem at hand. More interestingly, a set of policies learned on one dataset is proven to be transferable on other datasets.

It is indisputable that utilizing suitable augmentation policies during training alleviates overfitting and improves generalization. By contrast, better predictive performance can be achieved by incorporating test-time augmentation during inference. Extra computations are required since one has to process multiple augmented samples and aggregate the predictions. For image classification models in [71, 104], a 10-crop testing method is applied. Half samples comprise four corner patches and one center patch in the original image, while the other half is cropped from the horizontal flip. For person re-identification models in [72, 189], the mean feature vectors of the original and flipped images are calculated for evaluation.

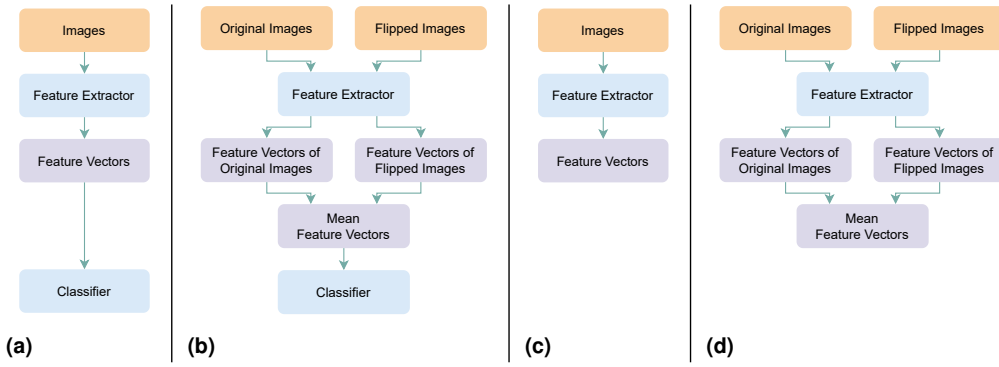


Figure 2.3 Comparisons among different options for training/inference: (a) training in baseline; (b) training in FlipReID; (c) inference with single image; (d) inference with double images. Reproduced with permission from [138].

2.3.2 Proposed Method

Figure 2.3 shows several structures that are adopted during training and inference. On the one hand, Figure 2.3a and Figure 2.3b illustrate the architectures of two neural networks used in training. On the other hand, Figure 2.3c and Figure 2.3d demonstrate the models deployed in inference. In the following, a more detailed explanation is given.

Figure 2.3a provides a high-level overview of the baseline method. Given a set of images, the feature extractor discovers feature representations. More specifically, it consists of two separate branches: the global branch extracts features from the complete feature maps, while the regional branches partition works on the horizontal stripes of the complete feature maps. During inference, the outputs of global and regional branches get concatenated. Furthermore, the classifier consists of batch normalization [85] and dense layers, and it generates the probabilities of classes. Last but not least, two types of loss terms are utilized to optimize the model. The batch hard triplet loss [74] is applied to the outputs of the feature extractor, and it exploits the hardest positive and negative samples within the batch when forming the triplets. In addition, the categorical cross-entropy loss [216] is applied to the outputs of the classifier so that the predicted probabilities are consistent with the ground truth labels.

Compared with Figure 2.3a, the classifier is absent in Figure 2.3c, and feature vectors generated by the feature extractor are used to evaluate the model's performance. Note that test-time augmentation is disabled, *i.e.*, features are directly calculated from only one sample. On the contrary, one may aggregate predictions of multiple augmented samples to boost performance even further. Figure 2.3d gives an alternative option, in which features of the original and flipped images are extracted. Afterward, the final representation is obtained by calculating the mean of the feature vectors.

In existing studies, the training procedure is performed using Figure 2.3a, while the inference procedure is in line with either Figure 2.3c or Figure 2.3d, depending on whether test-time augmentation is applied or not. However, adopting Figure 2.3a in training and choosing Figure 2.3d in inference is inconsistent from each other. In particular, the mean feature vectors in Figure 2.3d are not involved in optimizing the model in Figure 2.3a, and such gap leads to sub-optimal performance. Thus, we introduce the FlipReID structure that is illustrated in Figure 2.3b. Based on the architecture in Figure 2.3d, a classifier is appended at the end of Figure 2.3b so that the neural network can be optimized. Given a model trained using Figure 2.3b, opting for Figure 2.3d in inference does not induce inconsistency. Nevertheless, there exists a gap if test-time augmentation is disabled. The feature vectors extracted in Figure 2.3c are not explicitly used to optimize the model in Figure 2.3b, *i.e.*, the batch hard triplet loss [74] is only applied on the mean feature vectors. To address this problem, we incorporate the flipping loss, which refers to the mean squared error (MSE) between feature vectors of corresponding image pairs (see Figure 2.3b).

In summary, we have the following options:

- Figure 2.3a, 2.3c: Inference is consistent with training.
- Figure 2.3a, 2.3d: Inference deviates from training, *i.e.*, the classifier is only optimized on feature vectors of single image.
- Figure 2.3b, 2.3c: Inference deviates from training, *i.e.*, the classifier is only optimized on feature vectors of double images. Alternatively, the flipping loss can be integrated into the objective function.
- Figure 2.3b, 2.3d: Inference is consistent with training. Alternatively, the flipping loss can be integrated into the objective function.

Table 2.3 Comparisons among previous works, our baseline and FlipReID. mAP: mean Average Precision. R1: CMC rank-1 accuracy. †: inference with single image (see Figure 2.3c). ‡: inference with double images (see Figure 2.3d). §: the re-ranking [222] approach is utilized. The best results are highlighted in bold. Reproduced with permission from [138].

Method	Backbone	Market-1501		DukeMTMC-reID		MSMT17	
		mAP	R1	mAP	R1	mAP	R1
PCB, ECCV 2018 [176]	ResNet50	81.6	93.8	69.2	83.3	-	-
BoT, CVPRW 2019 [72, 124]	ResNet50	86.1	94.4	77.0	87.2	50.2	74.1
SCSN, CVPR 2020 [23]	ResNet50	88.5	95.7	79.0	91.0	58.5	83.8
GASM, ECCV 2020 [73]	ResNet50	84.7	95.3	74.4	88.3	52.5	79.5
AGW, TPAMI 2020 [72, 206]	ResNet50	88.2	95.3	79.9	89.0	55.6	78.3
FastReID, arXiv 2020 [72]	ResNet50	88.2	95.4	79.8	89.6	59.9	83.3
1.1 Baseline†	ResNet50	88.1	95.0	78.9	89.4	61.7	81.5
1.2 Baseline‡		88.6	95.0	79.5	89.4	62.9	82.1
2.1 FlipReID (w/o flipping loss)†	ResNet50	86.2	94.7	77.2	88.9	57.1	79.5
2.2 FlipReID (w/o flipping loss)‡		88.5	95.5	79.8	90.2	64.3	83.6
3.1 FlipReID (w/ flipping loss)†	ResNet50	87.6	95.2	78.9	89.1	61.4	81.9
3.2 FlipReID (w/ flipping loss)‡		88.5	95.3	79.8	89.4	64.3	83.3
3.3 FlipReID (w/ flipping loss)‡§		94.6	96.0	**90.9**	92.5	79.5	86.3
1.1 Baseline†	IBN-ResNet50	88.4	94.8	79.0	88.7	64.6	83.4
1.2 Baseline‡		88.9	95.5	79.6	88.8	65.7	84.2
2.1 FlipReID (w/o flipping loss)†	IBN-ResNet50	86.9	94.3	77.5	88.7	60.4	81.3
2.2 FlipReID (w/o flipping loss)‡		88.7	94.8	79.7	89.4	66.2	84.4
3.1 FlipReID (w/ flipping loss)†	IBN-ResNet50	87.9	94.7	78.8	88.9	63.2	83.1
3.2 FlipReID (w/ flipping loss)‡		88.6	95.0	79.8	89.6	65.9	84.5
3.3 FlipReID (w/ flipping loss)‡§		94.1	95.4	89.8	91.8	80.2	87.3
1.1 Baseline†	ResNeSt50	89.3	95.8	80.0	89.6	66.0	84.2
1.2 Baseline‡		89.7	**96.2**	80.5	89.9	66.9	84.5
2.1 FlipReID (w/o flipping loss)†	ResNeSt50	88.6	95.2	79.9	90.0	64.6	83.9
2.2 FlipReID (w/o flipping loss)‡		89.6	95.7	81.2	90.7	67.6	85.3
3.1 FlipReID (w/ flipping loss)†	ResNeSt50	88.9	95.2	80.7	90.5	66.0	84.6
3.2 FlipReID (w/ flipping loss)‡		89.6	95.5	81.5	90.9	68.0	85.6
3.3 FlipReID (w/ flipping loss)‡§		**94.7**	95.8	90.7	**93.0**	**81.3**	**87.5**

2.3.3 Experimental Results

Table 2.3 provides comprehensive comparisons among previous works, our baseline and FlipReID. Performance is reported on three datasets, namely, Market-1501 [219], DukeMTMC-reID [155] and MSMT17 [194]. Two evaluation metrics are utilized, *i.e.*, the mean Average Precision (mAP) score and the Cumulative Matching Characteristic (CMC) rank-k accuracy.

Among the three datasets that are included, the MSMT17 dataset is the largest, and it comprises a diverse set of pedestrian images taken under different conditions. Note that the other two datasets are already saturated, *i.e.*, recent methods are over-fitted to the test datasets, and scores are not informative [135]. In particular, the AGW [206] method is comparable with the FastReID [72] method on Market-1501 and DukeMTMC-reID, but noticeable performance difference can be observed on MSMT17.

Compared with existing studies, our baseline method using single image (see Figure 2.3c) achieves a higher score on MSMT17. Moreover, leveraging double images (see Figure 2.3d) leads to more distinctive feature representations, regardless of how the model is trained. Under the condition that test-time augmentation is present, the speed of feature extraction would be approximately halved.

Next, we investigate the gap between training and inference. On the one hand, entries starting with 1.2 present the case, in which Figure 2.3a and 2.3d are utilized. If test-time augmentation is used, optimizing the model with the baseline method produces inferior results because the classifier is only optimized on feature vectors of single image. By contrast, opting for the FlipReID method resolves such gap, and better scores are reached. On the other hand, entries starting with 2.1 or 3.1 represent the scenario that Figure 2.3b and 2.3c are employed. Choosing the FlipReID method during training while disabling test-time augmentation degrades the performance since the classifier is only optimized on feature vectors of double images. This problem can be addressed by adding the flipping loss, *i.e.*, the resulting method is close to the baseline method without test-time augmentation, and the loss term does not worsen the performance when using double images in inference. On a side note, adopting heavier backbones (*i.e.*, IBN-ResNet [141] and ResNeSt [211]) and applying the re-ranking [222] post-processing contribute to superior results.

2.3.4 Summary

We identified and analyzed the gap between training and inference. Given an image retrieval model such as person re-identification, test-time augmentation is commonly adopted to enhance feature representation without adjusting a trained model. More specifically, feature vectors of the original and flipped images are extracted, and the mean of the feature vectors is treated as the final representation. Nevertheless, this practice leads to a mismatch between training and inference, making the performance sub-optimal. Using the proposed FlipReID architecture, the model can be explicitly optimized on the mean feature vectors that will also be used during inference. In addition, the flipping loss is introduced to prevent possible performance degradation in the case that test-time augmentation is disabled. Experimental results indicate that better image embeddings can be obtained after closing the gap.

2.4 On the Importance of Encrypting Deep Features

2.4.1 Related Work

While remarkable progress has been made in improving the accuracy of machine learning models, identifying vulnerabilities of those models has gained increasing attention [106, 166, 184, 197]. In the following, we provide a high-level overview of the major classes of attacks, including adversarial example attacks, membership inference attacks, model extraction attacks, and model inversion attacks.

In adversarial example attacks, a small perturbation is added to input data in such a way that a human observer could not notice the manipulations while a model would produce erroneous predictions [106]. Such attacks pose a severe challenge for models deployed in safety-critical environments. Previous works have been successful in deceiving models in image classification [57], speech recognition [202], reinforcement learning [80], semantic segmentation and object detection [200].

In membership inference attacks, an algorithm identifies whether a specific sample has been used for optimizing a model [166]. Such attacks damage the interests of two parties, *i.e.*, the individuals whose data has been included in the training set and the owner of the model [185]. In the former case, the privacy of those individuals is violated without consent. In the latter case, the business value or trade secret of the service is leaked.

In model extraction attacks, an adversary seeks to duplicate an existing model with a black-box API [184]. Also known as Machine Learning as a Service (MLaaS), it is common for companies to provide a service so that a user can get predictions of data without training and deploying a model. A substitute model can be obtained by feeding the input samples and corresponding predictions made by the target model. In addition, transferable adversarial examples can be generated to mislead the target model [108].

In model inversion attacks, an adversary is desirous of inferring input data from the target model [197]. Early works have been able to crack relatively simple models, such as predicting private genomic attributes [47] and recovering identifiable images of faces [46]. More recently, it has been shown that training a generative model on public datasets can invert neural networks [215].

2.4.2 Proposed Method

2.4.2.1 Attack Scenarios

Figure 2.4 provides a high level view of the attack scenarios. Following the concept of Machine Learning as a Service (MLaaS), a proprietary model is deployed on a server, and it is trained on a proprietary dataset. Any user (either an ordinary user or an adversary) can send a request to the server, and a response is returned. The request contains raw data such as images, while the response includes feature vectors extracted by the proprietary model. In addition, the adversary has gained unauthorized access to feature vectors of user data, and a local dataset has been collected.

On the one hand, the adversary could train classifiers that recognize auxiliary attributes. Given a facial recognition model, it is also worthwhile determining a person's age and gender. Even though predicting the age and gender is different from recognizing the face, the feature representations still contain relevant information for estimating auxiliary attributes. The main limitation is that the local dataset has to be annotated. Without the supervision signal from the ground truth labels, it is infeasible to optimize the classifiers.

On the other hand, the adversary could train a regression model that reconstructs user data. The system is an autoencoder as a whole, *i.e.*, the encoder is the proprietary model which projects the raw data to an embedding, while the decoder reverts operations in the encoder and generates the reconstructed data. It is unnecessary to annotate samples in the local dataset because the decoder can be optimized in an unsupervised manner. The inputs are feature vectors extracted by the proprietary model, while the ground truth outputs are the raw data.

Conducting the aforementioned attack scenarios is challenging for the following three reasons. Firstly, different datasets are used for optimizing the proprietary and threat models. The domain gap between the proprietary and local datasets would degrade the accuracy of recognition and the quality of reconstruction. Secondly, only the output of the top layer in the proprietary model is provided. The intermediate outputs are unavailable with only the black-box access. Thirdly, there exist notable deviations in the optimization objectives of the proprietary and threat models.

2.4.2.2 ShuffleBits

In most existing studies, parameters and outputs of neural networks are stored using the single-precision floating-point format, in which each number is represented as a 32-bit binary sequence (*i.e.*, binary32). More specifically, the conversation procedure between decimal representation and binary32 format is standardized in IEEE 754 [84].

The binary sequence of a single-precision floating-point number x can be characterized by

$$(a_i)_{i \in I}, \tag{2.8}$$

where $a_i \in \{0, 1\}$, $I = \{1, \ldots, n\}$ and $n = 32$.

Bits in the original sequence can be shuffled based on an encryption key, and the encrypted sequence can be written as

$$(b_j)_{j \in J}, \tag{2.9}$$

where $J = \{1, \ldots, n\}$. The encryption key maps set I to set J, and it is a bijective function $f : I \to J$. Furthermore, $b_{f(i)} = a_i$ holds for $i \in I$.

Likewise, the decrypted sequence can be represented as

$$(c_k)_{k \in K}, \tag{2.10}$$

where $K = \{1, \ldots, n\}$. The decryption key $g : J \to K$ is also a bijective function, and it maps set J to set K. Besides, $c_{g(j)} = b_j$ holds for $j \in J$.

Considering that f is bijective, its inverse function g can be acquired by exchanging the inputs and outputs in f, and we have

$$a_i = b_{f(i)} = c_{g(f(i))} = c_i \text{ for } i \in I. \tag{2.11}$$

Under the condition that the decryption key is the inverse of the encryption key, the original values can be recovered without changes since the decrypted sequence is the same as the original sequence. Figure 2.5 presents a concrete example of ShuffleBits. In the encryption process, a left rotation operation is performed. In the decryption process, a right rotation operation is conducted. It is apparent that the decrypted value equals the original value.

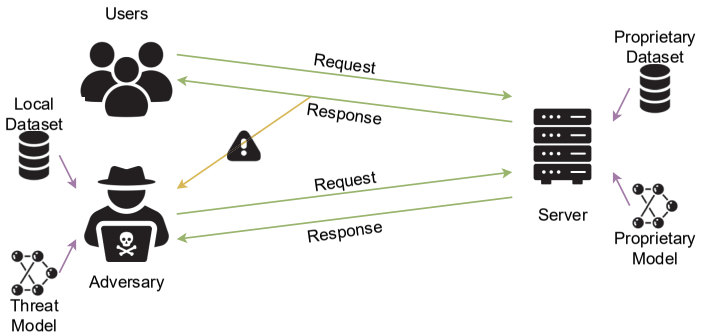


Figure 2.4 One can conduct inference using the proprietary model via a black-box API. In addition, the server's responses to the users have been intercepted by the adversary, and a local dataset has been collected. Reproduced with permission from [137].

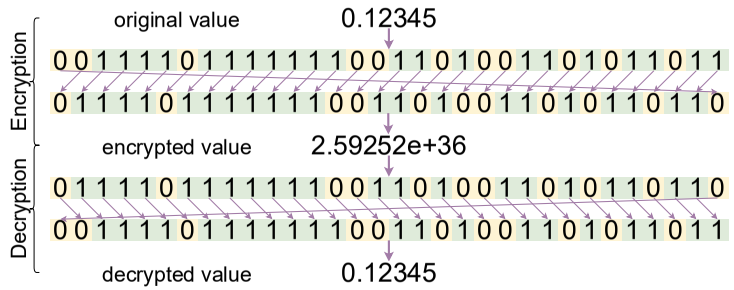
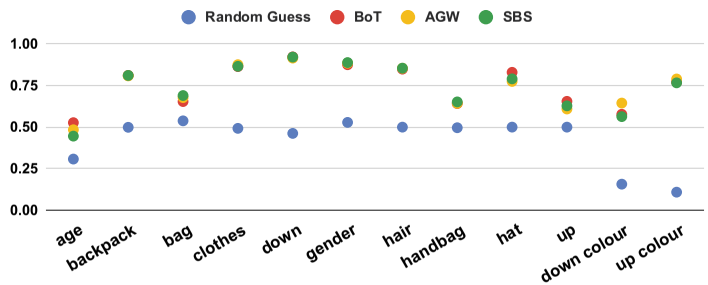
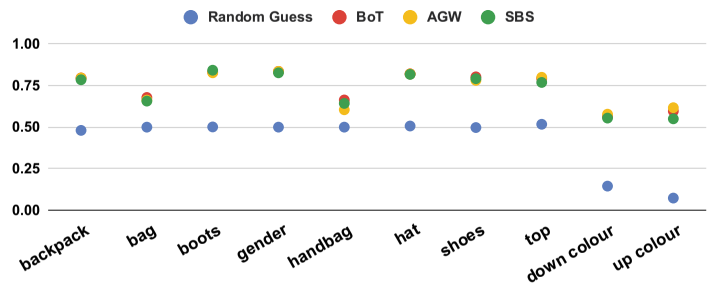


Figure 2.5 A concrete example of ShuffleBits. In the encryption process, a left rotation operation is performed. In the decryption process, a right rotation operation is conducted. Reproduced with permission from [137].



(a) Results on the test set in Market-1501.



(b) Results on the test set in DukeMTMC-reID.

Figure 2.6 The balanced accuracies of each auxiliary attribute. Reproduced with permission from [137].

2.4.3 Experimental Results

2.4.3.1 Background

Model inversion attacks have been performed on notable approaches in person re-identification. Three top-performing models from the FastReID[2] repository are included, namely, BoT [124], AGW [206] and SBS [72]. Considering that models trained on larger datasets are preferred in production, MSMT17 [194] is selected as the proprietary dataset since it contains the highest number of samples. By contrast, smaller datasets such as Market-1501 [219] and DukeMTMC-reID [155] are chosen as the local datasets.

2.4.3.2 Recognizing Auxiliary Attributes

The auxiliary attributes annotated in [116] provide detailed descriptions of pedestrians. Each sample is associated with multiple labels, and we instantiate one batch normalization [85] layer and one dense layer for each label. This relatively simple architecture is consistent with prior works on person re-identification [72, 124, 206], in which a model classifies person identities.

Since the outputs of dense layers denote the probabilities of classes, the cross-entropy loss [216] is applied. Considering that datasets have an uneven distribution of classes, the model would be dominated by the majority of classes, leading to poor classification results. To address the problem of class imbalance, a class weight is assigned to each class, and it is inversely proportional to the number of occurrences [143]. For a similar reason, we adopt balanced accuracy [132] as the evaluation metric, rather than accuracy. On the one hand, accuracy refers to the percentage of correctly predicted samples, and it would give inflated scores when the dataset is imbalanced. On the other hand, balanced accuracy is a more suitable option, and it is calculated by averaging the recall scores on each class.

Figure 2.6 visualizes the balanced accuracies of each auxiliary attribute. Using classifiers trained on the extracted feature vectors leads to significant improvements over random guessing. It shows that coarse-grained estimations can be inferred from deep features.

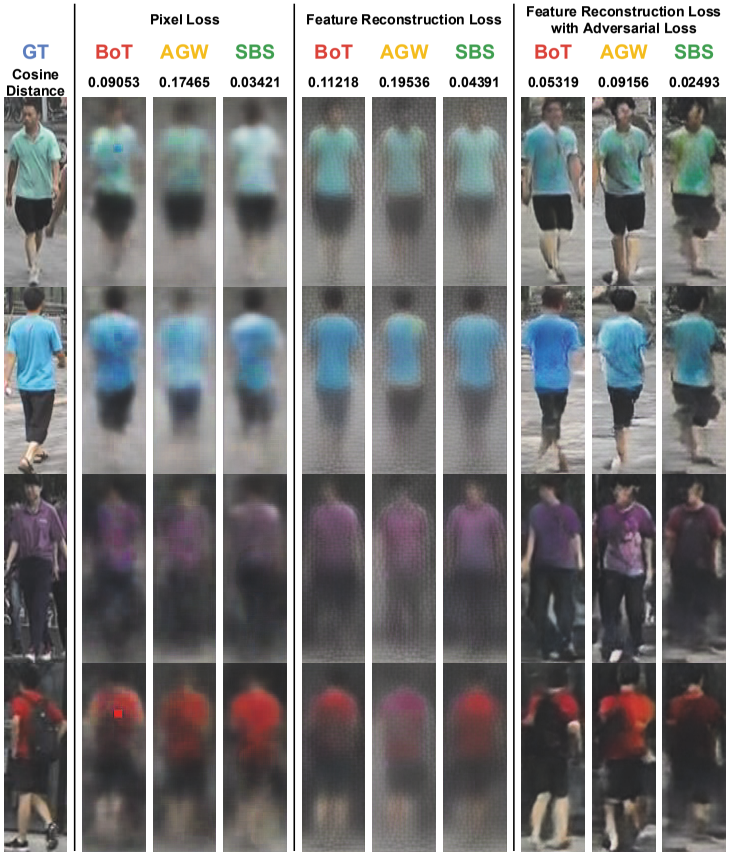
[2] `https://github.com/JDAI-CV/fast-reid`

2.4.3.3 Reconstructing User Data

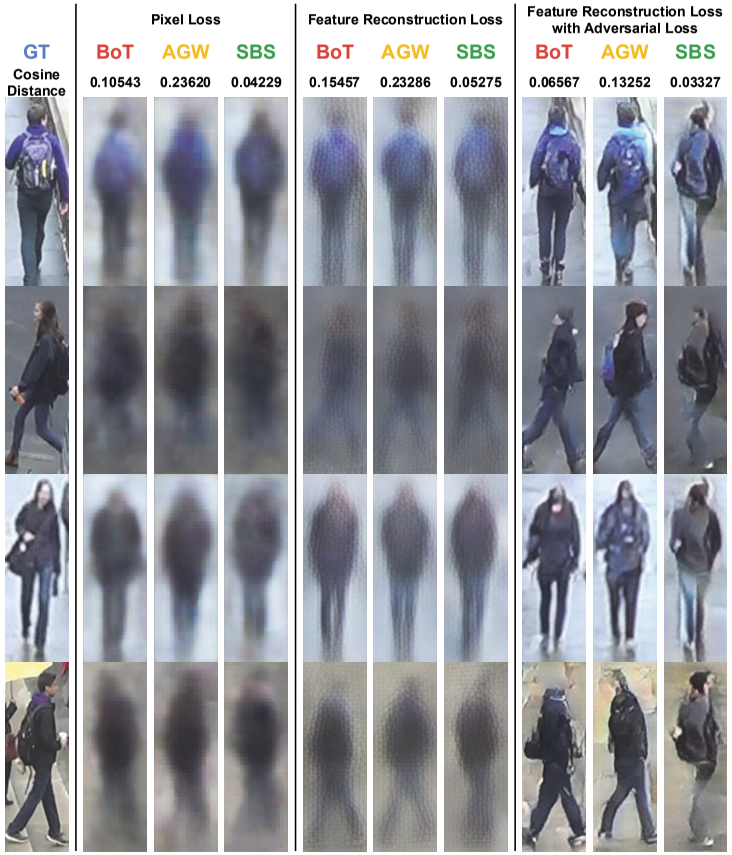
We introduce a GAN-based [56] algorithm for reconstructing user data. On the one hand, the generator reconstructs images from feature vectors. At first, two dense layers with a reshaping operation convert 1-D feature vectors to 3-D feature maps. Afterward, five residual blocks upsample the feature maps to the target resolution. Finally, a convolutional layer is appended, and the number of channels is set to 3 since an RGB image has three channels. The generator is optimized using a weighted sum of pixel loss [93], feature reconstruction loss [93], and adversarial loss [56]. On the other hand, the discriminator predicts whether an input image is real or fake. Five residual blocks reduce the spatial dimensionality of the feature maps, while the subsequent global average pooling layer produces 1-D feature vectors. Later on, a concatenation operation is applied to merge feature vectors extracted by the proprietary model and the discriminator. Last but not least, a dense layer is appended to the concatenated feature vectors, and estimations are generated. The discriminator is optimized using mean squared error loss [128].

Similar to existing studies on person re-identification, the distance between feature vectors of two samples is measured by calculating cosine distance. In order to make features comparable, it is necessary to utilize the same proprietary model for extracting features of the reconstructed images. Afterward, the mean of cosine distance scores between feature vectors of the original and reconstructed images is reported on corresponding test sets.

Figure 2.7 illustrates the reconstructed images under various settings. Different proprietary models, local datasets, and loss functions have been utilized in the experiments. Three observations are presented as follows. Firstly, reconstructions with the pixel loss tend to be blurry. Optimizing the mean squared error between the original and reconstructed images gives an average of all plausible outputs [86]. Secondly, opting for the feature reconstruction loss leads to sharper images while checkerboard artifacts are present. Thirdly, utilizing the feature reconstruction loss along with the adversarial loss generates visually pleasing predictions, in which the checkerboard artifacts are significantly suppressed. In this setting, the cosine distance scores are the lowest, *i.e.*, the reconstructed images are closer to the ground truth images.



(a) Samples from the test set in Market-1501.



(b) Samples from the test set in DukeMTMC-reID.

Figure 2.7 Qualitative and quantitative comparisons of the reconstructed images. Different proprietary models, local datasets, and loss functions have been utilized in the experiments. Reproduced with permission from [137].

2.4.3.4 Importance of Encrypting Deep Features

Deriving from deep features extracted by a proprietary model, an adversary can carry out model inversion attacks, in which sensitive information is inferred. The state-of-the-art models in person re-identification are included. Results demonstrate that it is achievable to recognize auxiliary attributes with admissible accuracy and obtain a reasonable reconstruction of user data.

Based on the results mentioned above, adopting an encryption method is essential, particularly for models deployed on a production server. An encryption key has to be included in each request, along with user data. Afterward, the server returns feature vectors in encrypted form, and the original values can be retrieved using the decryption key. On the one hand, threat models trained on original feature vectors could not produce reasonable predictions on encrypted feature vectors. On the other hand, optimizing threat models using encrypted feature vectors is meaningless since the encryption keys used by the users and the adversary differ.

2.4.3.5 ShuffleBits vs Traditional Encryption Methods

Compared with traditional encryption methods, the proposed method has several unique characteristics. The ShuffleBits module consists of operations that can be easily implemented with existing deep learning frameworks, such as TensorFlow [1] and PyTorch [144]. It can be integrated into any neural network as a plug-and-play component without the need to add extra dependencies. Furthermore, the resulting model would directly output feature vectors in encrypted form. For the reason that data has to be transferred between GPU and CPU, it is beneficial to keep data encrypted.

Among the symmetric-key algorithms in cryptography, Advanced Encryption Standard (AES) [29] is the de facto standard. By contrast, validating the security aspect of ShuffleBits is left out for future research. We introduce two workarounds to prevent potential vulnerabilities in ShuffleBits. On the one hand, it is worthwhile to utilize the one-time pad scheme, in which different encryption/decryption keys are used in each request. The adversary could only observe a few instances that are encrypted with the same key. On the other hand, cascade encryption can be adopted, and a feature vector is encrypted by both ShuffleBits and a traditional encryption method.

2.4.4 Summary

We have conducted model inversion attacks against state-of-the-art models in person re-identification. Two assumptions are made, *i.e.*, feature vectors of user data are accessible, and a black-box API is available for conducting inference. Experimental results indicate that auxiliary attributes can be classified with admissible accuracy, and a reasonable reconstruction of user data can be generated. As a countermeasure, we urge practitioners to utilize an encryption algorithm when transferring and storing deep features. Implemented using the one-time pad scheme, the ShuffleBits module can be seamlessly incorporated into any existing neural network, and the extracted feature vectors are in encrypted form.

3 VEHICLE ATTRIBUTE RECOGNITION

In this chapter, we consider the task of extracting distinctive image embeddings in vehicle attribute recognition. Section 3.1 includes the background information, including dataset and evaluation metric. Section 3.2 provides a concise overview of existing methods and formulates a novel pipeline that is preferable in a dynamic environment.

3.1 Background

Vehicle attribute recognition refers to the study of recognizing attributes that are associated with vehicles. Figure 3.1 shows a concrete example in which vehicle type, vehicle make, and vehicle model are identified. Deriving from an image that captures the appearance of a vehicle, we analyze the following problem settings:

- Vehicle Type Recognition: Depending on factors such as size, seating and body style, vehicle type indicates the intended use. Common vehicle types include motorcycle, sedan, van, truck, and bus. It provides insight into the distribution of different vehicle classes in a region.
- Vehicle Make Recognition: Also known as the manufacturer, vehicle make stands for the brand of a vehicle, *e.g.*, Audi, Honda and Toyota. It can be applied to find the optimal location of a car dealership, *i.e.*, choosing an area in which most drivers favor a specific vehicle make.
- Vehicle Model Recognition: Apart from the manufacturer's information, vehicle model provides a fine-grained description, *i.e.*, the specific model such as Audi A6, Honda Civic and Toyota Corolla. It can be utilized to conduct in-depth analyses of consumer behavior. As new products are introduced frequently, there exists the challenge of updating the machine learning model in a dynamic environment.

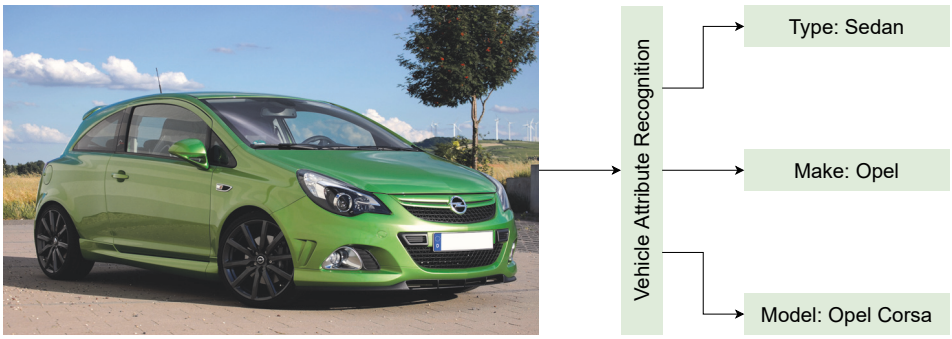


Figure 3.1 Illustration of vehicle attribute recognition, in which vehicle type, vehicle make and vehicle model are recognized. Photo by P.A.J.

Table 3.1 Popular datasets for vehicle attribute recognition, sorted by the year of publication. -: not available. Reproduced with permission from [136].

Dataset	Year	#Image	#Type	#Make	#Model
Petrovic and Cootes [148]	2004	1,132	-	-	77
Clady *et al.* [24]	2008	1,121	-	-	50
car-types [173]	2011	1,904	-	-	14
Peng *et al.* [146]	2012	4,924	5	-	-
BMW-10 [103]	2013	512	1	1	10
car-197 [103]	2013	16,185	7	-	197
FG3DCar [115]	2014	300	-	-	30
Liao *et al.* [113]	2015	1,482	-	8	-
BIT-Vehicle [41]	2015	9,850	6	-	-
CompCars [203]	2015	214,345	12	161	1,687
Huttunen *et al.* [83]	2016	6,555	4	-	-
BoxCars [170]	2016	63,750	-	27	148
BoxCars116k [171]	2018	116,286	-	45	693
MIO-TCD [125]	2018	648,959	11	-	-
VERI-Wild [122]	2019	416,314	14	149	-

3.1.1 Dataset

Table 3.1 lists a variety of datasets for vehicle attribute recognition. Three attributes are covered, varying from coarse-grained level (*i.e.*, vehicle type) to fine-grained level (*i.e.*, vehicle make and vehicle model).

Before the advent of deep learning techniques, most datasets in the early works are of limited size [24, 41, 83, 103, 113, 115, 146, 148, 173]. As one of the first attempts, 1,132 frontal images from 77 vehicle models are available in [148]. In a similar way, high-quality frontal images have been captured under daylight and nightlight in [146], and they are categorized into 5 vehicle types. Last but not least, it is of notable mention that the car-197 [103] dataset comprises a large number of images with annotations of vehicle type and vehicle model.

On account of the increasing interest in studying vehicle attribute recognition, more diverse datasets have been collected in recent works [122, 125, 170, 171, 203]. Firstly, the MIO-TCD [125] dataset marks a milestone for vehicle type recognition, and there are 648,959 images in total. Secondly, the VERI-Wild [122] dataset consists of 416,314 images captured under unconstrained scenarios. Initially designated for vehicle re-identification, it also features annotations of vehicle type and vehicle make. Thirdly, the CompCars [203] dataset includes labels of three attributes, making it an ideal option for the setting of Multi-Task Learning (MTL). In general, opting for a large-scale dataset in the wild is advisable since it enables future-proof applications by closing the gap between research and practice.

3.1.2 Evaluation Metric

For each label (*i.e.*, vehicle attribute), a model estimates the probabilities of all classes, and the most probable class is found for each sample. The score of classification accuracy refers to the percentage of correctly classified samples:

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} \delta(y_i, \hat{y}_i), \tag{3.1}$$

where i is the index of current sample among n samples, y_i is the ground truth label, $\hat{y}_i$ is the predicted label, and δ denotes the Kronecker delta function.

3.2 Vehicle Attribute Recognition in a Dynamic Environment

3.2.1 Vehicle Type Recognition

Table 3.2 summarizes representative algorithms for vehicle type recognition, including both the hand-crafted and deep learning approaches.

3.2.1.1 Hand-Crafted Approaches

Hand-crafted features can be constructed from a region of interest, *e.g.*, a bounding box that shows a vehicle. Calculating a set of low-level features (*e.g.*, height, width, and angle) is found to be effective in [95]. In addition, the SIFT [123] descriptor is applied in [40, 151], while the HOG [31] descriptor is adopted in [9]. Furthermore, the eigenvectors with the largest eigenvalues are selected in [145, 146]. Finally, local structure operators [148] and Gabor wavelet kernels [175] can be utilized.

Besides the differences in feature representations, there is a wide variety of classifiers to choose from. It is straightforward to take the label from the k-nearest neighbors as in [146, 148, 175]. Alternatively, higher accuracy can be obtained by incorporating the k-means [120] algorithm and finding representative centroids [146]. Other options include SVM [9, 145], neural network [151], multiple kernel learning [40] and dynamic Bayesian network [95].

3.2.1.2 Deep Learning Approaches

The availability of the MIO-TCD [125] dataset has accelerated the shift towards algorithms based on neural networks. In [98], multiple models are trained on different portions of the training set, and a voting process aggregates predictions of a test sample. Similarly, an ensemble of models with different backbones is employed in [178], while the most appropriate model is identified by utilizing a rule-based gating function [87]. Moreover, adopting ensembling on two independently trained models delivers a performance boost in [180]. As a regularization strategy, only a subset of models is optimized at each iteration in [94]. Last but not least, spatial pyramid features are extracted from several crops of the input, and a recurrent model is appended [152].

Table 3.2 Notable approaches on vehicle type recognition, grouped by the dataset. Reproduced with permission from [136].

Approach	Year	Dataset	#Image	#Type	Accuracy	Notes
Petrovic and Cootes [146, 148]	2004	Peng *et al.* [146]	4,924	5	84.3%	Hand-Crafted
Psyllos *et al.* [146, 151]	2011				78.3%	Hand-Crafted
Peng *et al.* [146]	2012				90.0%	Hand-Crafted
Dong and Jia [40]	2013				91.3%	Hand-Crafted
Peng *et al.* [145]	2013				93.7%	Hand-Crafted
Dong *et al.* [41]	2015				96.1%	Deep Learning
Kafai and Bhanu [95]	2012	Kafai and Bhanu [95]	845	4	96.6%	Hand-Crafted
Petrovic and Cootes [9, 148]	2004	BIT-Vehicle [41]	9,850	6	78.6%	Hand-Crafted
Psyllos *et al.* [9, 151]	2011				70.8%	Hand-Crafted
Peng *et al.* [9, 145]	2013				85.0%	Hand-Crafted
Dong *et al.* [41]	2015				88.1%	Deep Learning
Sun *et al.* [175]	2017				90.1%	Hand-Crafted
Yang *et al.* [203]	2015	Subset of CompCars [203]	52,083	12	63.1%	Deep Learning
Huttunen *et al.* [83]	2016	Huttunen *et al.* [83]	6,555	4	97.8%	Deep Learning
He *et al.* [70, 152]	2015	MIO-TCD [125]	648,959	11	96.5%	Deep Learning
Kim and Lim [98]	2017				97.8%	Deep Learning
Lee and Chung [178]	2017				97.9%	Deep Learning
Jung *et al.* [94]	2017				97.9%	Deep Learning
Theagarajan *et al.* [180]	2017				97.8%	Deep Learning
Rachmadi *et al.* [152]	2018				97.9%	Deep Learning

3.2.2 Vehicle Make and Model Recognition

Table 3.3 and Table 3.4 categorize prominent approaches on vehicle make recognition and vehicle model recognition, respectively.

3.2.2.1 Hand-Crafted Approaches

Early works are conducted on frontal images, using features extracted by local structure operators [148] or oriented-contour points [24]. To handle more complicated cases, approaches [113, 173] based on DPM [44] have been popular. In addition, leveraging the crowd-selected bubbles to detect discriminative regions [34] and switching to 3D object representations [103] bring significant improvements. Finally, two feature descriptors (*i.e.*, HOG [31] and FV [147]) are applied on a window centered around each landmark in [115].

The NNS [100] approach has been adopted in early works [24, 148]. The predicted class of a test sample is assigned after finding its top k nearest neighbors. By contrast, the SVM [26] approach is the prominent option among recent works [34, 103, 113, 115, 173]. Training samples are mapped to feature spaces so that the gap between different categories gets maximized.

3.2.2.2 Deep Learning Approaches

Apart from introducing the CompCars dataset, a baseline solution is presented in [203], *i.e.*, fine-tuning a pre-trained model for vehicle attribute recognition. Later on, a spatially weighted pooling scheme is adopted in [79], while the interaction between parts is taken into account in [199]. Subsequently, attention is shifted towards the BoxCars [170] and BoxCars116k [171] datasets. In [170], the performance is boosted by supplementing auxiliary information, including rasterized low-resolution shape, 3D vehicle bounding box, and 3D vehicle orientation. In [171], images are normalized based on the 3D bounding boxes, and additional data augmentation policies are applied. Additionally, a framework is proposed in [209] to extract a joint representation of 2-D global texture and 3-D bounding box, while a feature fusion network merges the aforementioned features. Last but not least, more general approaches on fine-grained image classification have been validated on vehicle attribute recognition as well [48, 51, 88, 114, 169].

Table 3.3 Notable approaches on vehicle make recognition, grouped by the dataset. Reproduced with permission from [136].

Approach	Year	Dataset	#Image	#Make	Accuracy	Notes
Liao *et al.* [113]	2015	Liao *et al.* [113]	1,482	8	81.3%	Hand-Crafted
Yang *et al.* [203]	2015	Subset of CompCars [203]	30,955	75	82.9%	Deep Learning
Hu *et al.* [79]	2017				99.3%	Deep Learning
Xiang *et al.* [199]	2019				99.6%	Deep Learning

Table 3.4 Notable approaches on vehicle model recognition, grouped by the dataset. Reproduced with permission from [136].

Approach	Year	Dataset	#Image	#Model	Accuracy	Notes
Petrovic and Cootes [148]	2004	Petrovic and Cootes [148]	1,132	77	93%	Hand-Crafted
Clady *et al.* [24]	2008	Clady *et al.* [24]	1,121	50	93.1%	Hand-Crafted
Stark *et al.* [173]	2011	car-types [173]	1,904	14	93.5%	Hand-Crafted
Krause *et al.* [103]	2013				94.5%	Hand-Crafted
Krause *et al.* [103]	2013	BMW-10 [103]	512	10	76.0%	Hand-Crafted
Krause *et al.* [103]	2013	car-197 [103]	16,185	197	67.6%	Hand-Crafted
Krause *et al.* [102]	2015				92.8%	Deep Learning
Hu *et al.* [79]	2017				93.1%	Deep Learning
Xiang *et al.* [199]	2019				94.3%	Deep Learning
Lin *et al.* [115]	2014	FG3DCar [115]	300	30	90.0%	Hand-Crafted
Yang *et al.* [203]	2015	Subset of CompCars [203]	52,083	431	76.7%	Deep Learning
Hu *et al.* [79]	2017				97.6%	Deep Learning
Xiang *et al.* [199]	2019				98.5%	Deep Learning
Jaderberg *et al.* [88, 209]	2015	Subset of BoxCars [170]	59,742	77	64.3%	Deep Learning
Sochor *et al.* [170]	2016				75.4%	Deep Learning
Fu *et al.* [48, 209]	2017				72.2%	Deep Learning
Zeng *et al.* [209]	2019				81.2%	Deep Learning
Lin *et al.* [114, 171]	2015	Subset of BoxCars116k [171]	90,840	107	69.6%	Deep Learning
Simon and Rodner [169, 171]	2015				75.9%	Deep Learning
Gao *et al.* [51, 171]	2016				70.6%	Deep Learning
Sochor *et al.* [171]	2018				84.1%	Deep Learning

3.2.3 Proposed Method

Given an image classification [104] model, the last dense layer produces the probabilities of each class, and the class with the highest probability is chosen as the prediction. Naturally, such model could only possibly work on a test sample with a known class, *i.e.*, the ground truth label must be included in the training set. Otherwise, predictions would be meaningless on samples out of the scope.

By contrast, a different pipeline is employed in an image retrieval system (*e.g.*, person re-identification [12]). Feature representations are extracted for both the query and gallery samples, and a distance metric measures the distance between each query-gallery pair. Also known as Cumulative Matching Characteristic (CMC) rank-k accuracy [192], one may check whether a correct match for the query sample is found among the top-k gallery samples. Note that the ground truth label of a test sample does not have to be included in the training set, *e.g.*, person identities in the training and test sets do not overlap.

To keep up with customer expectations, automobile manufacturers launch new products into the market regularly. Considering a vehicle attribute recognition system in practice, adopting the image classification approach is unsuitable. The underlying model must be frequently retrained while collecting supplementary samples of reasonable size for the new products is time-consuming. Conversely, the image retrieval approach is still applicable in a dynamic environment. In the case that a new product is introduced, exemplary images can be appended to the gallery set, and it is unnecessary to update the model.

As discussed in Section 3.2.1 and 3.2.2, remarkable progress has been achieved in recognizing attributes that are associated with vehicles. Compared with other well-studied areas (*e.g.*, face recognition [191] and person re-identification [12]), there is still room for improvement in vehicle attribute recognition. Some technological advances have not been validated in the study of vehicles, for example, triplet loss [165] and random erasing data augmentation [223]. In addition, the recently collected large-scale datasets (*e.g.*, CompCars [203], MIO-TCD [125] and VERI-Wild [122]) enable a more comprehensive investigation that can be further extended into a real-world application. In the following, we explain the details of an algorithm built on top of the state-of-the-art method in person re-identification [124].

The ResNet50 [71] architecture is adopted as the backbone, and the pre-trained weights on the ImageNet [33] dataset are used for initialization. Layers after the global average pooling layer are discarded because those are exclusive to ImageNet. Afterward, a batch normalization [85] layer and a dense layer with the softmax activation function are appended at the end. As the output of the dense layer refers to the probabilities of each class, the categorical cross-entropy loss [216] function is utilized. Alternatively, the model can also be optimized via metric learning, *e.g.*, triplet loss [165] and lifted structured loss [139]. The distance between an anchor and a positive gets minimized, while the distance between an anchor and a negative gets maximized. Such loss functions are applied to the output of the global average pooling layer.

In the training procedure, five different strategies have been integrated:

- In the last building block of the backbone, the stride argument in the first convolutional layer is set to 1. It results in larger feature maps while the pre-trained weights can still be loaded.
- The random erasing [223] policy is used in data augmentation. A randomly selected area is masked out, and the model would be able to generalize better.
- The one-hot encoding of labels is processed by label-smoothing [177]. It allows better tolerance to possible noise in the annotations.
- Imposing L_2 regularization drives the parameters toward zero, and it helps in reducing the overfitting issue [121].
- The learning rate is initialized with a low value and increases linearly over the warm-up period. It suppresses the issue with distorted gradients [118].

In the inference procedure, the output of the global average pooling layer is extracted. The gallery set is instantiated by randomly choosing 100 training samples from each class. Additionally, the distance between two feature vectors is measured by calculating the cosine distance. Under the setting of image retrieval, the test set is viewed as the query set, and a distance matrix is calculated by comparing each query-gallery pair. Similar to the nearest neighbor search (NNS) [100] method, the label of the nearest neighbor in the gallery set is taken as the predicted class, and accuracy is reported as the performance measure.

Table 3.5 Results on VERI-Wild [122] are summarized. Reproduced with permission from [136].

Approach	#Type	Accuracy	#Make	Accuracy
cross-entropy loss [216]		96.8%		95.6%
triplet loss [165]	14	97.4%	149	95.3%
lifted structured loss [139]		**97.7%**		**96.2%**

3.2.4 Experimental Results

Table 3.5 presents the results on the VERI-Wild [122] dataset. Choosing VERI-Wild has two advantages, *i.e.*, it is a large-scale dataset consisting of approximately 0.4 million images, and it has annotations of vehicle type and vehicle make. Apart from the categorical cross-entropy loss [216] function, the performances of using triplet loss [165] and lifted structured loss [139] are reported. On the one hand, accuracy is higher than 95.0% under all experimental settings. While there is always room for improvement, it indicates that vehicle attribute recognition is partially solved, and the focus can be shifted to monetizing the technology. On the other hand, utilizing the lifted structured loss function obtains higher accuracy scores than the other two alternatives. Therefore, more research can be done in the direction of improving metric learning.

3.2.5 Summary

We presented a comprehensive study on extracting image embeddings in vehicle attribute recognition. Three attributes are considered, namely, vehicle type, vehicle make, and vehicle model. First of all, a comparison among popular datasets is conducted. The CompCars [203], MIO-TCD [125] and VERI-Wild [122] datasets are the preferable options at the moment. Subsequently, a survey is carried out by researching notable methods, including hand-crafted and deep learning approaches. Finally, the state-of-the-art method in person re-identification [124] is ported for identifying vehicle attributes. Since an image classification model would not generate a meaningful prediction for a sample with an unseen class, formulating vehicle attribute recognition as an image retrieval problem is more suitable, especially in a dynamic environment.

4 NEURAL IMAGE COMPRESSION

In this chapter, we investigate how to extract compressible image embeddings that would be used to reconstruct the original images. Section 4.1 provides the background information of our study, while Section 4.2 summarizes an autoencoder based algorithm for image compression [7].

4.1 Background

By virtue of data compression, a compact representation of the original data can be generated so that fewer bits are required to store and transmit the compressed data [163]. It is a ubiquitous method that can be found in Information Technology (IT) systems, *e.g.*, when handling multimedia content such as images, audio, and videos. Given a one second long video in the 1080p 60 fps format, the raw images would take $1920 * 1080 * 3 * 8 * 60 * 1$ bits $\approx 2.99e9$ bits ≈ 0.37 gigabytes. Using the original data without compression is infeasible since it would pose a heavy burden on storage devices and computer networks. By contrast, adopting H.265 can decrease the file size by a factor between 300 to 1,000 [174].

A compression algorithm can be separated into two parts: an encoder (compressor) accepts the original data and generates a compact representation, while a decoder (decompressor) performs the reversal and produces a reconstruction. Depending on whether the reconstruction is identical to the original data, a compression algorithm can be classified as either lossless or lossy [163]. On the one hand, lossless compression would recover the original data precisely. It is suitable for scenarios where no deviation is acceptable, *e.g.*, compressing databases and text documents. On the one hand, lossy compression allows a certain level of deviations. It enables a higher compression rate at the cost of lower quality and has been widely applied in compressing multimedia content.

(a) Photo by Alexander Shustov.

(b) Photo by Stefan Kunze.

Figure 4.1 Sample images from the CLIC [19] dataset.

4.1.1 Dataset

Our study is conducted on the dataset that is provided in Challenge on Learned Image Compression (CLIC) [19], and two sample images are illustrated in Figure 4.1. The CLIC dataset consists of approximately 2,000 high-quality images taken under a wide variety of environments. Those images are chosen based on the criteria that they are representative of images used in practice so that the experimental protocol would be close to a real-life setting. Images are saved using the Portable Network Graphics (PNG) format in a lossless manner. Optionally, other image datasets can be utilized as well, *e.g.*, the ImageNet [33] dataset.

4.1.2 Evaluation Metric

Image Quality Assessment (IQA) refers to the measurement of the level of accuracy in images. Given an image compression algorithm, the original images are also the ground truth images, and they can be treated as reference images with perfect quality. The mean squared error (MSE) between the original images and the decompressed images is calculated over all pixels of all samples in the RGB color space. Subsequently, the Peak Signal-to-Noise Ratio (PSNR) score is obtained based on the MSE score, and it gives an estimation of image quality. Note that using MSE and PSNR might produce misleading results in certain cases [161], *e.g.*, shifting the original images by one pixel maintains similar visual quality while the MSE score could be excessively high.

4.2 Block-Optimized Variable Bit Rate Neural Image Compression

4.2.1 Related Work

In light of recent advances in deep learning, neural network techniques have been applied in the field of data compression. Depending on whether a traditional codec (*e.g.*, JPEG [187] and BPG [13]) is present, there are two types of methods, namely, hybrid and end-to-end approaches.

Given a lossy image compression algorithm, encoding at low bit rates might yield perceivable artifacts in the reconstructions, such as aliasing, blurring, checkerboarding, posterizing, and ringing. Categorized as hybrid approaches, machine learning techniques can be used along with traditional codecs, and a transformation function is learned to map the degraded images back to the original images. At first, four convolutional layers are adopted in [39], and the model performs feature extraction, feature enhancement mapping, and reconstruction in sequential order. Later on, feature representations are extracted in both the spatial domain and the frequency domain [60]. In addition, clear improvements can be obtained by adding hierarchical skip connections and optimizing the model at multiple scales [17]. Furthermore, other works explore possibilities for enabling deeper models [127], using generative adversarial networks [50] and improving the objective function [61].

An end-to-end approach differs in the sense that no traditional codec is applied, and a neural network is optimized on unprocessed images. Such method can avoid the drawbacks of existing codecs, while new challenges must be addressed since the model has to extract compressible image embeddings directly. Both AutoEncoder [11, 181] and Recurrent Neural Network [182, 183] are popular architectures in deep image compression. Feature vectors generated by those models are losslessly compressed by using entropy encoding (*e.g.*, arithmetic coding [195] and Huffman coding [81]) [2]. Binarization can be performed during training so that features get simplified and reaching a reasonable compression ratio becomes feasible. As its derivative is either zero or undefined, possible solutions include using randomized quantization [182], adding uniform noise [10], and utilizing a smooth approximation [181].

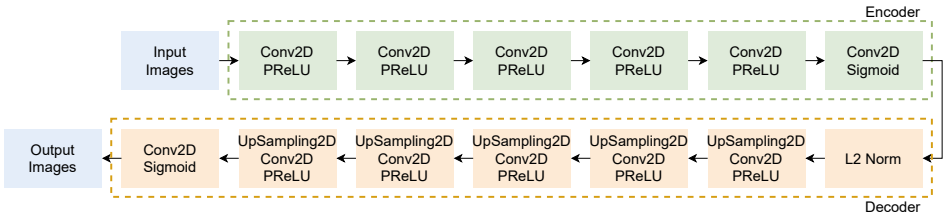


Figure 4.2 A high-level overview of the compression network. The encoder extracts feature vectors of the input images using a sequence of convolutional and activation layers. The decoder consists of normalization, upsampling, convolutional, and activation layers. The major challenge is to minimize the differences between the input and output images while keeping the feature vectors compact.

4.2.2 Proposed Method

In the JPEG [187] standard, an image is split into blocks of 8 x 8 pixels. By the same token, adopting a block-based method is preferable because training a neural network requires accumulating inputs of a fixed size within one batch. Additionally, the terminologies of encoder/decoder in a compression algorithm and an autoencoder show substantial similarities. In the following, we introduce an end-to-end image compression algorithm built on autoencoders.

As illustrated in Figure 4.2, the compression network is an autoencoder that operates on 32 x 32 blocks. On the one hand, the encoder has five blocks, while each block contains a 2-D convolutional layer (with stride set to 2) and the PReLU [69] activation. The resulting tensor is further processed by a 1 x 1 convolutional layer and the sigmoid [64] activation. Eventually, the encoder's output (referred to hereafter as encoded features) is a flattened feature vector in the range of $(0, 1)$. On the other hand, the decoder is a reversal of the encoder. As suggested in [6], a L_2 normalization layer is applied at first to map features to a unit hypersphere. Subsequently, a block composed of an upsampling layer, a 2-D convolutional layer, and the PReLU activation is repeated five times. While each upsampling layer increases the height/width of the feature maps by a factor of two, the resulting tensor has the same spatial size as the encoder's input. Finally, a 1 x 1 convolutional layer and the sigmoid [64] activation are appended at the end. The method falls under the category of unsupervised learning as the dataset does not need to be annotated, and the ground truth image is identical to the input image.

Naturally, the encoded features have to be compressed in a lossless manner. Without enforcing any constraints, each number would consume 32 bits, and it remains challenging to compress those single-precision floating-point numbers efficiently. As an alternative, we aim to derive a binarized feature vector consisting only of 0 and 1. However, integrating a non-differentiable binarization operation inside a neural network results in optimization difficulty because its derivative is either zero or undefined. This problem can be addressed by simulating the binarization process as follows: given a scalar x_i in a feature vector $\mathbf{x} = (x_1, x_2, x_3, \cdots, x_n)$, random noise sampled from a continuous uniform distribution with the interval $[-a_i, a_i]$ is added to x_i, where $a_i = |rint(x_i) - x_i|$ and the *rint* function rounds x_i to the nearest integer. After the initial training of the encoder and decoder is completed, another round of training is performed to reduce the inconsistency. More specifically, the encoder gets frozen, simulated binarization is replaced by actual binarization, and the decoder is further fine-tuned.

While the feature vector $\mathbf{x}$ is binarized, we introduce the entropy-friendly loss that makes the data more compressible:

$$L_{\text{entropy-friendly}} = \frac{1}{n+1}((x_1 - 0)^2 + \sum_{i=2}^{n}(x_i - x_{i-1})^2 + (0 - x_n)^2), \tag{4.1}$$

where n refers to the length of $\mathbf{x}$. Note that the feature vector gets zero-padded on both sides, and the difference between adjacent elements is minimized.

The encoder in a compression network generates encoded features of fixed size as the number of filters in the last convolutional layer is hard-coded. Such property is undesirable as a specific encoder might be too light for a complex block or too heavy for a simple block. To add support for Variable BitRate (VBR), we employ multiple compression networks that extract encoded features of different lengths. For a specific block, the compression network that reaches the target PSNR score and gives the smallest bit rate is chosen.

When deploying a machine learning model, no additional training is necessary as the model is already trained. However, this is not the case for an image compression algorithm. We incorporate a code optimization procedure during inference, *i.e.*, the encoder in a compression network gets optimized for a specific block while the decoder is frozen. In other words, the encoder is overfitted to a particular test sample, and the encoded features are optimized accordingly.

The original image is divided into non-overlapping blocks using raster scanning, while the compression network encodes and decodes each block separately. In consequence of discontinuities at block boundaries, the checkerboard artifact is clearly visible in the reconstructed images. To suppress such artifacts, we append a separate deblocking network. It shares a similar structure with U-Net [157], with skip connections bridging the encoder and decoder. The inputs are the reconstructed images generated by the compression network, while the ground truth outputs are the original images. Due to the very principle of being fully convolutional, it is feasible to generate the prediction for the entire image in one go.

Last but not least, we summarize the three essential elements that are required for reconstructing images in the decompression process. Firstly, the encoded features of all blocks are concatenated, and entropy encoding is applied to the difference between adjacent elements (see Equation 4.1). Secondly, it is mandatory to have an indicator of which compression network to use so that the proper model can be selected. Thirdly, the resolution of the original image has to be included. After reconstructing all blocks, each block will be placed to the correct coordinate based on its order and the image resolution.

4.2.3 Experimental Results

Table 4.1 shows an ablation study of the baseline method, in which only one compression network is used, and the length of encoded features is set to 216. More specifically, the contributions of the main components are analyzed. In each experiment, one component is removed from the baseline method, and the performance of the resulting model is reported. Firstly, it is essential to simulate the binarization process in the initial stage and switch to actual binarization afterward. These two components contribute to deriving distinctive feature representations that are binarized. Secondly, inserting a L_2 normalization layer at the beginning of the decoder leads to better image quality at the cost of a larger file size. We observe that a model without L_2 normalization can obtain comparable scores (*i.e.*, PSNR and bpp) as its counterpart, however at higher encoding dimension of 236. In other words, similar performance can be maintained with a lighter model by adopting a L_2 normalization layer. Thirdly, adding entropy-friendly loss during training significantly improves the efficiency of entropy encoding since a sharp decrease in the file size can be observed.

Table 4.1 Analysis of the contributions of the main components in the baseline method. Performance is evaluated on the validation set. Reproduced with permission from [7].

Method	Peak Signal-to-Noise Ratio	bits per pixel
Baseline w/o simulated binarization	23.882	-
Baseline w/o actual binarization	25.751	-
Baseline w/o L_2 normalization	26.778	0.134
Baseline w/o entropy-friendly loss	27.258	0.216
Baseline	27.055	0.151

Table 4.2 Effects of additional procedures on top of the baseline method. Performance is evaluated on the validation set. Reproduced with permission from [7].

Method	Peak Signal-to-Noise Ratio	bits per pixel
Baseline	27.055	0.151
+ multiple compression networks	27.691	-
+ sample selection	27.779	-
+ fine-tuning decoder	27.792	-
+ deblocking network	28.088	-
+ code optimization	28.929	0.149

The baseline method can be extended by incorporating additional procedures. Table 4.2 shows the detailed results, and each experiment is an increment of the previous one. Training multiple compression networks enables support for Variable BitRate so that the most suitable model can be selected for the block at hand. As a result, the PSNR score increases substantially. In addition, sample selection can be added, *i.e.*, samples of suitable difficulty are fed to the compression network in question. Light models will not be trained on complex blocks, while heavy models will not be trained on simple blocks. Likewise, the decoders can also be fine-tuned on selected blocks. Nevertheless, sample selection and fine-tuning decoder only yield marginal improvements. A plausible explanation is that the model is more prone to overfitting as the training set becomes less diverse. By contrast, the deblocking network generates visually pleasing images by suppressing the checkerboard artifact, and a higher PSNR score is reached. Finally, code optimization results in significant gains as the encoded features are specially optimized for each block. In summary, the PSNR score increases by 1.874 after utilizing these procedures, while bits per pixel remains comparable.

Table 4.3 Comparison with two traditional codecs on the test set. Reproduced with permission from [7].

Method	Peak Signal-to-Noise Ratio	bits per pixel
JPEG [187]	25.612	0.149
BPG [13]	29.587	0.148
Ours	27.920	0.148

In Table 4.3, the proposed method is compared with traditional codecs including JPEG [187] and BPG [13]. As specified in the requirements of [19], bits per pixel in all entries are close to the threshold 0.15. In term of the PSNR scores, our method lies between JPEG and BPG.

4.2.4 Summary

We introduced an autoencoder-based algorithm for neural image compression. The key challenge is deriving image embeddings that are compressible. Our major contributions include handling the binarization process, exploiting L_2 normalization, incorporating the entropy-friendly loss, training multiple compression networks, employing a deblocking network, and optimizing the encoded features. As one of the early attempts in compressing images with neural networks, we manage to obtain better image quality than JPEG that is the de facto standard for lossy image compression.

5 CLUSTERING AND UNSUPERVISED ANOMALY DETECTION

In this chapter, we study image embeddings in two related areas in Unsupervised Learning (UL), *i.e.*, clustering and unsupervised anomaly detection. Background information is explained in Section 5.1, and a more detailed investigation [6] can be found in Section 5.2.

5.1 Background

Clustering (or cluster analysis) provides insights into the data by categorizing its internal organization as a hierarchy of groups or a grouping of individuals [89]. The fundamental principle is that samples within the same group are closer to each other than samples from the other groups. It has practical applications in many areas, including bioinformatics [226], recommender system [162], customer segmentation [196], image analysis [21] and social network analysis [15]. Due to the absence of ground truth labels that constitute target values, clustering is an essential topic in Unsupervised Learning (UL). There are two major challenges, *i.e.*, defining a distance function [49] and measuring the performance of clustering [154].

Anomaly detection refers to identifying samples that do not exhibit the same pattern as the other samples [18]. Depending on the context, those instances could be anomalies, outliers, exceptions, surprises, or contaminants [20]. In accordance with the definition of anomaly, a common assumption is that the number of abnormal instances is considerably lower than normal instances. Exemplary use cases include detecting network intrusions [52], identifying fraudulent activities [3] and finding medical images that indicate the presence of a disease [164]. In addtion, methods on anomaly detection can be categorized as supervised [58], semi-supervised [4] or unsupervised [42].

(a) Sample images from MNIST [111].

(b) Sample images from USPS [82].

Figure 5.1 Sample images from the MNIST [111] and USPS [82] datasets.

5.1.1 Dataset

Experiments have been conducted on the MNIST [111] and USPS [82] datasets, and Figure 5.1 illustrates the sample images. In both datasets, greyscale images of handwritten digits are collected. On the one hand, the MNIST dataset contains 60,000 samples for training and 10,000 samples for testing, while the image resolution is 28x28. On the other hand, the USPS dataset includes a total of 9,298 16x16 images.

5.1.2 Evaluation Metric

Following the practice in [201, 205], clustering accuracy is defined as

$$\text{accuracy}(p,q) = \max_{m} \frac{1}{n} \sum_{i=1}^{n} \delta(p_i, m(q_i)), \tag{5.1}$$

where i is the index of current sample among n samples, p_i is the ground truth label, q_i is the predicted label, m refers to a mapping function between the predicted and ground truth labels, and δ denotes the Kronecker delta function.

Given the scores returned by an anomaly detection algorithm, the area under the Receiver Operating Characteristic (ROC) curve (*i.e.*, AUC) is reported. A list of true positive rate and false positive rate pairs is obtained at various thresholds [54].

5.2 L_2 Normalized Deep AutoEncoder Representations

5.2.1 Related Work

The study of clustering aims to group similar samples into the same cluster while samples from different clusters exhibit different patterns [89]. Autoencoder is a popular method in performing clustering with deep features that are extracted by neural networks [62, 63, 201]. Its main advantage is that a suitable feature representation is directly learned from data without requiring annotations for supervision. Existing studies focus on improving the architecture of neural networks and incorporating supplemental loss terms. At first, a dense autoencoder is trained by minimizing the reconstruction loss in [201], and the clusters are updated iteratively by matching the soft assignment to an auxiliary target distribution. Later on, two separate loss terms are utilized to optimize a dense autoencoder, while the clustering loss scatters the embedded points and the reconstruction loss preserves the local structure of data-generating distribution [62]. Finally, the drawback of using dense layers to process image data is addressed in [63], and better clustering accuracy is obtained by adopting convolutional layers instead.

Unsupervised anomaly detection refers to identifying abnormal samples without annotations [18]. Autoencoder has been applied in unsupervised anomaly detection [68, 218]. While a model is optimized by minimizing the reconstruction error during training, finding anomalies in the test dataset is feasible by inspecting the reconstruction error. A general assumption is that the model would not generalize well on abnormal samples since most training samples are normal. As a consequence, samples with higher reconstruction error are more likely to be abnormal. A fully convolutional autoencoder learns both the low-level motion features and the discriminative regular patterns in [68]. The resulting model generalizes well on multiple datasets, and a significant drop in the regularity scores is observed in the event of irregular motions. In addition, autoencoder is utilized to extract feature representations of a video from both spatial and temporal dimensions [218]. In order to enhance the motion features, a weight-decreasing prediction loss for generating future frames is used in training, along with the reconstruction loss.

5.2.2 Proposed Method

Methods based on AutoEncoder (AE) are capable of learning efficient codings in an unsupervised manner [119]. On the one hand, an encoder E converts the input I into a feature representation $E(I)$. Let f_i be the function applied at the i-th layer in a m-layer encoder, the extracted features can be written as

$$E(I) = f_m(f_{m-1}(\dots f_1(I))), \tag{5.2}$$

where f_i typically refers to a dense, convolutional, pooling or activation layer. On the other hand, a decoder D generates the reconstruction $D(E(I))$

$$D(E(I)) = g_n(g_{n-1}(\dots g_1(E(I)))), \tag{5.3}$$

where the decoder has n layers and g_j corresponds to the j-th layer. Additionally, the pooling layers in the encoder would be compensated by the upsampling layers in the decoder, so that the reconstruction has the same spatial size as the input.

Apart from defining the architecture of an autoencoder, it is necessary to specify the loss function l. Given a set of training samples as K, the resulting loss is

$$\text{loss} = \frac{1}{|K|} \sum_{k \in K} l(I_k, D(E(I_k))). \tag{5.4}$$

The mean squared error (MSE) loss is utilized to optimize the autoencoder, and it measures the difference between the input and the reconstruction. Alternatively, other loss functions are also applicable, such as the feature reconstruction loss [93] and the adversarial loss [56].

We suggest imposing a constraint on the feature representation $E(I)$. More specifically, a L_2 normalization layer is appended to the encoder so that the extracted features lie on a unit sphere. Note that the proposed method differs from L_2 regularization [135]. In general, L_2 regularization is applied to the model's parameters, and a separate loss term is added to the loss function during training. Adopting L_2 regularization drives the parameters toward zero and suppresses the overfitting issue [121]. By contrast, the L_2 normalization constraint is imposed on the extracted features rather than the parameters, and the length of a feature vector is fixed to 1.

With the aforementioned approach on L_2 normalized deep autoencoder representations, Equation 5.4 can be extended to

$$\begin{aligned} \text{loss}_c &= \frac{1}{|K|} \sum_{k \in K} l(I_k, D(E_c(I_k))), \\ E_c(I_k) &= \frac{E(I_k)}{||E(I_k)||_2}. \end{aligned} \tag{5.5}$$

Instead of only normalizing the extracted features during testing as a post-processing step, the autoencoder is directly optimized with the L_2 normalization constraint. After the training procedure is finalized, the k-means [120] algorithm is applied to perform clustering.

Furthermore, we introduce a clustering-based method for unsupervised anomaly detection. Given a dataset consisting of both normal and abnormal instances, an autoencoder is trained without annotations using Equation 5.5. Subsequently, the k-means [120] algorithm is employed to construct a set of clusters, and the cluster centroids are available. Under the antecedent that the percentage of normal instances is substantially higher than that of abnormal instances, the cluster centroids are representative of the normal instances, with some minor deviations due to the existence of abnormal instances. Given a test sample I_k, we quantify the normality score v_k as

$$v_k = \max_p (E_c(I_k) \cdot \frac{C_p}{||C_p||_2}), \tag{5.6}$$

where C_p is the cluster centroid with index p and $\cdot$ denotes the dot product operation. Even though the extracted features from the encoder are already L_2 normalized, it does not necessarily hold for the cluster centroids. For example, the centroid of points on a unit circle falls within the unit circle. As a result, we explicitly apply the L_2 normalization operation on C_p, and the normality score v_k is in the range of [-1, 1]. With the max operation, we iterate over all cluster centroids to calculate the normality score with respect to the closest cluster. A higher normality score indicates that the probability of being normal is greater. In the case that binary predictions are needed, a threshold $\tau \in [-1, 1]$ can be specified and instances with $v_k < \tau$ would be predicted as abnormal.

Table 5.1 Clustering accuracy with features extracted by dense autoencoders. †: ours. Reproduced with permission from [6].

	DAE [62] k-means	DEC [201]	IDEC [62]	DAE † k-means	DAE L_2 † k-means
MNIST [111]	81.82	86.55	88.06	81.43	**90.20**

5.2.3 Experimental Results

5.2.3.1 Clustering

Table 5.1 shows the clustering accuracy using dense autoencoders (DAE). Denoted by DAE k-means, we employ the baseline used in DEC [201] and IDEC [62]. The encoder has 4 dense layers with 500, 500, 2000 and 10 units, respectively. The decoder has another 4 dense layers with 2000, 500, 500 and d units, where d refers to the input dimensionality. In addition, the LeakyReLU activation function is applied with each dense layer, except for the last one. Three conclusions can be drawn from the results. Firstly, our re-implementation of the baseline is comparable to the reference baseline in IDEC. Secondly, the separate clustering loss terms introduced in DEC and IDEC bring substantial improvements. Thirdly, our proposed method achieves even better performance without the need to incorporate a clustering loss term.

Table 5.2 shows the clustering accuracy using convolutional autoencoders (CAE). Refereed as CAE k-means, the baseline from DCEC [63] is adopted in our study. The encoder consists of consecutive 2-D convolutional layers with 32, 64 and 128 filters, and each layer uses 2 x 2 strides with ReLU. Subsequently, a flatten layer is appended, and a dense layer reduces the dimensionality to 10. To provide 4-D feature maps to the decoder, another dense layer is applied with a reshaping operation. Afterward, upsampling layers and 2-D convolutional layers are used in the decoder. The numbers of filters in 2-D convolutional layers are 64, 32 and 1, respectively. Following the results of using dense autoencoders, similar patterns can be observed with convolutional autoencoders. Firstly, only minor differences can be found between the reference baseline and our re-implementation of the baseline. Secondly, DCEC reaches better performances than the baseline on account of the separate clustering loss term. Thirdly, our proposed method significantly increases the clustering accuracy, and it is superior to DCEC.

Table 5.2 Clustering accuracy with features extracted by convolutional autoencoders. †: ours. Reproduced with permission from [6].

	CAE [63] k-means	DCEC [63]	CAE † k-means	CAE L_2 † k-means
MNIST [111]	84.90	88.97	84.83	**95.11**
USPS [82]	74.15	79.00	73.52	**91.35**

Table 5.3 Comparison among normalization techniques. Reproduced with permission from [6].

	CAE k-means	CAE Batch [85] k-means	CAE Layer [8] k-means	CAE L_2 k-means
MNIST [111]	84.83	70.67	70.83	**95.11**
USPS [82]	73.52	74.95	75.26	**91.35**

Furthermore, comparison among normalization techniques is presented in Table 5.3. In particular, convolutional autoencoders are trained with batch normalization [85], layer normalization [8] or L_2 normalization. Using batch or layer normalization makes slight improvements over the baseline on the USPS [82] dataset, while a shape drop in clustering accuracy occurs on the MNIST [111] dataset. On the contrary, adopting L_2 normalization increases the performance consistently.

5.2.3.2 Unsupervised Anomaly Detection

Given a classification dataset with n classes, a specific class is randomly selected, and it is treated as abnormal. Instances in the abnormal class are split into multiple (*e.g.*, 10) partitions, and only one partition is kept while the other partitions are discarded. Under this setting, the number of abnormal instances would be significantly lower than normal instances, and it complies with the definition of anomaly. Afterward, an autoencoder is trained on the updated dataset, with or without the L_2 normalization constraint. With the feature representations extracted by the encoder, the k-means [120] algorithm is utilized, and a set of cluster centroids is obtained. It is assumed that the number of normal classes is known, *i.e.*, k is set to n-1. Subsequently, Equation 5.6 is applied to calculate the normality score of each test sample, and the area under the ROC curve is calculated. Due to the randomness of the aforementioned procedure, we iterate over each partition and use the average of AUC scores.

Table 5.4 Results of unsupervised anomaly detection on the MNIST [111] dataset. Reproduced with permission from [6].

Digit	CAE recons	CAE cluster	CAE L_2 cluster	Digit	AE [5] recons	VAE [5] recons	CAE L_2 cluster
0	0.7025	0.7998	**0.9615**	0	0.825	0.917	**0.9615**
1	0.0782	0.8871	**0.9673**	1	0.135	0.136	**0.9673**
2	0.8790	0.7512	**0.9790**	2	0.874	0.921	**0.9790**
3	0.8324	0.8449	**0.9382**	3	0.761	0.781	**0.9382**
4	0.7149	0.4988	**0.7825**	4	0.727	**0.808**	0.7825
5	0.8359	0.7635	**0.9136**	5	0.792	0.862	**0.9136**
6	0.6925	0.7896	**0.9497**	6	0.812	0.848	**0.9497**
7	0.5767	0.7421	**0.9100**	7	0.508	0.596	**0.9100**
8	0.8912	0.9200	**0.9237**	8	0.869	0.895	**0.9237**
9	0.5140	**0.8944**	0.7495	9	0.548	0.545	**0.7495**

Table 5.4 shows a detailed comparison among the results of unsupervised anomaly detection on the MNIST [111] dataset. Since there are 10 digits in MNIST, separate experiments have been conducted by treating each digit as abnormal.

In the left part of Table 5.4, the convolutional autoencoders from DCEC [63] are used. In addition to calculating the normality score based on clustering, another option is validated, *i.e.*, measuring the reconstruction error as in [5, 198]. It is evident that using the clustering-based method introduces noticeable improvements over calculating the reconstruction error, especially for digits 1, 7, and 9. For example, the digit 1 is relatively simple to reconstruct, and a model would still generate reasonable reconstructions of corresponding samples. In addition, imposing the L_2 normalization constraint is beneficial in 9 out of 10 cases, except for digit 9.

In the right part of Table 5.4, we provide comparisons with two methods in [5]. Both methods are based on dense autoencoders, and the reconstruction error is utilized. On the one hand, variational autoencoders perform better than vanilla autoencoders. On the other hand, our proposed method with convolutional autoencoders and the L_2 normalization constraint obtains the best performance in 9 out of 10 cases. Note that abnormal instances are not included in the training procedure in [5], *i.e.*, the model is optimized using only the normal instances. It introduces inexplicit supervision that the training samples are normal, while our scenario is entirely unsupervised.

5.2.4 Summary

We studied the effects of imposing the L_2 normalization constraint on feature representations when training an autoencoder. More specifically, the extracted features lie on a unit sphere, and the length of a feature vector is fixed to 1. Applying the k-means [120] algorithm on such features yields superior clustering accuracy without the need to incorporate a clustering loss term. In addition, we extend the clustering method by quantifying the normality score of a test sample. The resulting unsupervised anomaly detection method introduces significant improvements over the methods that are based on reconstruction error.

6 CONCLUSIONS

In this dissertation, we study image embeddings extracted by neural networks. The main challenge is to develop a feature extraction scheme that is suitable for solving a particular task. More specifically, three research questions have been rigorously investigated in the context of person re-identification, vehicle attribute recognition, neural image compression, clustering and unsupervised anomaly detection.

The first research question is how to extract distinctive features so that features of samples from distinct categories would differ significantly. It is a core aspect for image retrieval systems, in which searching is performed over image features based on similarities. In total, we have summarized four publications that are dedicated to addressing this problem.

First of all, Section 2.2 introduces an adaptive L_2 regularization mechanism [135]. In existing studies, the conventional L_2 regularization method is widely adopted to suppress the overfitting issue. Ideally, each regularization factor should be hand-picked via hyperparameter optimization. However, modern neural networks have hundreds of distinct parameters, making the search space excessively large. To address this issue, we suggest learning the regularization factors through backpropagation. A trainable scalar variable is defined for each parameter, and it is further processed by a scaled hard sigmoid function. The resulting tensor represents the regularization strength.

Later on, Section 2.3 analyzes the gap between training and inference [138]. Adopting data augmentation during training is beneficial because it would suppress overfitting and improve generalization. Also known as test-time augmentation, feature vectors are obtained for the original and flipped images, and the mean feature vector is utilized in inference. However, the mean feature vector is absent when optimizing the model, thus the training procedure differs from the inference procedure. To solve this issue, we explore different combinations of model architectures. Using the FlipReID architecture with the flipping loss gives the best performance.

In addition, Section 3.2 presents a study on recognizing vehicle attributes, including type, make, and model [136]. Firstly, a detailed comparison is made among relevant datasets. Considering the size and diversity of a dataset, CompCars [203], MIO-TCD [125] and VERI-Wild [122] are the most suitable options for future works. Secondly, a comprehensive survey has been carried out on existing methods, including hand-crafted and deep learning approaches. As expected, deep learning approaches are more competent in processing challenging samples. Thirdly, we address the problem of vehicle attribute recognition from the perspective of image retrieval, and results of different loss functions are reported. Compared with the image classification approach, retraining the model for unseen classes is no longer necessary.

Finally, Section 5.2 introduces the method of applying L_2 normalization on feature representations when optimizing an autoencoder in an unsupervised manner [6]. As a result, the extracted features lie on a unit sphere, and the length of a feature vector is fixed to 1. Experiments have been conducted on two tasks, *i.e.*, clustering and unsupervised anomaly detection. For the task of clustering, features learned are fed to the k-means [120] algorithm. For the task of unsupervised anomaly detection, normality scores are defined by comparing test samples with cluster centroids. Results in both tasks indicate that imposing the L_2 normalization constraint improves the performance significantly.

The second research question is how to make features privacy-preserving so that an adversary can not infer sensitive information. While significant progress has been made in productizing Machine Learning (ML) algorithms, the trustworthiness of those techniques remains an open question. Section 2.4 examines model inversion attacks against ML models under a black-box setting [137]. Two assumptions are made, *i.e.*, feature vectors of user data are accessible, and a black-box API is available for conducting inference. Based on feature vectors extracted by state-of-the-art models in person re-identification, it is achievable to recognize auxiliary attributes with admissible accuracy and obtain a reasonable reconstruction of user data. As a countermeasure, it is essential to utilize an encryption algorithm when transferring and storing deep features. Furthermore, we propose a method termed ShuffleBits, in which shuffling is performed on the binary sequence of each floating-point number. It can be seamlessly incorporated into any existing neural network, and the resulting model would directly yield encrypted feature vectors.

The third research question is how to obtain compressible features so that the storage requirements are lowered. It contributes to achieving data compression on deep features. Section 4.2 analyzes an autoencoder-based algorithm to compress images block by block [7]. While the encoder extracts a feature vector of an image, the decoder aims to reconstruct the original images. On the one hand, feature representations should be compressible. On the other hand, the differences between the original and reconstructed images should be minimized. Several tricks have been incorporated, including handling the binarization process, exploiting L_2 normalization, incorporating the entropy-friendly loss, training multiple compression networks, employing a deblocking network, and optimizing the encoded features. At similar bits per pixel, the proposed method obtains higher PSNR score than JPEG [187].

Studies included in this dissertation shine a light on possible solutions for the aforementioned three research questions. Deriving distinctive, privacy-preserving and compressible feature representations is still a major challenge. More research is required to sharpen the understanding of Machine Learning.

REFERENCES

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard et al. TensorFlow: A system for large-scale machine learning. *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*. 2016, 265–283.

[2] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte and L. V. Gool. Generative adversarial networks for extreme learned image compression. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, 221–231.

[3] M. Ahmed, A. N. Mahmood and M. R. Islam. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems* 55 (2016), 278–288.

[4] S. Akcay, A. Atapour-Abarghouei and T. P. Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. *Asian conference on computer vision*. Springer. 2018, 622–637.

[5] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE* 2.1 (2015), 1–18.

[6] C. Aytekin, X. Ni, F. Cricri and E. Aksu. Clustering and Unsupervised Anomaly Detection with L2 Normalized Deep Auto-Encoder Representations. *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, 1–6. DOI: 10.1109/IJCNN.2018.8489068.

[7] C. Aytekin, X. Ni, F. Cricri, J. Lainema, E. Aksu and M. Hannuksela. Block-optimized Variable Bit Rate Neural Image Compression. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018, 2551–2554.

[8] J. L. Ba, J. R. Kiros and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[9] S. Bai, Z. Liu and C. Yao. Classify vehicles in traffic scene images with deformable part-based models. *Machine Vision and Applications* 29.3 (2018), 393–403.

[10] J. Ballé, V. Laparra and E. P. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. *2016 Picture Coding Symposium (PCS)*. IEEE. 2016, 1–5.

[11] J. Ballé, V. Laparra and E. P. Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704* (2016).

[12] A. Bedagkar-Gala and S. K. Shah. A survey of approaches and trends in person re-identification. *Image and vision computing* 32.4 (2014), 270–286.

[13] F. Bellard. *BPG Image Format*. 2014. URL: `https://bellard.org/bpg/`.

[14] Y. Bengio, A. Courville and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), 1798–1828.

[15] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan and L. Jiao. Greedy discrete particle swarm optimization for large-scale social network clustering. *Information Sciences* 316 (2015), 503–516.

[16] R. Caruana. Multitask learning. *Machine learning* 28.1 (1997), 41–75.

[17] L. Cavigelli, P. Hager and L. Benini. CAS-CNN: A deep convolutional neural network for image compression artifact suppression. *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, 752–759.

[18] R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[19] *Challenge on Learned Image Compression*. 2018. URL: `http://clic.compression.cc/2018/challenge/`.

[20] V. Chandola, A. Banerjee and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41.3 (2009), 1–58.

[21] J. Chang, L. Wang, G. Meng, S. Xiang and C. Pan. Deep adaptive image clustering. *Proceedings of the IEEE international conference on computer vision*. 2017, 5879–5887.

[22] B. Chen, W. Deng and J. Hu. Mixed high-order attention network for person re-identification. *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 371–381.

[23] X. Chen, C. Fu, Y. Zhao, F. Zheng, J. Song, R. Ji and Y. Yang. Salience-Guided Cascaded Suppression Network for Person Re-Identification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 3300–3310.

[24] X. Clady, P. Negri, M. Milgram and R. Poulenard. Multi-class vehicle type recognition system. *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*. Springer. 2008, 228–239.

[25] G. Claeskens, N. L. Hjort et al. Model selection and model averaging. *Cambridge Books* (2008).

[26] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning* 20.3 (1995), 273–297.

[27] J. L. Crowley and A. C. Parker. A representation for shape based on peaks and ridges in the difference of low-pass transform. *IEEE transactions on pattern analysis and machine intelligence* 2 (1984), 156–170.

[28] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan and Q. V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501* (2018).

[29] J. Daemen and V. Rijmen. AES proposal: Rijndael. (1999).

[30] Z. Dai, M. Chen, X. Gu, S. Zhu and P. Tan. Batch DropBlock network for person re-identification and beyond. *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 3691–3701.

[31] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, 886–893.

[32] S. Deerwester, S. Dumais, T. Landauer, G. Furnas and L. Beck. Improving information-retrieval with latent semantic indexing. *Proceedings of the ASIS annual meeting*. Vol. 25. INFORMATION TODAY INC 143 OLD MARLTON PIKE, MEDFORD, NJ 08055-8750. 1988, 36–40.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee. 2009, 248–255.

[34] J. Deng, J. Krause and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, 580–587.

[35] J. Deng, J. Guo, N. Xue and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 4690–4699.

[36] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[37] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552* (2017).

[38] S. Ding, H. Zhu, W. Jia and C. Su. A survey on feature extraction for pattern recognition. *Artificial Intelligence Review* 37.3 (2012), 169–180.

[39] C. Dong, Y. Deng, C. C. Loy and X. Tang. Compression artifacts reduction by a deep convolutional network. *Proceedings of the IEEE International Conference on Computer Vision*. 2015, 576–584.

[40] Z. Dong and Y. Jia. Vehicle type classification using distributions of structural and appearance-based features. *2013 IEEE International Conference on Image Processing*. IEEE. 2013, 4321–4324.

[41] Z. Dong, Y. Wu, M. Pei and Y. Jia. Vehicle type classification using a semisupervised convolutional neural network. *IEEE transactions on intelligent transportation systems* 16.4 (2015), 2247–2256.

[42] E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo. A geometric framework for unsupervised anomaly detection. *Applications of data mining in computer security*. Springer, 2002, 77–101.

[43] P. Fang, J. Zhou, S. K. Roy, L. Petersson and M. Harandi. Bilinear attention networks for person retrieval. *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 8030–8039.

[44] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence* 32.9 (2009), 1627–1645.

[45] W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*. Interlaken. 1987, 281–305.

[46] M. Fredrikson, S. Jha and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, 1322–1333.

[47] M. Fredrikson, E. Lantz, S. Jha, S. Lin, D. Page and T. Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, 17–32.

[48] J. Fu, H. Zheng and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 4438–4446.

[49] Z. Fu, W. Hu and T. Tan. Similarity based vehicle trajectory clustering and anomaly detection. *IEEE International Conference on Image Processing 2005*. Vol. 2. Ieee. 2005, II–602.

[50] L. Galteri, L. Seidenari, M. Bertini and A. Del Bimbo. Deep generative adversarial compression artifact removal. *Proceedings of the IEEE International Conference on Computer Vision*. 2017, 4826–4835.

[51] Y. Gao, O. Beijbom, N. Zhang and T. Darrell. Compact bilinear pooling. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 317–326.

[52] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández and E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security* 28.1-2 (2009), 18–28.

[53] J. J. Gibson. The perception of the visual world. (1950).

[54] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one* 11.4 (2016), e0152173.

[55] I. Goodfellow, Y. Bengio and A. Courville. *Deep learning*. MIT press, 2016.

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*. 2014, 2672–2680.

[57] I. J. Goodfellow, J. Shlens and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[58] N. Görnitz, M. Kloft, K. Rieck and U. Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research* 46 (2013), 235–262.

[59] P. Grosche, M. Müller and F. Kurth. Cyclic tempogram—A mid-level tempo representation for musicsignals. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2010, 5522–5525.

[60] J. Guo and H. Chao. Building dual-domain representations for compression artifacts reduction. *European Conference on Computer Vision*. Springer. 2016, 628–644.

[61] J. Guo and H. Chao. One-to-many network for visually pleasing compression artifacts reduction. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 3038–3047.

[62] X. Guo, L. Gao, X. Liu and J. Yin. Improved Deep Embedded Clustering with Local Structure Preservation. *Ijcai*. 2017, 1753–1759.

[63] X. Guo, X. Liu, E. Zhu and J. Yin. Deep clustering with convolutional autoencoders. *International conference on neural information processing*. Springer. 2017, 373–382.

[64] J. Han and C. Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. *International workshop on artificial neural networks*. Springer. 1995, 195–201.

[65] R. M. Haralick. Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing* 22.1 (1983), 28–38.

[66] C. G. Harris and M. Stephens. A combined corner and edge detector. *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, 10–5244.

[67] C. Harte, M. Sandler and M. Gasser. Detecting harmonic change in musical audio. *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*. 2006, 21–26.

[68] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury and L. S. Davis. Learning temporal regularity in video sequences. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, 733–742.

[69] K. He, X. Zhang, S. Ren and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *Proceedings of the IEEE International Conference on Computer Vision*. 2015, 1026–1034.

[70] K. He, X. Zhang, S. Ren and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence* 37.9 (2015), 1904–1916.

[71] K. He, X. Zhang, S. Ren and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 770–778.

[72] L. He, X. Liao, W. Liu, X. Liu, P. Cheng and T. Mei. Fastreid: A pytorch toolbox for general instance re-identification. *arXiv preprint arXiv:2006.02631* 6.7 (2020), 8.

[73] L. He and W. Liu. Guided Saliency Feature Learning for Person Re-identification in Crowded Scenes. *European Conference on Computer Vision*. Springer. 2020, 357–373.

[74] A. Hermans, L. Beyer and B. Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737* (2017).

[75] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold et al. CNN architectures for large-scale audio classification. *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2017, 131–135.

[76] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science* 313.5786 (2006), 504–507.

[77] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[78] R. Hou, B. Ma, H. Chang, X. Gu, S. Shan and X. Chen. Interaction-and-aggregation network for person re-identification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 9317–9326.

[79] Q. Hu, H. Wang, T. Li and C. Shen. Deep CNNs with spatially weighted pooling for fine-grained car recognition. *IEEE Transactions on Intelligent Transportation Systems* 18.11 (2017), 3147–3156.

[80] S. Huang, N. Papernot, I. Goodfellow, Y. Duan and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* (2017).

[81] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE* 40.9 (1952), 1098–1101.

[82] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence* 16.5 (1994), 550–554.

[83] H. Huttunen, F. S. Yancheshmeh and K. Chen. Car type recognition with deep neural networks. *2016 IEEE intelligent vehicles symposium (IV)*. IEEE. 2016, 1115–1120.

[84] IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)* (2019), 1–84. DOI: 10.1109/IEEESTD.2019.8766229.

[85] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[86] P. Isola, J.-Y. Zhu, T. Zhou and A. A. Efros. Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, 1125–1134.

[87] R. A. Jacobs, M. I. Jordan, S. J. Nowlan and G. E. Hinton. Adaptive mixtures of local experts. *Neural computation* 3.1 (1991), 79–87.

[88] M. Jaderberg, K. Simonyan, A. Zisserman and K. Kavukcuoglu. Spatial transformer networks. *Advances in neural information processing systems*. 2015, 2017–2025.

[89] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[90] S. Ji, W. Xu, M. Yang and K. Yu. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012), 221–231.

[91] D.-N. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao and L.-H. Cai. Music type classification by spectral contrast feature. *Proceedings. IEEE International Conference on Multimedia and Expo*. Vol. 1. IEEE. 2002, 113–116.

[92] X. Jin, C. Lan, W. Zeng, Z. Chen and L. Zhang. Style normalization and restitution for generalizable person re-identification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 3143–3152.

[93] J. Johnson, A. Alahi and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *European conference on computer vision*. Springer. 2016, 694–711.

[94] H. Jung, M.-K. Choi, J. Jung, J.-H. Lee, S. Kwon and W. Young Jung. ResNet-based vehicle classification and localization in traffic surveillance systems. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, 61–67.

[95] M. Kafai and B. Bhanu. Dynamic Bayesian networks for vehicle classification in video. *IEEE Transactions on Industrial Informatics* 8.1 (2011), 100–109.

[96] P. Kanerva, J. Kristoferson and A. Holst. Random indexing of text samples for latent semantic analysis. *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 22. 22. 2000.

[97] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei. Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014, 1725–1732.

[98] P.-K. Kim and K.-T. Lim. Vehicle type classification using bagging and convolutional neural network on multi view surveillance image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, 41–46.

[99] A. Klapuri and M. Davy. Signal processing methods for music transcription. (2007).

[100] D. E. Knuth. *The art of computer programming*. Vol. 3. Pearson Education, 1997.

[101] J. R. Koza, F. H. Bennett, D. Andre and M. A. Keane. Automated design of both the topology and sizing of analog electrical circuits using genetic programming. *Artificial Intelligence in Design'96*. Springer, 1996, 151–170.

[102] J. Krause, H. Jin, J. Yang and L. Fei-Fei. Fine-grained recognition without part annotations. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, 5546–5555.

[103] J. Krause, M. Stark, J. Deng and L. Fei-Fei. 3d object representations for fine-grained categorization. *Proceedings of the IEEE international conference on computer vision workshops*. 2013, 554–561.

[104] A. Krizhevsky, I. Sutskever and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*. 2012, 1097–1105.

[105] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio and T. Serre. HMDB: a large video database for human motion recognition. *2011 International conference on computer vision*. IEEE. 2011, 2556–2563.

[106] A. Kurakin, I. Goodfellow, S. Bengio et al. *Adversarial examples in the physical world*. 2016.

[107] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig et al. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982* (2018).

[108] P. Laskov et al. Practical evasion of a learning-based classifier: A case study. *2014 IEEE symposium on security and privacy*. IEEE. 2014, 197–211.

[109] J. A. Lasserre, C. M. Bishop and T. P. Minka. Principled hybrids of generative and discriminative models. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 1. IEEE. 2006, 87–94.

[110] Y. LeCun, Y. Bengio and G. Hinton. Deep learning. *nature* 521.7553 (2015), 436–444.

[111] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86.11 (1998), 2278–2324.

[112] H. Lee, P. Pham, Y. Largman and A. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information processing systems* 22 (2009), 1096–1104.

[113] L. Liao, R. Hu, J. Xiao, Q. Wang, J. Xiao and J. Chen. Exploiting effects of parts in fine-grained categorization of vehicles. *IEEE International Conference on Image Processing*. IEEE. 2015, 745–749.

[114] T.-Y. Lin, A. RoyChowdhury and S. Maji. Bilinear cnn models for fine-grained visual recognition. *Proceedings of the IEEE international conference on computer vision*. 2015, 1449–1457.

[115] Y.-L. Lin, V. I. Morariu, W. Hsu and L. S. Davis. Jointly optimizing 3D model fitting and fine-grained classification. *European Conference on Computer Vision*. Springer. 2014, 466–480.

[116] Y. Lin, L. Zheng, Z. Zheng, Y. Wu, Z. Hu, C. Yan and Y. Yang. Improving person re-identification by attribute and identity learning. *Pattern Recognition* (2019).

[117] T. Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision* 30.2 (1998), 79–116.

[118] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao and J. Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265* (2019).

[119] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu and F. E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing* 234 (2017), 11–26.

[120] S. Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory* 28.2 (1982), 129–137.

[121] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2018).

[122] Y. Lou, Y. Bai, J. Liu, S. Wang and L. Duan. VERI-Wild: A large dataset and a new method for vehicle re-identification in the wild. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 3235–3243.

[123] D. G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the seventh IEEE international conference on computer vision*. Vol. 2. Ieee. 1999, 1150–1157.

[124] H. Luo, Y. Gu, X. Liao, S. Lai and W. Jiang. Bag of tricks and a strong baseline for deep person re-identification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[125] Z. Luo, F. Branchaud-Charron, C. Lemaire, J. Konrad, S. Li, A. Mishra, A. Achkar, J. Eichel and P.-M. Jodoin. MIO-TCD: A new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing* 27.10 (2018), 5129–5141.

[126] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng and C. Potts. Learning word vectors for sentiment analysis. *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 2011, 142–150.

[127] X. Mao, C. Shen and Y.-B. Yang. Image restoration using very deep convolutional encoder-decoder networks with symmetric skip connections. *Advances in neural information processing systems* 29 (2016), 2802–2810.

[128] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang and S. Paul Smolley. Least squares generative adversarial networks. *Proceedings of the IEEE international conference on computer vision*. 2017, 2794–2802.

[129] N. Martinel, G. L. Foresti and C. Micheloni. Deep pyramidal pooling with attention for person re-identification. *IEEE Transactions on Image Processing* 29 (2020), 7306–7316.

[130] S. Merity, C. Xiong, J. Bradbury and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).

[131] M. Mohri, A. Rostamizadeh and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[132] L. Mosley. A balanced approach to the multi-class imbalance problem. (2013).

[133] A. Nagrani, J. S. Chung and A. Zisserman. Voxceleb: a large-scale speaker identification dataset. *arXiv preprint arXiv:1706.08612* (2017).

[134] A. Ng. Machine Learning and AI via Brain simulations. *Accessed: May* 3 (2013), 2018.

[135] X. Ni, L. Fang and H. Huttunen. Adaptive L2 Regularization in Person Re-Identification. *2020 25th International Conference on Pattern Recognition (ICPR)*. 2021, 9601–9607. DOI: 10.1109/ICPR48806.2021.9412481.

[136] X. Ni and H. Huttunen. Vehicle Attribute Recognition by Appearance: Computer Vision Methods for Vehicle Type, Make and Model Classification. *Journal of Signal Processing Systems* 93.4 (2021), 357–368. DOI: 10.1007/s11265-020-01567-6.

[137] X. Ni, H. Huttunen and E. Rahtu. On the Importance of Encrypting Deep Features. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2021, 4125–4132. DOI: 10.1109/ICCVW54120.2021.00460.

[138] X. Ni and E. Rahtu. FlipReID: Closing the Gap Between Training and Inference in Person Re-Identification. *2021 9th European Workshop on Visual Information Processing (EUVIP)*. 2021, 1–6. DOI: 10.1109/EUVIP50544.2021.9484010.

[139] H. Oh Song, Y. Xiang, S. Jegelka and S. Savarese. Deep metric learning via lifted structured feature embedding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 4004–4012.

[140] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499* (2016).

[141] X. Pan, P. Luo, J. Shi and X. Tang. Two at once: Enhancing learning and generalization capacities via ibn-net. *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, 464–479.

[142] V. Panayotov, G. Chen, D. Povey and S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2015, 5206–5210.

[143] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister and S. Vitaladevuni. Multi-task learning and weighted cross-entropy for DNN-based keyword spotting. *Interspeech*. Vol. 9. 2016, 760–764.

[144] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[145] Y. Peng, J. S. Jin, S. Luo, M. Xu, S. Au, Z. Zhang and Y. Cui. Vehicle type classification using data mining techniques. *The Era of Interactive Media*. Springer, 2013, 325–335.

[146] Y. Peng, J. S. Jin, S. Luo, M. Xu and Y. Cui. Vehicle type classification using PCA with self-clustering. *IEEE International Conference on Multimedia and Expo Workshops*. IEEE. 2012, 384–389.

[147] F. Perronnin, J. Sánchez and T. Mensink. Improving the fisher kernel for large-scale image classification. *European conference on computer vision*. Springer. 2010, 143–156.

[148] V. S. Petrovic and T. F. Cootes. Analysis of Features for Rigid Structure Vehicle Type Recognition. *BMVC*. Vol. 2. Kingston University, London. 2004, 587–596.

[149] L. Prechelt. Early stopping-but when?: *Neural Networks: Tricks of the trade*. Springer, 1998, 55–69.

[150] J. M. S. Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics* 10.1 (1970), 15–19.

[151] A. Psyllos, C.-N. Anagnostopoulos and E. Kayafas. Vehicle model recognition from frontal view image measurements. *Computer Standards & Interfaces* 33.2 (2011), 142–151.

[152] R. F. Rachmadi, K. Uchimura, G. Koutaki and K. Ogata. Single image vehicle classification using pseudo long short-term memory classifier. *Journal of Visual Communication and Image Representation* 56 (2018), 265–274.

[153] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever. Improving language understanding by generative pre-training. (2018).

[154] E. Rendón, I. Abundez, A. Arizmendi and E. M. Quiroz. Internal versus external cluster validation indexes. *International Journal of computers and communications* 5.1 (2011), 27–34.

[155] E. Ristani, F. Solera, R. Zou, R. Cucchiara and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. *European Conference on Computer Vision*. Springer. 2016, 17–35.

[156] L. G. Roberts. Machine perception of three-dimensional solids. PhD thesis. Massachusetts Institute of Technology, 1963.

[157] O. Ronneberger, P. Fischer and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, 234–241.

[158] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science* 290.5500 (2000), 2323–2326.

[159] G. Salton, A. Wong and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM* 18.11 (1975), 613–620.

[160] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development* 3.3 (1959), 210–229.

[161] U. Sara, M. Akter and M. S. Uddin. Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study. *Journal of Computer and Communications* 7.3 (2019), 8–18.

[162] B. M. Sarwar, G. Karypis, J. Konstan and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. *Proceedings of the fifth international conference on computer and information technology*. Vol. 1. Citeseer. 2002, 291–324.

[163] K. Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.

[164] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. *International conference on information processing in medical imaging*. Springer. 2017, 146–157.

[165] F. Schroff, D. Kalenichenko and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, 815–823.

[166] R. Shokri, M. Stronati, C. Song and V. Shmatikov. Membership inference attacks against machine learning models. *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, 3–18.

[167] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data* 6.1 (2019), 1–48.

[168] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al. Mastering the game of Go with deep neural networks and tree search. *nature* 529.7587 (2016), 484.

[169] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. *Proceedings of the IEEE International Conference on Computer Vision*. 2015, 1143–1151.

[170] J. Sochor, A. Herout and J. Havel. Boxcars: 3D boxes as cnn input for improved fine-grained vehicle recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 3006–3015.

[171] J. Sochor, J. Spanhel and A. Herout. Boxcars: Improving fine-grained recognition of vehicles using 3D bounding boxes in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems* 20.1 (2018), 97–108.

[172] K. Soomro, A. R. Zamir and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402* (2012).

[173] M. Stark, J. Krause, B. Pepik, D. Meger, J. J. Little, B. Schiele and D. Koller. Fine-grained categorization for 3D scene understanding. *International Journal of Robotics Research* 30.13 (2011), 1543–1552.

[174] G. Sullivan and J. Ohm. Meeting report of the 13th meeting of the joint collaborative team on video coding (jct-vc). *ITU-T/ISO/IEC Joint Collaborative Team on Video Coding*. 2013.

[175] W. Sun, X. Zhang, S. Shi, J. He and Y. Jin. Vehicle type recognition combining global and local features via two-stage classification. *Mathematical Problems in Engineering* 2017 (2017).

[176] Y. Sun, L. Zheng, Y. Yang, Q. Tian and S. Wang. Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline). *Proceedings of the European Conference on Computer Vision*. 2018, 480–496.

[177] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens and Z. Wojna. Rethinking the inception architecture for computer vision. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, 2818–2826.

[178] J. Taek Lee and Y. Chung. Deep learning-based vehicle classification using an ensemble of local expert and global networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, 47–52.

[179] C.-P. Tay, S. Roy and K.-H. Yap. Aanet: Attribute attention network for person re-identifications. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 7134–7143.

[180] R. Theagarajan, F. Pala and B. Bhanu. EDeN: Ensemble of deep networks for vehicle classification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, 33–40.

[181] L. Theis, W. Shi, A. Cunningham and F. Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395* (2017).

[182] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell and R. Sukthankar. Variable rate image compression with recurrent neural networks. *arXiv preprint arXiv:1511.06085* (2015).

[183] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor and M. Covell. Full resolution image compression with recurrent neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 5306–5314.

[184] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter and T. Ristenpart. Stealing Machine Learning Models via Prediction APIs. *Proceedings of the 25th USENIX Conference on Security Symposium*. 2016, 601–618.

[185] S. Truex, L. Liu, M. E. Gursoy, L. Yu and W. Wei. Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing* (2019).

[186] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*. 2017, 5998–6008.

[187] G. K. Wallace. The JPEG still picture compression standard. *IEEE transactions on consumer electronics* 38.1 (1992), xviii–xxxiv.

[188] G. Wang, S. Yang, H. Liu, Z. Wang, Y. Yang, S. Wang, G. Yu, E. Zhou and J. Sun. High-Order Information Matters: Learning Relation and Topology for Occluded Person Re-Identification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 6449–6458.

[189] G. Wang, Y. Yuan, X. Chen, J. Li and X. Zhou. Learning discriminative features with multiple granularities for person re-identification. *Proceedings of the 26th ACM international conference on Multimedia*. 2018, 274–282.

[190] H. Wang and C. Schmid. Action recognition with improved trajectories. *Proceedings of the IEEE international conference on computer vision*. 2013, 3551–3558.

[191] M. Wang and W. Deng. Deep face recognition: A survey. *arXiv preprint arXiv:1804.06655* (2018).

[192] X. Wang, G. Doretto, T. Sebastian, J. Rittscher and P. Tu. Shape and appearance context modeling. *2007 ieee 11th international conference on computer vision*. Ieee. 2007, 1–8.

[193] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio et al. Tacotron: Towards end-to-end speech synthesis. *arXiv preprint arXiv:1703.10135* (2017).

[194] L. Wei, S. Zhang, W. Gao and Q. Tian. Person transfer gan to bridge domain gap for person re-identification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, 79–88.

[195] I. H. Witten, R. M. Neal and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM* 30.6 (1987), 520–540.

[196] R.-S. Wu and P.-H. Chou. Customer segmentation of multiple category data in e-commerce using a soft-clustering approach. *Electronic Commerce Research and Applications* 10.3 (2011), 331–341.

[197] X. Wu, M. Fredrikson, S. Jha and J. F. Naughton. A methodology for formalizing model-inversion attacks. *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE. 2016, 355–370.

[198] Y. Xia, X. Cao, F. Wen, G. Hua and J. Sun. Learning discriminative reconstructions for unsupervised outlier removal. *Proceedings of the IEEE International Conference on Computer Vision*. 2015, 1511–1519.

[199] Y. Xiang, Y. Fu and H. Huang. Global topology constraint network for fine-grained vehicle recognition. *IEEE Transactions on Intelligent Transportation Systems* (2019).

[200] C. Xie, J. Wang, Z. Zhang, Y. Zhou, L. Xie and A. Yuille. Adversarial examples for semantic segmentation and object detection. *Proceedings of the IEEE International Conference on Computer Vision*. 2017, 1369–1378.

[201] J. Xie, R. Girshick and A. Farhadi. Unsupervised deep embedding for clustering analysis. *International conference on machine learning*. PMLR. 2016, 478–487.

[202] H. Yakura and J. Sakuma. Robust audio adversarial example for a physical attack. *arXiv preprint arXiv:1810.11793* (2018).

[203] L. Yang, P. Luo, C. Change Loy and X. Tang. A large-scale car dataset for fine-grained categorization and verification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, 3973–3981.

[204] W. Yang, H. Huang, Z. Zhang, X. Chen, K. Huang and S. Zhang. Towards rich feature discovery with class activation maps augmentation for person re-identification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 1389–1398.

[205] Y. Yang, D. Xu, F. Nie, S. Yan and Y. Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing* 19.10 (2010), 2761–2773.

[206] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao and S. C. H. Hoi. Deep learning for person re-identification: A survey and outlook. *arXiv preprint arXiv:2001.04193* (2020).

[207] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao and S. C. H. Hoi. Deep learning for person re-identification: A survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[208] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga and G. Toderici. Beyond short snippets: Deep networks for video classification. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, 4694–4702.

[209] R. Zeng, Z. Ge, S. Denman, S. Sridharan and C. Fookes. Geometry-constrained car recognition using a 3D perspective network. *arXiv preprint arXiv:1903.07916* (2019).

[210] C. Zhang, S. Bengio, M. Hardt, B. Recht and O. Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM* 64.3 (2021), 107–115.

[211] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, Z. Zhang, H. Lin, Y. Sun, T. He, J. Mueller, R. Manmatha et al. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955* (2020).

[212] H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[213] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang and J. Sun. Alignedreid: Surpassing human-level performance in person re-identification. *arXiv preprint arXiv:1711.08184* (2017).

[214] Y. Zhang, J. D. Lee and M. I. Jordan. l1-regularized neural networks are improperly learnable in polynomial time. *International Conference on Machine Learning*. PMLR. 2016, 993–1001.

[215] Y. Zhang, R. Jia, H. Pei, W. Wang, B. Li and D. Song. The secret revealer: Generative model-inversion attacks against deep neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 253–261.

[216] Z. Zhang and M. R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836* (2018).

[217] Z. Zhang, C. Lan, W. Zeng, X. Jin and Z. Chen. Relation-Aware Global Attention for Person Re-identification. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, 3186–3195.

[218] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu and X.-S. Hua. Spatio-temporal autoencoder for video anomaly detection. *Proceedings of the 25th ACM international conference on Multimedia*. 2017, 1933–1941.

[219] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang and Q. Tian. Scalable person re-identification: A benchmark. *Proceedings of the IEEE International Conference on Computer Vision*. 2015, 1116–1124.

[220] L. Zheng, Y. Yang and A. G. Hauptmann. Person re-identification: Past, present and future. *arXiv preprint arXiv:1610.02984* (2016).

[221] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang and J. Kautz. Joint discriminative and generative learning for person re-identification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, 2138–2147.

[222] Z. Zhong, L. Zheng, D. Cao and S. Li. Re-ranking person re-identification with k-reciprocal encoding. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, 1318–1327.

[223] Z. Zhong, L. Zheng, G. Kang, S. Li and Y. Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896* (2017).

[224] K. Zhou, Y. Yang, A. Cavallaro and T. Xiang. Omni-scale feature learning for person re-identification. *Proceedings of the IEEE International Conference on Computer Vision*. 2019, 3702–3712.

[225] S. Zhu and K.-K. Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE transactions on Image Processing* 9.2 (2000), 287–290.

[226] Q. Zou, G. Lin, X. Jiang, X. Liu and X. Zeng. Sequence clustering in bioinformatics: an empirical study. *Briefings in bioinformatics* 21.1 (2020), 1–10.

PUBLICATIONS

PUBLICATION
I

Adaptive L2 Regularization in Person Re-Identification

X. Ni, L. Fang and H. Huttunen

2020 25th International Conference on Pattern Recognition (ICPR), 2021, 9601–9607
DOI: 10.1109/ICPR48806.2021.9412481

Adaptive L_2 Regularization in Person Re-Identification

Xingyang Ni, Liang Fang, Heikki Huttunen
Faculty of Information Technology and Communication Sciences
Tampere University, Finland
Email: {xingyang.ni, liang.fang, heikki.huttunen}@tuni.fi

***Abstract*—We introduce an adaptive L_2 regularization mechanism in the setting of person re-identification. In the literature, it is common practice to utilize hand-picked regularization factors which remain constant throughout the training procedure. Unlike existing approaches, the regularization factors in our proposed method are updated adaptively through backpropagation. This is achieved by incorporating trainable scalar variables as the regularization factors, which are further fed into a scaled hard sigmoid function. Extensive experiments on the Market-1501, DukeMTMC-reID and MSMT17 datasets validate the effectiveness of our framework. Most notably, we obtain state-of-the-art performance on MSMT17, which is the largest dataset for person re-identification. Source code is publicly available at https://github.com/nixingyang/AdaptiveL2Regularization.**

I. Introduction

Person re-identification involves retrieving corresponding samples from a gallery set based on the appearance of a query sample across multiple cameras. It is a challenging task since images may differ significantly due to variations in factors such as illumination, camera angle and human pose. On account of the availability of large-scale datasets [1], [2], [3], remarkable progress has been witnessed in recent studies on person re-identification, *e.g.*, utilizing local feature representations [4], [5], leveraging extra attribute labels [6], [7], improving policies for data augmentation [8], [9], adding a separate re-ranking step [10], [11] and switching to video-based datasets [12], [13].

L_2 regularization imposes constraints on the parameters of neural networks and adds penalties to the objective function during optimization. It is a commonly adopted technique which can improve the model's generalization ability. Although some works [14], [15], [16], [17] provide insights on the underlying mechanism of L_2 regularization, it is an understudied topic and has not received sufficient attention. In most literature, L_2 regularization is taken for granted, and the text dedicated to it is typically shrunk into one sentence as in [18]. On the other hand, existing approaches assign constant values to regularization factors in the training procedure, and such hyperparameters are hand-picked via hyperparameter optimization which is a tedious and time-consuming process. The primary purpose of this work is to address the bottleneck of conventional L_2 regularization and introduce a mechanism which learns the regularization factors and update those values adaptively.

In this paper, our major contributions are twofold:

- We introduce an adaptive L_2 regularization mechanism, which optimizes each regularization factor adaptively as the training procedure progresses.
- With the proposed framework, we obtain state-of-the-art performance on MSMT17, which is the largest dataset for person re-identification.

The rest of this paper is organized as follows. Section II reviews important works in person re-identification and L_2 regularization. In Section III, we present the essential components of our baseline, alongside the proposed adaptive L_2 regularization mechanism. Section IV describes the details of our experiments, including datasets, evaluation metrics and comprehensive analysis of our proposed method. Finally, Section V concludes the paper.

II. Related work

In this section, we give a brief overview of two distinct research topics, namely, person re-identification and L_2 regularization.

A. Person Re-Identification

Utilizing local feature representations which are specific to certain regions, has been shown successful. Varior *et al*. [4] propose a Long Short-Term Memory architecture which models the spatial dependency and thus extracts more discriminative local features. Sun *et al*. [5] apply a uniform partition strategy which divides the feature maps evenly into individual parts, and the part-informed features are concatenated to form the final descriptor.

Besides, methods based on auxiliary features are advocated, aiming to utilize extra attributes in addition to the identity labels. Su *et al*. [6] shows that learning mid-level human attributes can be used to address the challenge of visual appearance variations. Specifically, an attribute prediction model is trained on an independent dataset which contains the attribute labels. Lin *et al*. [7] manually annotate attribute labels which contain detailed local descriptions. A multi-task network is proposed to learn an embedding for re-identification and also predict the attribute labels. In addition to the performance improvement in re-identification, such system can speed up the retrieval process by ten times.

By applying random manipulations on training samples, data augmentation has played an essential role in suppressing the overfitting issue and improving the generalization of models. Zhong *et al*. [8] introduce an approach which erases the pixel values in a random rectangle region during training. By contrast, Dai *et al*. [9] suggest dropping the same region for all samples in the same batch. Such feature dropping branch strengthens the learned features of local regions.

Adding a separate re-ranking step to refine the initial ranking list can lead to significant improvements. Zhong *et al*. [10] develop a k-reciprocal encoding method based on the hypothesis that a gallery image is more likely to be a true match if it is similar to the probe in the k-reciprocal nearest neighbours. Zhou *et al*. [11] rank the predictions with a specified local metric by exploiting negative samples for each online query, rather than implementing a general global metric for all query probes.

Lastly, some works shift the emphasis from image-based to video-based person re-identification. Liu *et al*. [12] introduce a spatio-temporal body-action model which exploits the periodicity exhibited by a walking person in a video sequence. Alternatively, Dai *et al*. [13] present a learning approach which unifies two modules: one module extracts the features of consecutive frames, and the other module tackles the poor spatial alignment of moving pedestrians.

B. L_2 regularization

Laarhoven [14] prove that L_2 regularization would not regularize properly in the presence of normalization operations, *i.e*., batch normalization [19] and weight normalization [20]. Instead, L_2 regularization will affect the scale of weights, and therefore it has an influence on the effective learning rate.

Similarly, Hoffer *et al*. [15] investigate how does applying weight decay before batch normalization affect learning dynamics. Combining weight decay and batch normalization would constrain the norm to a small range of values and lead to a more stable step size for the weight direction. It enables better control over the effective step size through the learning rate.

Later on, Loshchilov *et al*. [16] clarify a long-established misunderstanding that L_2 regularization is equivalent to weight decay. The aforementioned statement does not hold when applying adaptive gradient algorithms, *e.g*., Adam [21]. Furthermore, they suggest decoupling the weight decay from the optimization steps, and it leads to the original formulation of weight decay.

Most recently, Lewkowycz *et al*. [17] present an empirical study on the relations among the L_2 coefficient, the learning rate, and the number of training epochs and the performance of the model. In a similar manner as learning rate schedules, a manually designed schedule for the L2 parameter is proposed to increase training speed and boost model's performance.

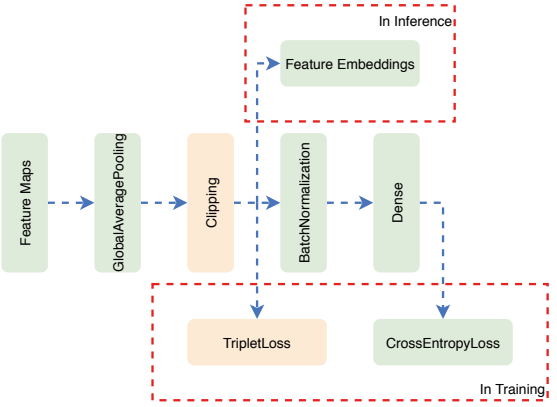


Fig. 1. Structure of an objective module. In the training procedure, two objective functions are applied: triplet loss [22] and categorical cross-entropy loss. In the inference procedure, the feature embeddings before the batch normalization [19] layer are extracted as the representations. Note that blocks in yellow are excluded from the minimal setup.

III. Proposed Method

In this section, we first present a minimal setup for person re-identification. Later on, we explain five components that contribute to significant improvements in performance and use the resulting method as the baseline in our study. Most importantly, we discuss the proposed adaptive L_2 regularization mechanism at the end.

A. Minimal setup

Backbone: ResNet50 [18], initialized with ImageNet [23] pre-trained weights, is selected as the backbone model. For convenience, it is separated into five individual blocks, *i.e*., block 1-5, as illustrated in Figure 2. Additionally, the stride arguments of the first convolution layer in block 5 are set to 1, rather than default value 2. This enlarges the feature maps by a scale factor of 2 along with both height and width dimensions while reusing the pre-trained weights and keeping the total amount of parameters identical.

Objective module: Figure 1 demonstrates the structure of an objective module that converts the feature maps to learning objectives. A global average pooling layer squeezes the spatial dimensions in the feature maps, and the following batch normalization [19] layer generates the normalized feature vectors. The concluding fully-connected layer does not contain a bias vector, and it produces the predicted probabilities of each unique identity so that the model can be optimized using the categorical cross-entropy loss. In the inference procedure, the feature embeddings before the batch normalization layer are extracted as the representations, and cosine distance is adopted to measure the distance between two samples.

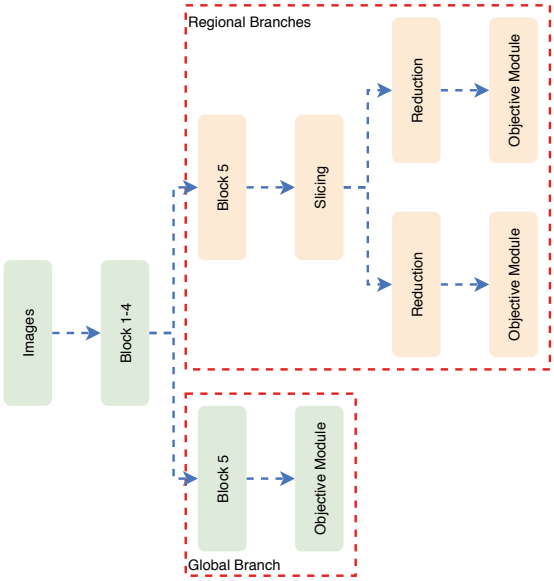


Fig. 2. Topology of the overall model. Feature embeddings from multiple objective modules are concatenated in the inference procedure. Note that blocks in yellow are excluded from the minimal setup.

Overall topology: The topology of the overall model is shown in Figure 2. It is to be observed that the minimal setup only contains the global branch. Given a batch of images, the individual blocks from the backbone model utilize successively, and an objective module is appended at the end.

Data augmentation: The image is resized to target resolution using a bilinear interpolation method. Besides, the image is flipped horizontally at random with probability set to 0.5. Zero paddings are added to all sides of the image, *i.e.*, the top, bottom, left, and right sides. A random part with target resolution is subsequently cropped.

Learning rate: The learning rate increases linearly from a low value to the pre-defined base learning rate in the early stage of the training procedure, and it is divided by ten once the performance on the validation set plateaus. On the one hand, the warmup strategy suppresses the distorted gradient issue at the beginning [24]. On the other hand, periodically reducing the learning rate boosts the performance even further.

Label smoothing: The label smoothing regularization [25] is applied alongside with the categorical cross-entropy loss function. Given a sample with ground truth label $y \in \{1, 2, \ldots, N\}$, the one-hot encoded label $\boldsymbol{q}(i)$ equals to 1 only if the index i is as the same as label y, and 0 otherwise. The smoothed label introduces a hyperparameter $\epsilon \in (0, 1)$ and is calculated as:

$$\boldsymbol{q}'(i) = (1 - \epsilon)\boldsymbol{q}(i) + \frac{\epsilon}{N}. \tag{1}$$

B. Baseline

Triplet loss: As highlighted in Figure 1, the triplet loss [22] is applied on the feature embeddings before the batch normalization layer. It mines the moderate hard triplets instead of all possible combinations of triplets, given that using all possible triplets may destabilize the training procedure. Considering that multiple loss functions are present, the weighting coefficients of each loss function are set to 1 on account of simplicity.

Regional branches: In addition to the global branch, two regional branches are integrated into the model. Figure 2 illustrates the diagram of those regional branches. Firstly, the block 5 from the backbone model is replicated, and it is not shared with the global branch. Secondly, we adopt the uniform partition scheme as in [5]. The slicing layer explicitly divides the feature maps into two horizontal stripes. Lastly, dimensionality reduction is performed using a convolutional layer on each stripe. Separate objective modules are appended afterwards. In the inference procedure, feature embeddings from multiple objective modules are concatenated.

Random erasing: In addition to random horizontal flipping, random erasing [8] is utilized in data augmentation. During training, it erases an area of original images to improve the robustness of the model, especially for the occlusion cases.

Clipping: The clipping layer is inserted between the global average pooling layer and the batch normalization layer in Figure 1. It performs element-wise value clipping so that values in its output are contained in a closed interval. The clipping layer works in a similar manner as the ReLU-n units [26], and it relieves optimization difficulties in the succeeding triplet loss [22].

L_2 regularization: Conventional L_2 regularization is utilized to all trainable parameters, *i.e.*, the regularization factors remain constant throughout the training procedure. Additionally, those regularization factors need to be hand-picked via hyperparameter optimization.

C. Adaptive L_2 regularization

A neural network consists of a set of N distinct parameters,

$$P = \{\boldsymbol{w}_n \mid n = 1, \ldots, N\}, \tag{2}$$

with P containing all trainable parameters. Each $\boldsymbol{w}_n$ is an array which could be a vector, a matrix or a 3rd-order tensor. For example, the kernel and bias terms in a fully-connected layer are a matrix and a vector, respectively.

Conventional L_2 regularization imposes an additional penalty term to the objective function, which can be formulated as follows:

$$L_\lambda(P) = L(P) + \lambda \sum_{n=1}^{N} \|\boldsymbol{w}_n\|_2^2, \tag{3}$$

where $L(P)$ and $L_\lambda(P)$ denote the original and updated objective functions, respectively. In our case (see Figures 1 and 2), $L(P)$ is a weighted sum of triplet loss [22] and categorical cross-entropy loss functions. In addition, $\|\boldsymbol{w}_n\|_2^2$

refers to the square of the L_2 norm[1] of $\boldsymbol{w}_n$, and the constant coefficient $\lambda \in \mathbb{R}_+$ defines the regularization strength.

One may wish to add penalties in a different way, *e.g.*, applying lighter regularization in the early layers but stronger in the last ones. Thus, it is possible to generalize even further, *i.e.*, defining a unique coefficient for each $\|\boldsymbol{w}_n\|_2^2$:

$$L_\lambda(P) = L(P) + \sum_{n=1}^{N} \left(\lambda_n \|\boldsymbol{w}_n\|_2^2\right), \tag{4}$$

where each parameter $\boldsymbol{w}_n$ is associated with an individual regularization factor $\lambda_n \in \mathbb{R}_+$.

Obviously, it is infeasible to manually fine-tune those regularization factors λ_n for $n = 1, \ldots, N$ one by one, since N is in the order of 100 for models trained with ResNet50. Therefore, we treat them as any other learnable parameters and find suitable values from the data itself.

To make the aforementioned regularization factors adaptive, a straightforward extension is obtained by replacing the predefined constant λ_n with scalar variables which are trainable through backpropagation. After the modification, Equation 4 remains unchanged while $\lambda_n \in \mathbb{R}$. However, such an approach without any constraints on λ_n will fail. Namely, setting negative values for λ_n allows naively increasing $\|\boldsymbol{w}_n\|_2^2$ so that $L_\lambda(P)$ decreases sharply. In other words, the L_2 regularization penalties would become dominant in the optimization process. Thus the model collapses and would not learn useful feature embeddings.

To address the collapse problem, we apply the hard sigmoid function which assures that the regularization factor λ_n would always have non-negative values. The hard sigmoid function is defined as

$$f(x) = \begin{cases} 0, & \text{if } x < -c \\ 1, & \text{if } x > c \\ x/(2c) + 0.5, & \text{otherwise.} \end{cases} \tag{5}$$

In our experiments, we use $c = 2.5$, but any other positive values can be used as well.

The regularization factor λ_n is obtained by applying the hard sigmoid on the raw parameters as

$$\lambda_n = f(\theta_n), \tag{6}$$

where $\theta_n \in \mathbb{R}$ $(n = 1, \ldots, N)$ are the trainable scalar variables. Furthermore, we introduce a hyperparameter $A \in \mathbb{R}_+$ which represents the amplitude. Hence, we get

$$\lambda_n = A f(\theta_n). \tag{7}$$

The amplitude A offers flexibility of avoiding excessively large regularization factors which could deteriorate the training procedure. Combining Equation (4) and (7) gives

$$L_\lambda(P) = L(P) + \sum_{n=1}^{N} \left(A f(\theta_n) \|\boldsymbol{w}_n\|_2^2\right). \tag{8}$$

[1]We define $\|\boldsymbol{w}\|_2^2$ to denote the sum of squares of all elements also when $\boldsymbol{w}$ is a matrix or a 3rd-order tensor.

TABLE I
COMPARISON OF THREE PERSON RE-IDENTIFICATION DATASETS, NAMELY, MARKET-1501 [1], DUKEMTMC-REID [2] AND MSMT17 [3].

Dataset	Market-1501	DukeMTMC-reID	MSMT17
Train Samples	12,936	16,522	32,621
Train Identities	751	702	1,041
Test Query Samples	3,368	2,228	11,659
Test Gallery Samples	15,913	17,661	82,161
Test Identities	751	1,110	3,060
Cameras	6	8	15

IV. EXPERIMENTS

In this section, we explain datasets, evaluation metrics and comprehensive analysis of our proposed method.

A. Datasets

We conduct experiments on three person re-identification datasets, namely, Market-1501 [1], DukeMTMC-reID [2] and MSMT17 [3]. Table I makes a comparison of those datasets. The MSMT17 dataset outshines the other two due to its large scale.

The Market-1501 dataset is collected with six different cameras in total. It contains 32,217 images from 1,501 pedestrians, and at least two cameras capture each pedestrian. The training set includes 751 pedestrians with 12,936 images, while the test set consists of the remaining images from 750 pedestrians and one distractor class.

The DukeMTMC-reID dataset includes 1,404 pedestrians that appear in at least two cameras and 408 pedestrians that appear only in one camera. The training and test sets contain 16,522 and 19,889 images, respectively. The query and gallery samples in the test set are randomly split.

The MSMT17 dataset is the largest person re-identification dataset which is publicly available, as of July 2020. It contains 126,441 images from 4,101 pedestrians, while 3 indoor cameras and 12 outdoor cameras are employed. In particular, the test set has approximately three times as much samples as the training set. Such setting motivates the research community to leverage a limited number of training samples that are available since data annotation is costly.

B. Evaluation metrics

Following the practices in [1], two evaluation metrics are applied to measure the performance, *i.e.*, mean Average Precision (mAP), and Cumulative Matching Characteristic (CMC) rank-k accuracy. The metrics take the distance matrix between query and gallery samples, in conjunction with the ground truth identities and camera IDs as input arguments. Gallery samples are discarded if they have been taken from the same camera as the query sample. As a result, greater emphasis is laid on the performance in the cross-camera setting.

Since the query samples may have multiple ground truth matches in the gallery set, mAP is preferable than rank-k accuracy for the reason that mAP considers both precision and recall.

TABLE II
PERFORMANCE COMPARISONS AMONG BASELINE, ADAPTIVE L_2 REGULARIZATION AND EXISTING APPROACHES. THE MAP SCORE ON MSMT17 IS THE MOST RELIABLE INDICATOR OF PERFORMANCE. R1: RANK-1 ACCURACY. -: NOT AVAILABLE. †: RE-RANKING [10] IS APPLIED.

Method	Venue	Backbone	Market-1501		DukeMTMC-reID		MSMT17	
			mAP	R1	mAP	R1	mAP	R1
Annotators [27]	arXiv 2017	-	-	93.5	-	-	-	-
PCB [5]	ECCV 2018	ResNet50	81.6	93.8	69.2	83.3	-	-
IANet [28]	CVPR 2019	ResNet50	83.1	94.4	73.4	87.1	46.8	75.5
AANet [29]	CVPR 2019	ResNet50	82.5	93.9	72.6	86.4	-	-
CAMA [30]	CVPR 2019	ResNet50	84.5	94.7	72.9	85.8	-	-
DGNet [31]	CVPR 2019	ResNet50	86.0	94.8	74.8	86.6	52.3	77.2
OSNet [32]	ICCV 2019	OSNet	84.9	94.8	73.5	88.6	52.9	78.7
MHN [33]	ICCV 2019	ResNet50	85.0	95.1	77.2	89.1	-	-
BDB [9]	ICCV 2019	ResNet50	86.7	95.3	76.0	89.0	-	-
BAT-net [34]	ICCV 2019	GoogLeNet	87.4	95.1	77.3	87.7	56.8	79.5
SNR [35]	CVPR 2020	ResNet50	84.7	94.4	72.9	84.4	-	-
HOReID [36]	CVPR 2020	ResNet50	84.9	94.2	75.6	86.9	-	-
RGA-SC [37]	CVPR 2020	ResNet50	88.4	96.1	-	-	57.5	80.3
SCSN [38]	CVPR 2020	ResNet50	88.5	95.7	79.0	91.0	58.5	83.8
Baseline (Ours)	-	ResNet50	87.2	94.6	78.9	88.0	57.7	79.1
Adaptive L_2 Regularization (Ours)	-	ResNet50	88.3	95.3	79.9	88.9	59.4	79.6
		ResNet101	88.6	94.8	80.6	89.2	61.9	81.3
		ResNet152	88.9	95.6	81.0	90.2	62.2	81.7
		ResNet152†	94.4	96.0	90.7	92.2	76.7	84.9

TABLE III
ABLATION STUDY OF BASELINE USING THE RESNET50 BACKBONE. R1: RANK-1 ACCURACY.

Method	Market-1501		DukeMTMC-reID		MSMT17	
	mAP	R1	mAP	R1	mAP	R1
Minimal Setup	28.3	60.0	28.7	49.9	11.4	34.0
+ Triplet Loss	79.9	92.0	68.8	82.2	44.0	70.9
+ Regional Branches	81.3	93.3	71.2	84.2	47.9	74.2
+ Random Erasing	85.8	94.4	76.6	87.0	54.1	77.0
+ Clipping	86.8	94.3	78.1	87.6	56.5	78.4
+ L_2 regularization	87.2	94.6	78.9	88.0	57.7	79.1

C. Ablation study of baseline

The baseline differs from the minimal setup in five aspects, as discussed in Section III-B. Table III presents an ablation study to demonstrate how each component contributes to the performance on person re-identification. On the one hand, the triplet loss [22] brings the most significant improvements on all three datasets. The boost is due to the fact that the triplet loss is applied to the feature embeddings which are retrieved in the inference procedure (see Figure 1). Since the triplet loss directly optimizes the model in a manner comparable to similarity search, it closes the gap between the training and inference procedures. On the other hand, the other four components bring moderate improvements. It is conceivable that the model reaches better generalization by using hand-picked L_2 regularization factors which remain constant throughout the training procedure.

D. Comparisons with existing approaches

Table II shows performance comparisons among baseline, adaptive L_2 regularization and existing approaches.

Firstly, all methods listed in Table II have surpassed the best-performing human annotators [27] on the Market-1501 dataset. In light of the scale of the Market-1501 and DukeMTMC-reID datasets (see Table I), these two small-scale datasets might have been saturated and more emphasis should be put on the MSMT17 dataset. Since mAP is preferable than rank-k accuracy, the mAP score on MSMT17 is the most reliable indicator of performance.

Secondly, the proposed adaptive L_2 regularization mechanism contributes with decent improvements to the baseline, especially on MSMT17 in which the mAP score increases from 57.7% to 59.4%. Among methods which utilize the ResNet50 backbone, it is noteworthy that the adaptive L_2 regularization method obtains the state-of-the-art performance on DukeMTMC-reID and MSMT17, very close to state-of-the-art performance on Market-1501.

Last but not least, deeper backbones (*i.e.*, ResNet101 and ResNet152) further improve the performance, at the cost of extra computations. Attributed to the re-ranking [10] method which exploits the test data in the inference procedure, new milestones have been accomplished, *i.e.*, the mAP scores on Market-1501, DukeMTMC-reID and MSMT17 stand at 94.4%, 90.7% and 76.7%, respectively.

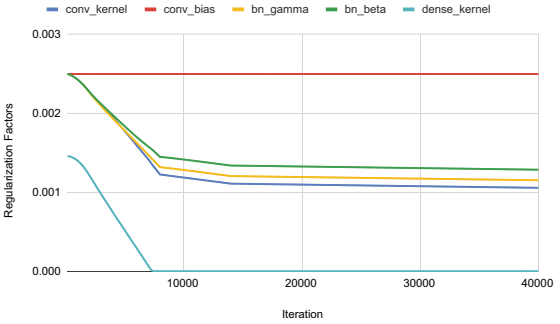


Fig. 3. The median value of regularization factors in each category, with respect to the number of iterations.

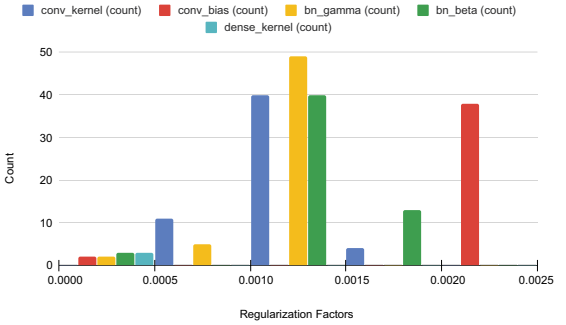


Fig. 4. Histogram of regularization factors in the last epoch.

E. Quantitative analysis of regularization factors

Depending on the associated distinct parameter (see Equation 2), the regularization factors can be classified into five categories: *conv_kernel*, *conv_bias*, *bn_gamma*, *bn_beta* and *dense_kernel*, where *conv*, *bn* and *dense* denote the convolutional, batch normalization and fully-connected layers. In the following, we examine the regularization factors for a model trained on MSMT17 using the ResNet50 backbone.

Figure 3 visualizes the median value of regularization factors in each category, with respect to the number of iterations. Note that the learning rate gets reduced at iterations 8000 and 14000. While *conv_kernel*, *bn_gamma* and *bn_beta* behave similarly, *conv_bias* remains constant throughout the training procedure and *dense_kernel* drops to 0 in the early stage.

Figure 4 demonstrates a histogram of regularization factors in the last epoch, *i.e.*, the training procedure completes. The interval $[0, 0.0025]$ is divided evenly into five buckets. For regularization factors from the same category, the values could differ significantly, *e.g.*, 2 and 38 regularization factors from *conv_bias* fall within the interval $[0, 0.0005]$ and $[0.0020, 0.0025]$, respectively. To be specific, the regularization factors from *conv_bias* in the two *Reduction* blocks are 0 (see Figure 2). If omitting the effects of the *Clipping* layer in Figure 1, those convolutional layers are followed by batch normalization layers which intrinsically cancels out the bias terms in aforementioned convolutional layers. Consequently, such regularization factors would converge to 0. In summary, this phenomenon reflects the superiority of our proposed method, in which each regularization factor is optimized separately.

F. Qualitative analysis of predictions

Figure 5 illustrates selected query samples with corresponding top 5 matches from the gallery set. Although query samples and erroneous matches may have similar appearances, minor differences could be observed under careful inspection, *e.g.*, the dissimilarity between backpacks. Furthermore, our models could retrieve correct matches even in the presence of large illumination changes, *e.g.*, the two examples from the MSMT17 dataset.

Fig. 5. Selected query samples with corresponding top 5 matches from the gallery set. Images with orange, green and red border are query samples, correct matches and erroneous matches, respectively.

V. Conclusion

In this work, we revisited L_2 regularization in neural networks. Unlike employing constant and hand-picked regularization factors, our proposed method optimizes the strength of L_2 regularization adaptively through backpropagation. More specifically, we applied a scaled hard sigmoid function to trainable scalar variables and used those as the regularization factors. Extensive experiments substantiate that our framework boosts model's performance, and we obtained state-of-the-art performance on MSMT17 which is the largest person re-identification dataset. Although the current study is constrained within the domain of person re-identification, it is self-evident that the proposed adaptive L_2 regularization mechanism can be seamlessly integrated into the training procedure of any neural networks. Accordingly, further studies can be conducted on other topics, *e.g.*, image classification, image retrieval and object detection.

Acknowledgement

This work was financially supported by Business Finland project 408/31/2018 MIDAS.

REFERENCES

[1] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1116–1124.

[2] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 17–35.

[3] L. Wei, S. Zhang, W. Gao, and Q. Tian, "Person transfer gan to bridge domain gap for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 79–88.

[4] R. R. Varior, B. Shuai, J. Lu, D. Xu, and G. Wang, "A siamese long short-term memory architecture for human re-identification," in *European conference on computer vision*. Springer, 2016, pp. 135–153.

[5] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 480–496.

[6] C. Su, S. Zhang, J. Xing, W. Gao, and Q. Tian, "Deep attributes driven multi-camera person re-identification," in *European conference on computer vision*. Springer, 2016, pp. 475–491.

[7] Y. Lin, L. Zheng, Z. Zheng, Y. Wu, Z. Hu, C. Yan, and Y. Yang, "Improving person re-identification by attribute and identity learning," *Pattern Recognition*, 2019.

[8] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

[9] Z. Dai, M. Chen, X. Gu, S. Zhu, and P. Tan, "Batch DropBlock network for person re-identification and beyond," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3691–3701.

[10] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1318–1327.

[11] J. Zhou, P. Yu, W. Tang, and Y. Wu, "Efficient online local metric adaptation via negative samples for person re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2420–2428.

[12] K. Liu, B. Ma, W. Zhang, and R. Huang, "A spatio-temporal appearance representation for viceo-based pedestrian re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3810–3818.

[13] J. Dai, P. Zhang, D. Wang, H. Lu, and H. Wang, "Video person re-identification by temporal residual learning," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1366–1377, 2018.

[14] T. Van Laarhoven, "L2 regularization versus batch and weight normalization," *arXiv preprint arXiv:1706.05350*, 2017.

[15] E. Hoffer, R. Banner, I. Golan, and D. Soudry, "Norm matters: efficient and accurate normalization schemes in deep networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 2160–2170.

[16] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2018.

[17] A. Lewkowycz and G. Gur-Ari, "On the training dynamics of deep networks with $ L_2 $ regularization," *arXiv preprint arXiv:2006.08643*, 2020.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[20] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 901–909.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[24] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *arXiv preprint arXiv:1908.03265*, 2019.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[26] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on cifar-10," *Unpublished manuscript*, vol. 40, no. 7, pp. 1–9, 2010.

[27] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, "Alignedreid: Surpassing human-level performance in person re-identification," *arXiv preprint arXiv:1711.08184*, 2017.

[28] R. Hou, B. Ma, H. Chang, X. Gu, S. Shan, and X. Chen, "Interaction-and-aggregation network for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9317–9326.

[29] C.-P. Tay, S. Roy, and K.-H. Yap, "Aanet: Attribute attention network for person re-identifications," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7134–7143.

[30] W. Yang, H. Huang, Z. Zhang, X. Chen, K. Huang, and S. Zhang, "Towards rich feature discovery with class activation maps augmentation for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1389–1398.

[31] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang, and J. Kautz, "Joint discriminative and generative learning for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2138–2147.

[32] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3702–3712.

[33] B. Chen, W. Deng, and J. Hu, "Mixed high-order attention network for person re-identification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 371–381.

[34] P. Fang, J. Zhou, S. K. Roy, L. Petersson, and M. Harandi, "Bilinear attention networks for person retrieval," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 8030–8039.

[35] X. Jin, C. Lan, W. Zeng, Z. Chen, and L. Zhang, "Style normalization and restitution for generalizable person re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3143–3152.

[36] G. Wang, S. Yang, H. Liu, Z. Wang, Y. Yang, S. Wang, G. Yu, E. Zhou, and J. Sun, "High-Order Information Matters: Learning Relation and Topology for Occluded Person Re-Identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6449–6458.

[37] Z. Zhang, C. Lan, W. Zeng, X. Jin, and Z. Chen, "Relation-Aware Global Attention for Person Re-identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3186–3195.

[38] X. Chen, C. Fu, Y. Zhao, F. Zheng, J. Song, R. Ji, and Y. Yang, "Salience-Guided Cascaded Suppression Network for Person Re-Identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3300–3310.

PUBLICATION
II

FlipReID: Closing the Gap Between Training and Inference in Person Re-Identification

X. Ni and E. Rahtu

2021 9th European Workshop on Visual Information Processing (EUVIP), 2021, 1–6
DOI: 10.1109/EUVIP50544.2021.9484010

FLIPREID: CLOSING THE GAP BETWEEN TRAINING AND INFERENCE IN PERSON RE-IDENTIFICATION

Xingyang Ni *Esa Rahtu*

Tampere University, Finland

ABSTRACT

Since neural networks are data-hungry, incorporating data augmentation in training is a widely adopted technique that enlarges datasets and improves generalization. On the other hand, aggregating predictions of multiple augmented samples (*i.e*., test-time augmentation) could boost performance even further. In the context of person re-identification models, it is common practice to extract embeddings for both the original images and their horizontally flipped variants. The final representation is the mean of the aforementioned feature vectors. However, such scheme results in a gap between training and inference, *i.e*., the mean feature vectors calculated in inference are not part of the training pipeline. In this study, we devise the FlipReID structure with the flipping loss to address this issue. More specifically, models using the FlipReID structure are trained on the original images and the flipped images simultaneously, and incorporating the flipping loss minimizes the mean squared error between feature vectors of corresponding image pairs. Extensive experiments show that our method brings consistent improvements. In particular, we set a new record for MSMT17 which is the largest person re-identification dataset. The source code is available at `https://github.com/nixingyang/FlipReID`.

***Index Terms*—** Person re-identification, test-time augmentation

1. INTRODUCTION

The purpose of person re-identification is retrieving the person of interest across multiple cameras, based on either images or videos [1]. While large-scale datasets [2, 3, 4] have been collected, academic research is progressing in three major directions: feature representation learning, deep metric learning, and ranking optimization.

Feature representation learning refers to developing strategies for feature construction [1]. Early works only extract a global representation for each person image [5, 6]. Later on, incorporating local features from body parts/regions has been proven beneficial [7, 8]. In addition, Lin *et al*. [9] propose leveraging the auxiliary attributes, while Wei *et al*. [4] generate synthetic images to reduce the performance drop when training and testing on different datasets.

Deep metric learning studies objective functions which are utilized to optimize neural networks [1]. With the identity loss, a model classifies the identities, and each identity is a distinct class [5, 6]. By comparison, the verification loss exploits the pairwise relationship between samples. Varior *et al*. [7] adopt the contrastive loss function for learning the embeddings, and Zheng *et al*. [10] use the binary verification loss by feeding positive and negative pairs. Lastly, the triplet loss is based on the assumption that the distance between the positive pair should be smaller than the negative pair [11].

Ranking optimization is a post-processing step that refines the retrieved ranking list [1]. Liu *et al*. [12] devise a method that requires only one negative feedback. Wang *et al*. [13] propose a hybrid model that learns cumulatively from users' feedback, and it is scalable to large-scale gallery sets. Since human interaction is time-consuming, Zhong *et al*. [14] opt for a fully automated solution instead. The pairwise distance between query and gallery samples is updated by comparing their k-reciprocal nearest neighbors.

Apart from person re-identification, data augmentation is the other subject of this study. Transformations are applied on the samples while preserving the labels, thus avoiding the need for re-annotating [15]. It is widely adopted in the training procedure to diversify samples, reduce overfitting and improve model robustness. In the mixup [16] work, models are trained on convex combinations of pairs of samples and their labels. Cubuk *et al*. [17] propose a search algorithm that finds the best data augmentation policies for the task at hand. The learned policies perform better than manually designed policies and are transferable between datasets. Optionally, data augmentation can be utilized in the inference procedure as well, *i.e*., one could generate predictions of multiple augmented samples and aggregate them into a more accurate prediction. Employing test-time augmentation improves performance at the cost of extra computations.

For image classification models such as AlexNet [18] and ResNet [19], a 10-crop testing method is applied. Five patches are extracted from the original image, *i.e*., four corner patches and one center patch. Another five patches are obtained from the horizontal reflection. Since the output of the last dense layer with softmax activation represents the probabilities of classes, it is trivial to use the mean of these ten patches' predictions.

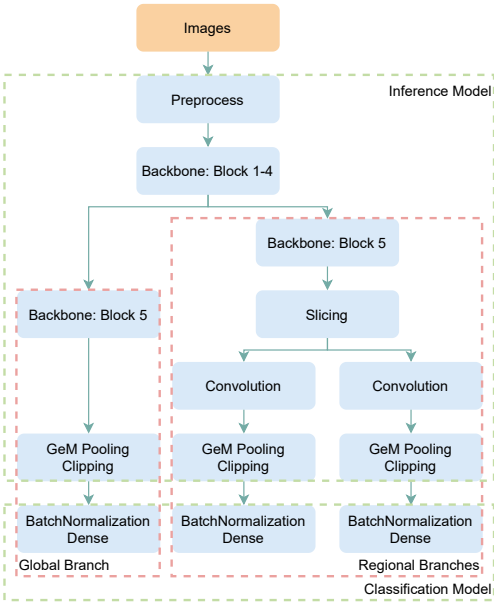


Fig. 1. Overall structure of the baseline method.

For person re-identification models such as MGN [20] and FastReID [21], features are extracted from both the original images and their horizontally flipped variants. Afterward, the mean of these feature vectors is used for evaluation. Even though this practice brings performance improvements, it lacks a thorough explanation for aggregating feature vectors by calculating the mean. More specifically, such mean feature vectors are not involved in optimizing the model, and calculating the mean in inference deviates from the training objectives. As a consequence, there exists a gap between training and inference.

In this study, we examine test-time augmentation in person re-identification. Our contribution is twofold:

- We identify a largely neglected issue, *i.e.*, utilizing the mean feature vectors of multiple augmented samples for evaluation results in suboptimal performance due to the gap between training and inference.
- We devise the FlipReID structure with the flipping loss. For models using the FlipReID structure, the original images and the flipped images are both used for training. Additionally, incorporating the flipping loss minimizes the mean squared error between feature vectors of corresponding image pairs. The resulting method achieves state-of-the-art performance on popular person re-identification datasets.

2. PROPOSED METHOD

2.1. Baseline

We leverage the method described in [22] as the baseline since it achieves high performance among recent works in person re-identification. Figure 1 illustrates the overall structure of the baseline method, and its essential components are explained as follows.

Backbone. Given images with pixel values in $\{0, \dots, 255\}$, the pre-processing layer scales pixels to $[0, 1]$ and normalizes each channel. The backbone model is an image classification model pre-trained on ImageNet [23], *e.g.*, ResNet [19], IBN-ResNet [24] and ResNeSt [25]. Due to its pyramidal architecture, the backbone model can be separated into consecutive blocks, and higher blocks refine feature maps extracted by lower blocks.

Global Branch. Following the last block of backbone, the Generalized-Mean (GeM) pooling [26] layer learns a trainable power parameter, and it generalizes the max and average operations. The subsequent clipping layer performs element-wise value clipping, which introduces constraints on feature vectors. Lastly, the batch normalization [27] and dense layers produce the probabilities of classes.

Regional Branches. The regional branches differ from the global branch in two aspects. On the one hand, the slicing layer [8] divides the feature maps along the height axis. For example, it outputs the upper and lower stripes in the case of using two partitions. On the other hand, the following convolution layer reduces the number of channels so that the resulting feature vector is not excessively long.

Objective Function. In the training procedure, the objective function is a weighted sum of batch hard triplet loss [11] and categorical cross-entropy loss. The batch hard triplet loss utilizes the hardest positive and negative samples within the batch when forming the triplets, and it is applied to the clipping layer's output. By contrast, the categorical cross-entropy loss measures the difference between the ground truth and predicted probability distributions, and it is applied to the dense layer's output.

Sub-Models. In the inference procedure, the outputs of the clipping layers from both global branch and regional branches are concatenated to get the embedded feature vector. From another perspective, the pipeline starting from the pre-processing layer to the clipping layers can be viewed as the inference model, while the remaining batch normalization and dense layers constitute the classification model.

Mini-Batch. Due to the nature of the batch hard triplet loss [11], each mini-batch comprises samples from both the same and different identities so that the positive and negative exemplars can be selected. To alleviate the overfitting issue, random horizontal flipping, random grayscale patch replacement [28] and random erasing [29] are used as the data augmentation policies.

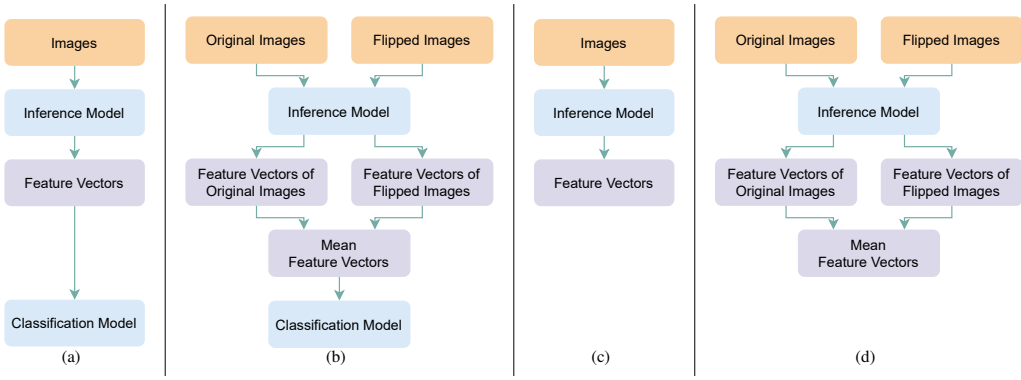


Fig. 2. Overview of the training and inference procedures in different settings: (a) training in the baseline method; (b) training in the FlipReID method; (c) inference using single image; (d) inference using double images.

2.2. FlipReID

Training. Figure 2(a) visualizes a higher level of abstraction of Figure 1. In the baseline method, random horizontal flipping is adopted for data augmentation. However, only one image will be used for one sample at a time, *i.e.*, either the original image or the flipped image. By contrast, Figure 2(b) shows the diagram of the FlipReID method. Both the original images and the flipped images are feed into the inference model, and the classification model takes in the mean feature vectors. As an optional loss term, we introduce the flipping loss, which calculates the mean squared error between the feature vectors of the original images and the flipped images.

Inference. Figure 2(c) and 2(d) show the inference pipeline using single image and double images, respectively. In Figure 2(c), data augmentation is disabled, and feature vectors are extracted from the original images. By comparison, horizontally flipped images are used in Figure 2(d), and the final representation is the mean feature vectors of the original images and horizontal reflections.

Options for Training and Inference. Figure 2(a) and 2(b) provide two choices for training, while Figure 2(c) and 2(d) offer two inference strategies. This gives the following four options for training and inference:

- Figure 2(a), 2(c): Inference is consistent with training, and the classification model is discarded in inference.
- Figure 2(a), 2(d): There exists a gap between training and inference, *i.e.*, the classification model is only trained on feature vectors of single image.
- Figure 2(b), 2(c): There exists a gap between training and inference, *i.e.*, the classification model is only trained on feature vectors of double images.
- Figure 2(b), 2(d): Inference is consistent with training, and the classification model is discarded in inference.

3. EXPERIMENTS

3.1. Datasets

Experiments are carried out on three person re-identification datasets, *i.e.*, Market-1501 [2], DukeMTMC-reID [3] and MSMT17 [4].

Market-1501. It consists of 32,217 images taken from 1,501 pedestrians. Each pedestrian is captured by at least two cameras, and there are six cameras in total.

DukeMTMC-reID. Eight cameras were deployed to collect the dataset. The training set contains 702 pedestrians with 16,522 images, and the test set contains 1,110 pedestrians with 19,889 images. Note that 408 pedestrians appear only in one camera, and those samples serve as distractors.

MSMT17. Compared with the previous two datasets, the MSMT17 dataset is closer to real scenarios due to its diversity. Collected by three indoor cameras and twelve outdoor cameras, it comprises 126,441 images from 4,101 pedestrians. The ratio of training to test samples is set to 1:3 with the intention of limiting available training samples and therefore emphasizing effective training methods.

3.2. Evaluation metrics

Models are evaluated using mean Average Precision (mAP) and Cumulative Matching Characteristic (CMC) rank-k accuracy [2]. The mAP score is preferable to the CMC rank-k accuracy because the former metric considers both precision and recall while the latter metric does not report the performance on hard positive samples. Furthermore, gallery samples taken from the same camera as the query sample are discarded during evaluation so that the metrics would emphasize the performance in the cross-camera setting.

Table 1. Performance comparisons among existing studies, our baseline method, and our FlipReID method. †: inference using single image (see Figure 2(c)). ‡: inference using double images (see Figure 2(d)). §: re-ranking [14] is applied.

Method	Backbone	Market-1501		DukeMTMC-reID		MSMT17	
		mAP	R1	mAP	R1	mAP	R1
PCB, ECCV 2018 [8]	ResNet50	81.6	93.8	69.2	83.3	-	-
BoT, CVPRW 2019 [6, 21]	ResNet50	86.1	94.4	77.0	87.2	50.2	74.1
SCSN, CVPR 2020 [30]	ResNet50	88.5	95.7	79.0	91.0	58.5	83.8
GASM, ECCV 2020 [31]	ResNet50	84.7	95.3	74.4	88.3	52.5	79.5
AGW, TPAMI 2020 [1, 21]	ResNet50	88.2	95.3	79.9	89.0	55.6	78.3
FastReID, arXiv 2020 [21]	ResNet50	88.2	95.4	79.8	89.6	59.9	83.3
1.1 Baseline†	ResNet50	88.1	95.0	78.9	89.4	61.7	81.5
1.2 Baseline‡		88.6	95.0	79.5	89.4	62.9	82.1
2.1 FlipReID (without flipping loss)†	ResNet50	86.2	94.7	77.2	88.9	57.1	79.5
2.2 FlipReID (without flipping loss)‡		88.5	95.5	79.8	90.2	64.3	83.6
3.1 FlipReID (with flipping loss)†	ResNet50	87.6	95.2	78.9	89.1	61.4	81.9
3.2 FlipReID (with flipping loss)‡		88.5	95.3	79.8	89.4	64.3	83.3
3.3 FlipReID (with flipping loss)‡§		94.6	96.0	90.9	92.5	79.5	86.3
1.1 Baseline†	IBN-ResNet50	88.4	94.8	79.0	88.7	64.6	83.4
1.2 Baseline‡		88.9	95.5	79.6	88.8	65.7	84.2
2.1 FlipReID (without flipping loss)†	IBN-ResNet50	86.9	94.3	77.5	88.7	60.4	81.3
2.2 FlipReID (without flipping loss)‡		88.7	94.8	79.7	89.4	66.2	84.4
3.1 FlipReID (with flipping loss)†	IBN-ResNet50	87.9	94.7	78.8	88.9	63.2	83.1
3.2 FlipReID (with flipping loss)‡		88.6	95.0	79.8	89.6	65.9	84.5
3.3 FlipReID (with flipping loss)‡§		94.1	95.4	89.8	91.8	80.2	87.3
1.1 Baseline†	ResNeSt50	89.3	95.8	80.0	89.6	66.0	84.2
1.2 Baseline‡		89.7	96.2	80.5	89.9	66.9	84.5
2.1 FlipReID (without flipping loss)†	ResNeSt50	88.6	95.2	79.9	90.0	64.6	83.9
2.2 FlipReID (without flipping loss)‡		89.6	95.7	81.2	90.7	67.6	85.3
3.1 FlipReID (with flipping loss)†	ResNeSt50	88.9	95.2	80.7	90.5	66.0	84.6
3.2 FlipReID (with flipping loss)‡		89.6	95.5	81.5	90.9	68.0	85.6
3.3 FlipReID (with flipping loss)‡§		94.7	95.8	90.7	93.0	81.3	87.5

3.3. Analysis of results

Table 1 compares the mAP scores and the rank-1 accuracies of existing studies and our methods.

Datasets. Limited by the number and quality of samples, the Market-1501 [2] and DukeMTMC-reID [3] datasets are saturated, and scores reported on these two datasets may not be indicative [22]. For example, the FastReID [21] method surpasses the AGW [1] method by a large margin on MSMT17 [4], while the scores on Market-1501 and DukeMTMC-reID are close. In the remainder of this study, we mainly compare the mAP scores on MSMT17.

Existing Studies versus Baseline. Among methods built on the ResNet50 [19] backbone, it is evident that the baseline method performs better than existing studies on MSMT17. This makes a good starting point since the FlipReID method is an extension of the baseline method.

Single Image versus Double Images. Independent of how the training is performed, utilizing data augmentation in inference always boosts performance. The notable downside of test-time augmentation is the extra computations which may pose a constraint for real-time applications, in which execution speed is crucial.

Inconsistency between Figure 2(a) and 2(d), *i.e.*, entries starting with 1.2. If data augmentation is enabled in inference, choosing the baseline scheme during training leads to suboptimal performance because the classification model is only trained on feature vectors of single image. Such gap can be solved by switching to the FlipReID method, and noticeable improvement can be observed.

Inconsistency between Figure 2(b) and 2(c), *i.e.*, entries starting with 2.1 or 3.1. If data augmentation is disabled in inference, selecting the FlipReID mechanism during training results in inferior performance since the classification model is only trained on feature vectors of double images. Adding the flipping loss is an effective approach to suppress this problem. On the one hand, the resulting method is on par with the baseline method when using single image in inference. On the other hand, the flipping loss does not degrade performance when using double images in inference.

Backbone and Re-Ranking. In addition to ResNet [19], experiments have been conducted with IBN-ResNet [24] and ResNeSt [25]. The latter two backbone models improve performance. Moreover, adding re-ranking [14] as a post-processing step introduces significant improvements.

4. CONCLUSION

In this study, we recognize and investigate the gap between training and inference in person re-identification. Prior works typically use the mean of feature vectors extracted from the original images and their horizontally flipped variants in inference. However, such mean feature vectors are not present when optimizing the model in training. In order to close the gap, we propose to utilize the FlipReID structure with the flipping loss. On the one hand, both the original images and the flipped images are feed into models with the FlipReID structure. On the other hand, incorporating the flipping loss minimizes the mean squared error between feature vectors of corresponding image pairs. Using the proposed method, models work as expected regardless of whether test-time augmentation is enabled or not, and the inconsistency issue is solved. An extension of this study is to design a module that learns to aggregate feature vectors from multiple sources, rather than calculating the mean.

5. REFERENCES

[1] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven C H Hoi, "Deep learning for person re-identification: A survey and outlook," *arXiv preprint arXiv:2001.04193*, 2020.

[2] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian, "Scalable person re-identification: A benchmark," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1116–1124.

[3] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *European Conference on Computer Vision*. Springer, 2016, pp. 17–35.

[4] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian, "Person transfer gan to bridge domain gap for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 79–88.

[5] Liang Zheng, Hengheng Zhang, Shaoyan Sun, Manmohan Chandraker, Yi Yang, and Qi Tian, "Person re-identification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1367–1376.

[6] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang, "Bag of tricks and a strong baseline for deep person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, p. 0.

[7] Rahul Rama Varior, Bing Shuai, Jiwen Lu, Dong Xu, and Gang Wang, "A siamese long short-term memory architecture for human re-identification," in *European conference on computer vision*. Springer, 2016, pp. 135–153.

[8] Yifan Sun, Liang Zheng, Yi Yang, Qi Tian, and Shengjin Wang, "Beyond part models: Person retrieval with refined part pooling (and a strong convolutional baseline)," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 480–496.

[9] Yutian Lin, Liang Zheng, Zhedong Zheng, Yu Wu, Zhilan Hu, Chenggang Yan, and Yi Yang, "Improving person re-identification by attribute and identity learning," *Pattern Recognition*, 2019.

[10] Zhedong Zheng, Liang Zheng, and Yi Yang, "A discriminatively learned cnn embedding for person reidentification," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 1–20, 2017.

[11] Alexander Hermans, Lucas Beyer, and Bastian Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[12] Chunxiao Liu, Chen Change Loy, Shaogang Gong, and Guijin Wang, "Pop: Person re-identification post-rank optimisation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 441–448.

[13] Hanxiao Wang, Shaogang Gong, Xiatian Zhu, and Tao Xiang, "Human-in-the-loop person re-identification," in *European conference on computer vision*. Springer, 2016, pp. 405–422.

[14] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1318–1327.

[15] Connor Shorten and Taghi M Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[16] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[17] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le, "Autoaugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "ImageNet classification with deep convolutional

neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[20] Guanshuo Wang, Yufeng Yuan, Xiong Chen, Jiwei Li, and Xi Zhou, "Learning discriminative features with multiple granularities for person re-identification," in *Proceedings of the 26th ACM international conference on Multimedia*, 2018, pp. 274–282.

[21] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei, "Fastreid: A pytorch toolbox for general instance re-identification," *arXiv preprint arXiv:2006.02631*, vol. 6, no. 7, pp. 8, 2020.

[22] Xingyang Ni, Liang Fang, and Heikki Huttunen, "Adaptive L2 Regularization in Person Re-Identification," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 9601–9607.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*. Ieee, 2009, pp. 248–255.

[24] Xingang Pan, Ping Luo, Jianping Shi, and Xiaoou Tang, "Two at once: Enhancing learning and generalization capacities via ibn-net," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 464–479.

[25] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R Manmatha, and others, "Resnest: Split-attention networks," *arXiv preprint arXiv:2004.08955*, 2020.

[26] Filip Radenovic, Giorgos Tolias, and Ondrej Chum, "Fine-tuning CNN image retrieval with no human annotation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 7, pp. 1655–1668, 2018.

[27] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[28] Yunpeng Gong and Zhiyong Zeng, "An Effective Data Augmentation for Person Re-identification," *arXiv preprint arXiv:2101.08533*, 2021.

[29] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang, "Random erasing data augmentation," *arXiv preprint arXiv:1708.04896*, 2017.

[30] Xuesong Chen, Canmiao Fu, Yong Zhao, Feng Zheng, Jingkuan Song, Rongrong Ji, and Yi Yang, "Salience-Guided Cascaded Suppression Network for Person Re-Identification," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3300–3310.

[31] Lingxiao He and Wu Liu, "Guided Saliency Feature Learning for Person Re-identification in Crowded Scenes," in *European Conference on Computer Vision*. Springer, 2020, pp. 357–373.

PUBLICATION III

On the Importance of Encrypting Deep Features

X. Ni, H. Huttunen and E. Rahtu

2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW), 2021, 4125–4132

DOI: 10.1109/ICCVW54120.2021.00460

On the Importance of Encrypting Deep Features

Xingyang Ni
Tampere University
Tampere, Finland
xingyang.ni@tuni.fi

Heikki Huttunen
Visy Oy
Tampere, Finland
heikki.huttunen@visy.fi

Esa Rahtu
Tampere University
Tampere, Finland
esa.rahtu@tuni.fi

Abstract

*In this study, we analyze model inversion attacks with only two assumptions: feature vectors of user data are known, and a black-box API for inference is provided. On the one hand, limitations of existing studies are addressed by opting for a more practical setting. Experiments have been conducted on state-of-the-art models in person re-identification, and two attack scenarios (*i.e.*, recognizing auxiliary attributes and reconstructing user data) are investigated. Results show that an adversary could successfully infer sensitive information even under severe constraints. On the other hand, it is advisable to encrypt feature vectors, especially for a machine learning model in production. As an alternative to traditional encryption methods such as AES, a simple yet effective method termed ShuffleBits is presented. More specifically, the binary sequence of each floating-point number gets shuffled. Deployed using the one-time pad scheme, it serves as a plug-and-play module that is applicable to any neural network, and the resulting model directly outputs deep features in encrypted form. Source code is publicly available at* `https://github.com/nixingyang/ShuffleBits`.

1. Introduction

Due to the availability of large-scale datasets [47, 34, 39] and affordable computing resources, the field of machine learning has witnessed rapid progress over the past decade. Real-world applications can be found in everyday life, *e.g.*, targeted advertising in online shopping, recommender systems in video streaming services, and virtual assistants on smart devices. With the widespread adoption of techniques such as person re-identification [48, 24, 26, 42, 14], the concern over security issues can not be overemphasized. Significant efforts have been put into understanding the vulnerabilities in machine learning models [23, 35, 37, 40]. In the following, we outline four types of attacks, *i.e.*, adversarial example attacks, membership inference attacks, model extraction attacks, and model inversion attacks.

In adversarial example attacks, input data is slightly manipulated so that a human may not observe the changes while the model would make incorrect predictions [23]. In [7], a momentum term is integrated into the iterative process for performing attacks, and it stabilizes the direction for updates, avoids poor local maxima, and improves the success rate. Afterward, Su *et al*. [36] analyses an extreme case where only one pixel can be modified. Perturbation is encoded into an array, and the candidate solution is optimized by adopting differential evolution. By contrast, He *et al*. [15] generates an ensemble of weak defenses, while the resulting method does not always promote resilience.

In membership inference attacks, an adversary is interested in identifying whether a specific sample is included in a model's training set [35]. Multiple shadow models are trained to simulate the target model while the membership in their training sets is available [35, 25]. Subsequently, a separate threat model is trained on the input-output pairs of the shadow models, and it behaves differently depending on whether the sample is used for training the target model. In [43], the relation between overfitting and membership vulnerability has been studied, and results indicate that overfitting is a sufficient but not necessary condition for membership vulnerability.

In model extraction attacks, an adversary has black-box access to a target model, and the primary purpose is to duplicate the functionality of the target model [37]. Experiments on simple target models show that one could train substitute models locally on public datasets with near-perfect fidelity [37]. Under similar settings, a reinforcement learning approach is proposed in [31] to improve sample efficiency of queries, and a real-world image recognition model was pirated with reasonable performance. Juuti *et al*. [20] design a countermeasure that analyses the distribution of consecutive query requests and raises the alarm when suspicious activities are detected. Later on, two defense strategies are presented in [22]: the first membership inference strategy checks whether inputs are outliers, and the second watermarking strategy generates wrong outputs deliberately for a tiny fraction of queries.

In model inversion attacks, an adversary intends to infer input data from a released model [40]. Fredrikson *et al*. [11] managed to invert a linear regression model and predict the patient's genetic markers based on demographic information. With confidence scores returned by a facial recognition model, one could recover face images that are representative of a specific person in the training set [10]. In the case that a partial prediction vector is returned, truncation is applied to feature vectors when training the inversion model in [41]. By contrast, Zhang *et al*. [45] shifts the focus to a white-box setting and theoretically proves that the vulnerability to model inversion attacks is unavoidable for models with high predictive power.

Existing studies on model inversion attacks are subject to the following limitations: (1) The threat model is trained on the same dataset as the proprietary model [8, 9, 27, 30]; (2) The adversary has white-box access to the proprietary model [44, 45]; (3) Experiments are limited to small-scale low-resolution datasets [10, 30, 41, 45]. To handle these problems, we investigate model inversion attacks in a more practical setting: (1) The proprietary dataset is unavailable, and the adversary has to collect and utilize a different local dataset; (2) A black-box API for inference is provided, while the architecture and parameters of the proprietary model are unknown; (3) Experiments are conducted on large-scale high-resolution datasets with state-of-the-art proprietary models.

In this study, our contribution is twofold:

- We adopt an experimental setting that is more practical than previous works. Only two assumptions are made, and they hold true in most, if not all, image retrieval systems. On the one hand, an adversary has illegitimate access to feature vectors of user data. On the other hand, the adversary can extract feature vectors of samples in a local dataset via a black-box API. Furthermore, two attack scenarios have been validated on state-of-the-art person re-identification models. In the presence of severe constraints, results indicate that it is still feasible to recognize auxiliary attributes with decent accuracy and reconstruct user data that are recognizable.

- In light of the aforementioned results, we suggest that practitioners incorporate an encryption method when transferring and storing deep features. Note that Advanced Encryption Standard (AES) [5] is the de facto standard for symmetric-key algorithms. We introduce an alternative method termed ShuffleBits, in which the binary sequence of each floating-point number gets shuffled. Unlike traditional encryption methods, it can be implemented as a plug-and-play module inside neural networks, and the resulting model generates encrypted deep features in a straightforward manner.

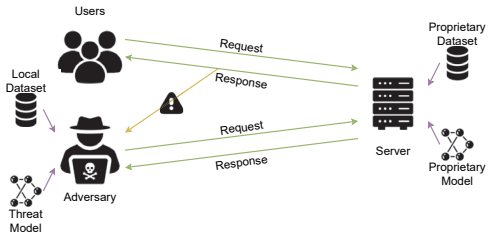


Figure 1. Both the users and the adversary have access to the server, in which a proprietary model is deployed. Meanwhile, the adversary intercepts the server's responses to the users, and a local dataset is available.

2. Attack scenarios

Preliminaries. Figure 1 illustrates the background of attack scenarios in this study. A server runs a proprietary model that is trained on a proprietary dataset, and a response containing feature vectors is returned after processing a request containing user data. Additionally, the server's responses to the users are intercepted by the adversary, *i.e*., feature vectors of user data are known. Since the proprietary dataset is unreachable, the adversary collects and utilizes a local dataset instead. The primary purpose is to train a threat model that sniffs sensitive information of user data from the feature vectors.

Recognizing auxiliary attributes. Depending on the proprietary model in question, certain auxiliary attributes may be relevant. For example, one might be interested in a person's age and gender when using a facial recognition model. Although the original task (*i.e*., recognizing faces) is inherently different from the auxiliary task (*i.e*., predicting age and gender), the feature vectors for the original task may still contain relevant information for solving the auxiliary task. With a local dataset at hand, the adversary could annotate auxiliary attributes and construct a predictive model. The multi-layer perceptron is suitable for solving multi-class classification problems, where each sample might be associated with multiple labels.

Reconstructing user data. Alternatively, one could interpret the whole system as an autoencoder. The proprietary model on the server is the encoder that maps raw data into feature vectors. The adversary builds a decoder that reconstructs raw data from feature vectors. The decoder is trained in an unsupervised manner, *i.e*., it does not require a labeled dataset. The inputs are feature vectors extracted by the proprietary model, and the ground truth outputs are raw data. The decoder is optimized with an objective function so that the difference between ground truth data and reconstructed data is minimized.

Constraints. Multiple constraints complicate matters for the adversary. Firstly, the proprietary model and the threat model are trained on different datasets. Since samples from different datasets vary in factors such as background, weather condition and camera angle, the domain gap would degrade performance. Secondly, the proprietary model's internal workings are out of reach because the adversary can only access it through a black-box API. Outputs of intermediate layers in the proprietary model are unattainable, and methods such as lateral shortcut connections [38] can not be applied. Thirdly, the threat model can not be optimized simultaneously with the proprietary model since the proprietary model is fixed. It leads to a mismatch between the objectives of the proprietary model and the threat model, *e.g.*, the proprietary model learns representative features for facial recognition while the threat model is trained to reconstruct face images.

3. ShuffleBits

In spite of recent studies on binarized neural networks [33, 4] which reduce memory consumption and improve inference speed, storing weights and activations in the single-precision floating-point format is still the predominant option. Each single-precision floating-point value can be viewed as a 32-bit binary sequence (*i.e.*, binary32). The IEEE 754 standard [1] defines the procedure which converts a real number from decimal representation to binary32 format, and vice versa.

Given a single-precision floating-point value x, it can be represented as a finite binary sequence

$$(a_i)_{i\in I}, \tag{1}$$

where $a_i \in \{0, 1\}$, $I = \{1, \ldots, n\}$ and $n = 32$.

One could shuffle the original sequence according to an encryption key, and the encrypted sequence is

$$(b_j)_{j\in J}, \tag{2}$$

where $J = \{1, \ldots, n\}$. The encryption key is a bijective function $f\colon I \to J$, and it is an injective and surjective mapping of set I to set J. In addition, we have $b_{f(i)} = a_i$ for $i \in I$.

Similarly, the decrypted sequence is

$$(c_k)_{k\in K}, \tag{3}$$

where $K = \{1, \ldots, n\}$. The decryption key is another bijective function $g\colon J \to K$ which maps set J to set K. Furthermore, we have $c_{g(j)} = b_j$ for $j \in J$.

Since f is a bijection, it has an inverse function obtained by swapping the inputs and outputs in f. Let g be the inverse function of f, we have

$$a_i = b_{f(i)} = c_{g(f(i))} = c_i \text{ for } i \in I. \tag{4}$$

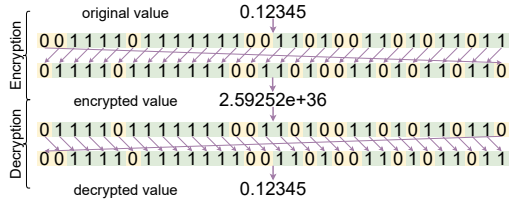


Figure 2. A specific case of ShuffleBits: a left rotation operation is applied in the encryption process, and a right rotation operation is applied in the decryption process.

With the correct decryption key, it is apparent that the decrypted sequence is identical to the original sequence. Finally, the encrypted sequence and the decrypted sequence can be converted to decimal representation.

Figure 2 provides a step-by-step explanation of the proposed method using specific encryption and decryption keys. The original value's binary sequence is shuffled according to the encryption key, and the encrypted sequence corresponds to the encrypted value. By contrast, modifications are reverted so that the decrypted value is the same as the original value.

In the event of a brute-force attack, the adversary must systematically enumerate all possible decryption keys and check each of them. It is computationally infeasible to conduct exhaustive key search for three reasons. Firstly, there are $32! \approx 2.63\text{e}{+35}$ unique keys, thus the number of candidate keys is large. Secondly, it is not straightforward to validate whether the decrypted values are correct or not. Thirdly, bit shuffling can be deployed using the one-time pad scheme, and the decryption keys in each request would be different.

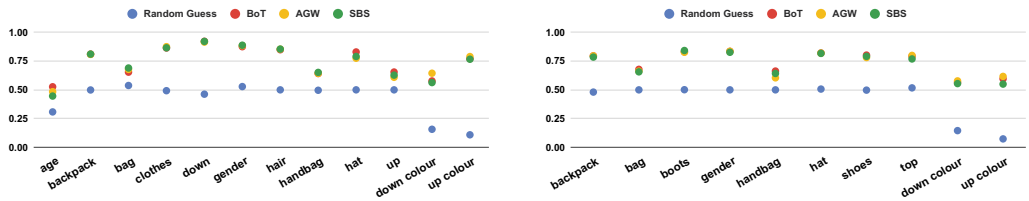
4. Experiments

4.1. Background

Domain. We conduct experiments in the domain of person re-identification, in which the objective is to retrieve a person of interest across multiple cameras [42].

Datasets. We select the following datasets that are widely used: Market-1501 [47], DukeMTMC-reID [34] and MSMT17 [39]. In each dataset, there are three partitions, namely, training set, query set, and gallery set. The latter two sets are merged as the test set. Throughout this study, we use MSMT17 as the proprietary dataset, while the local dataset is either Market-1501 or DukeMTMC-reID.

Models. The FastReID repository provides a unified instance re-identification library, along with a set of pre-trained models [14]. We include three top-performing methods which are built using the ResNet50 [13] backbone, *i.e.*, BoT [26], AGW [42] and SBS [14].



(a) Scores on the test set in Market-1501.

(b) Scores on the test set in DukeMTMC-reID.

Figure 3. The balanced accuracies of each auxiliary attribute.

4.2. Recognizing auxiliary attributes

Model. For each auxiliary attribute, batch normalization [17] and dense layers are stacked to obtain the probabilities of each class. Similar to the proprietary models [26, 42, 14] that classify person identities, we use only one batch normalization layer and one dense layer. The dimensionality of the output space in the dense layer equals the number of classes. Opting for this relatively simple architecture gives the best results in our experiments.

Loss function. Similar to conventional classification models [13], the cross-entropy loss [46] is utilized on the outputs of dense layers. Given an imbalanced dataset with unequal distribution of classes, classifiers would be biased in favor of the dominant classes. To address this issue, we assign a scalar value to each class during training so that more attention is paid to the under-represented classes [32]. The class weights are inversely proportional to the count number of occurrences of each class.

Evaluation metric. The accuracy score measures the percentage of samples in which the predicted label matches the corresponding ground truth. However, it may give inflated performance estimates on imbalanced datasets. Thus, we adopt balanced accuracy [29] which is a better option, and it is defined as the average of recall calculated on each class.

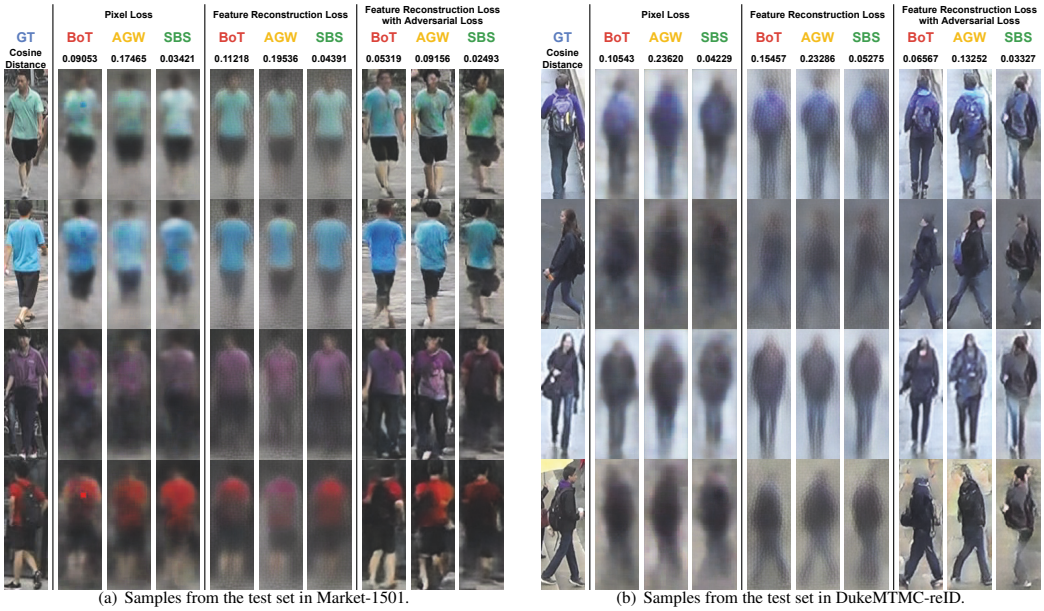
Implementation. The batch size is set to 128, and the number of epochs is limited to 100. The Adam [21] optimizer is utilized in training the model. The learning rate is fixed to $5\mathrm{e}{-5}$ in the first 50 epochs, and it decreases by a factor of five in the remaining epochs. The mean of balanced accuracies of all labels is monitored so that the optimal model can be identified.

Analysis. We leverage the auxiliary attributes in [24]. These annotations provide detailed descriptions of pedestrians. Multiple labels are present while each label corresponds to a binary or multi-class classification problem. Figure 3 visualizes the balanced accuracies of each auxiliary attribute in two local datasets. Using feature vectors extracted by proprietary models yields significantly more accurate predictions than guessing randomly, and it gives coarse-grained estimations of user data.

4.3. Reconstructing user data

Model. Two sub-models are involved in reconstructing user data, *i.e.*, a generator and a discriminator. We instantiate models with similar architectures to the BigGAN [2] work. On the one hand, the generator maps feature vectors to images. At the start, a dense layer with 256 units transforms the feature vectors into a low-dimensional space, while another dense layer and a reshaping operation generate the smallest feature maps. Subsequently, five upsampling residual blocks increase the spatial dimensionality to the target resolution (*i.e.*, $128{\times}384$). The last convolutional layer reduces the number of channels to 3, and the resulting predictions are in the RGB color space. On the other hand, the discriminator classifies whether the images are original or synthetic. Five downsampling residual blocks decrease the spatial dimensionality, and a global average pooling layer generates flattened feature vectors of the images. Complemented with the feature vectors extracted by the proprietary model, a dense layer generates the estimations based on the concatenated feature vectors. More details regarding the architectural layout of residual blocks can be found in the appendix of [2].

Loss function. Different loss functions are utilized for updating the generator and discriminator. The generator can be optimized with a weighted sum of the following loss functions: (1) The pixel loss [19] calculates the mean squared error between the ground truth images and the reconstructed images; (2) Given a pre-trained model, one may extract an intermediate layer's outputs as feature maps. The feature reconstruction loss [19] refers to the mean squared error between the feature maps of the ground truth images and the reconstructed images. More specifically, we use the outputs of layer "conv2_block3_out" in a ResNet50 [13] model that is pre-trained on the ImageNet [6] dataset; (3) The adversarial loss [12] measures how well the generator can fool the discriminator when feeding the outputs of the generator to the discriminator. By contrast, the discriminator is optimized using the mean squared error loss that is proposed in [28]. Compared with the cross-entropy loss [46], it suppresses the vanishing gradients problem and stabilizes the learning process.



(a) Samples from the test set in Market-1501. (b) Samples from the test set in DukeMTMC-reID.

Figure 4. Comparison of the reconstructed images using different proprietary models, local datasets, and loss functions.

Evaluation metric. For each reconstructed image, the ground truth image is available for reference. Instead of comparing these images pixel by pixel, we extract the reconstructed images' feature vectors using the same proprietary model and calculate the cosine distance between feature vector pairs. It follows the same principle as the feature reconstruction loss [19], while there exist differences in the underlying model and the distance metric. The cosine distance metric is widely used when comparing two feature vectors extracted by person re-identification models. The cosine distance scores range from 0 to 1, and the minimum value 0 is obtained when the angle between feature vectors is 0. The aforementioned evaluation metric provides quantitative performance measures for studying the effects of loss functions.

Implementation. The batch size is set to 64, and the number of iterations is limited to $60,000$. The Adam [21] optimizer is utilized in training both the generator and discriminator, with the learning rate fixed to $1\mathrm{e}{-4}$. The training procedure within one batch can be separated into multiple consecutive steps. Firstly, the generator produces the reconstructed images based on the pre-computed feature vectors. Secondly, the discriminator is updated by feeding the ground truth or reconstructed images alongside the corresponding labels. Thirdly, the generator is updated while keeping the parameters of the discriminator frozen.

Analysis. Generating perfect reconstructions merely from feature vectors is a difficult task. Otherwise, the algorithm can be treated as a promising solution for neural image compression. As discussed in Section 2, there are three constraints that pose significant challenges, namely, the domain gap in datasets, the nature of having only black-box access, and the mismatch between optimization objectives. Figure 4 illustrates randomly chosen reconstructed images under various settings. Those experiments differ in terms of the proprietary model, the local dataset, and the loss function. In addition, the mean of cosine distance scores is calculated on the corresponding test set. Three observations can be drawn:

- Using the pixel loss gives inherently blurry predictions for the reason that the Euclidean distance is minimized by averaging all plausible outputs [18].
- Switching to the feature reconstruction loss sharpens the images, while noticeable checkerboard artifacts are present.
- The checkerboard artifacts can be suppressed significantly by adding the adversarial loss, and the reconstructed images share strong similarities with the ground truth images. Furthermore, combining the feature reconstruction loss with the adversarial loss results in the lowest cosine distance score.

4.4. Importance of encrypting deep features

Results in Section 4.2 and 4.3 demonstrate that an adversary could successfully infer sensitive information even under severe constraints. In particular, it is still feasible to recognize auxiliary attributes with decent accuracy and reconstruct user data that are recognizable.

Given a machine learning model in production, it is of great importance to adopt an encryption method. Under the condition that an encryption method is utilized, one has to include an encryption key in each request, and feature vectors in the corresponding response are encrypted (see Figure 1). Since the decryption key is kept on the client side, the original values can be recovered without changes. The adversary could still train threat models on original feature vectors. However, the decryption key required to decrypt feature vectors of user data is unknown, while the threat models would not generate meaningful predictions on encrypted feature vectors. In addition, training threat models directly on encrypted feature vectors is not an option because the users and the adversary are using different encryption keys.

4.5. ShuffleBits vs traditional encryption methods

The ShuffleBits method differs from traditional encryption methods in two aspects. On the one hand, computations in ShuffleBits can be translated into operations on tensors. Incorporating ShuffleBits into an existing neural network is straightforward, and it can be implemented as a plug-and-play module without extra dependencies. On the other hand, a model with ShuffleBits would generate encrypted deep features directly. It eliminates the risk of transferring unencrypted data from GPU to CPU.

The Advanced Encryption Standard (AES) [5] method is widely accepted as the de facto standard for symmetric-key algorithms. Since the security of ShuffleBits is yet to be validated, a natural extension would be to develop dedicated attacks against ShuffleBits from the perspective of cryptography. In the work of InstaHide [16], a method is introduced to encrypt training images in a federated learning scenario. However, it is later found to be insecure in [3]. To address the potential vulnerabilities in ShuffleBits, we introduce two workarounds. On the one hand, the one-time pad scheme can be utilized. In each request, different encryption/decryption keys are used. It is unlikely to recover the original feature vectors by observing a few instances. On the other hand, ShuffleBits can be seamlessly applied alongside traditional encryption methods. Such cascade encryption pipeline leads to better security, as an adversary has to break all the encryption algorithms to obtain useful information.

5. Conclusion

This study emphasizes the importance of encrypting deep features when deploying a machine learning model in production. On the one hand, we adopt an experimental setting with only two assumptions, and it is more practical than previous works. Two attack scenarios have been proposed to reverse state-of-the-art person re-identification models. Results show that an adversary could recognize auxiliary attributes and reconstruct user data, thus breaching user privacy. On the other hand, adopting an encryption method when transferring and storing deep features would prevent model inversion attacks. By performing manipulations on the binary sequence of each floating-point number, we introduce the ShuffleBits method, and it can be implemented as a plug-and-play module inside neural networks.

References

[1] IEEE Standard for Floating-Point Arithmetic. *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pages 1–84, 2019.

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[3] Nicholas Carlini, Samuel Deng, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Shuang Song, Abhradeep Thakurta, and Florian Tramer. Is Private Learning Possible with Instance Encoding?, 2021.

[4] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[5] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael. 1999.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[7] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[8] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*, 2016.

[9] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4829–4837, 2016.

[10] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[11] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmaco-

genetics: An end-to-end case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. Fastreid: A pytorch toolbox for general instance re-identification. *arXiv preprint arXiv:2006.02631*, 6(7):8, 2020.

[15] Warren He, James Wei, Xinyun Chen, Nicholas Carlini, and Dawn Song. Adversarial example defense: Ensembles of weak defenses are not strong. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*, 2017.

[16] Yangsibo Huang, Zhao Song, Kai Li, and Sanjeev Arora. Instahide: Instance-hiding schemes for private distributed learning. In *International Conference on Machine Learning*, pages 4507–4518. PMLR, 2020.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[19] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[20] Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. PRADA: protecting against DNN model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE, 2019.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366*, 2019.

[23] Alexey Kurakin, Ian Goodfellow, Samy Bengio, and others. Adversarial examples in the physical world, 2016.

[24] Yutian Lin, Liang Zheng, Zhedong Zheng, Yu Wu, Zhilan Hu, Chenggang Yan, and Yi Yang. Improving person re-identification by attribute and identity learning. *Pattern Recognition*, 2019.

[25] Yunhui Long, Vincent Bindschaedler, and Carl A Gunter. Towards measuring membership privacy. *arXiv preprint arXiv:1712.09136*, 2017.

[26] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[27] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016.

[28] Xudong Mao, Qing Li, Haoran Xie, Raymond Y K Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[29] Lawrence Mosley. A balanced approach to the multi-class imbalance problem. 2013.

[30] Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

[31] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4954–4963, 2019.

[32] Sankaran Panchapagesan, Ming Sun, Aparna Khare, Spyros Matsoukas, Arindam Mandal, Björn Hoffmeister, and Shiv Vitaladevuni. Multi-task learning and weighted cross-entropy for DNN-based keyword spotting. In *Interspeech*, volume 9, pages 760–764, 2016.

[33] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[34] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[35] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18. IEEE, 2017.

[36] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[37] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing Machine Learning Models via Prediction APIs. In *Proceedings of the 25th USENIX Conference on Security Symposium*, pages 601–618, 2016.

[38] Harri Valpola. From neural PCA to deep unsupervised learning. In *Advances in independent component analysis and learning machines*, pages 143–171. Elsevier, 2015.

[39] Longhui Wei, Shiliang Zhang, Wen Gao, and Qi Tian. Person transfer gan to bridge domain gap for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 79–88, 2018.

[40] Xi Wu, Matthew Fredrikson, Somesh Jha, and Jeffrey F Naughton. A methodology for formalizing model-inversion attacks. In *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*, pages 355–370. IEEE, 2016.

[41] Ziqi Yang, Ee-Chien Chang, and Zhenkai Liang. Adversarial neural network inversion via auxiliary knowledge alignment. *arXiv preprint arXiv:1902.08552*, 2019.

[42] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven C H Hoi. Deep learning for person re-identification: A survey and outlook. *arXiv preprint arXiv:2001.04193*, 2020.

[43] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, pages 268–282. IEEE, 2018.

[44] Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8715–8724, 2020.

[45] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 253–261, 2020.

[46] Zhilu Zhang and Mert R Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*, 2018.

[47] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1116–1124, 2015.

[48] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.

PUBLICATION
IV

Vehicle Attribute Recognition by Appearance: Computer Vision Methods for Vehicle Type, Make and Model Classification

X. Ni and H. Huttunen

Journal of Signal Processing Systems 93.4 (2021), 357–368
DOI: 10.1007/s11265-020-01567-6

Journal of Signal Processing Systems (2021) 93:357–368
https://doi.org/10.1007/s11265-020-01567-6

Vehicle Attribute Recognition by Appearance: Computer Vision Methods for Vehicle Type, Make and Model Classification

Xingyang Ni[1] · **Heikki Huttunen**[1]

Received: 12 November 2019 / Revised: 29 March 2020 / Accepted: 8 June 2020 / Published online: 26 June 2020

Abstract
This paper studies vehicle attribute recognition by appearance. In the literature, image-based target recognition has been extensively investigated in many use cases, such as facial recognition, but less so in the field of vehicle attribute recognition. We survey a number of algorithms that identify vehicle properties ranging from coarse-grained level (vehicle type) to fine-grained level (vehicle make and model). Moreover, we discuss two alternative approaches for these tasks, including straightforward classification and a more flexible metric learning method. Furthermore, we design a simulated real-world scenario for vehicle attribute recognition and present an experimental comparison of the two approaches.

Keywords Vehicle attribute recognition · Image classification · Metric learning

1 Introduction

Traffic monitoring is an essential tool for collecting statistics to enable better design and planning of transport infrastructure. Often, plain vehicle counting is not enough, and there is a need to capture extended attributes of the vehicles; for example, can separation of heavy traffic from lighter vehicles, or following individual vehicles to find out which routes the drivers usually take? Such data allows more fine-grained analysis and more accurate profiling of users of transport infrastructure, which is necessary for assessing the effects of future changes in transportation.

The traditional approach for collecting traffic data is to organize manual data collection campaigns and enroll human labor for counting the vehicles, or implement road-side questionnaires after stopping the vehicles. Needless to say, such labor-intensive operations are quite insulting to drivers' experience. Alternatively, various technologies could be used to facilitate the data collection procedure. The inductive ground loops (see, e.g., MAVE®-L product line of AVE GmbH)[1] measure the magnetic profile of vehicles passing by and provide a coarse-grained classification of vehicle type. Moreover, laser scanners (see, e.g., traffic counters from SICK GmbH)[2] can gather similar information from vehicles. Even audio can be applied as a means of identifying the type of a vehicle by extracting the predefined feature set from segments of short audio signal [75].

More recently, camera-based techniques for traffic monitoring have become more widespread. Cameras are ubiquitous, cost-effective, and often have been already utilized in other surveillance use cases. Moreover, they provide a rich source of information with which new generation of recognition methods become feasible; for example, inductive ground loops and laser scanners enable only a very coarse-grained categorization due to the nature of the input data they have. The level of computer vision techniques was the bottleneck of camera-based traffic monitoring systems for a long time. However, the emergence of deep learning has thoroughly changed the situation. In particular, image classification has progressed to an entirely new level within the last ten years and is reaching human-level accuracy in many domains. An essential factor in this transformation is the availability of large-scale datasets. Significant milestones in the history of such datasets include the

✉ Xingyang Ni
xingyang.ni@tuni.fi

Heikki Huttunen
heikki.huttunen@tuni.fi

[1] Tampere University, Tampere, Finland

[1] http://www.ave-web.de/
[2] http://www.sick.de/

ImageNet dataset for image classification [9], the Microsoft COCO dataset for object detection [42], and more recent vehicle-specific datasets such as the KITTI dataset for autonomous driving [18], the VERI-Wild dataset for vehicle re-identification [48], and the CCPD dataset for license plate recognition [79].

Deep learning techniques have enabled diverse practical applications of vehicle attribute recognition. In this paper, we will discuss the following three problem settings:

- **Vehicle type recognition** attempts to characterize vehicles to coarse-grained categories by their size or intended usage, e.g., sedan, bus and truck. A common use case is statistical: What is the distribution of vehicle categories at a specific checkpoint?
- **Vehicle make recognition** categorizes vehicles by their manufacturer, e.g., Ford, Toyota and Chevrolet. An example use case is searching for stolen vehicles or license plates: Did the make of this vehicle change since last month?
- **Vehicle model recognition** learns to predict the vehicle model, e.g., Ford Puma, Toyota Corolla and Chevrolet Volt. This task is significantly more detailed than the aforementioned vehicle make recognition, and it can be used as a method to study consumer behavior regarding a specific vehicle model. Additional challenges are introduced due to the dynamic nature of the data: Manufacturers introduce new designs annually, and the prediction model needs to be regularly updated.

Additional topics outside the scope of this paper include vehicle re-identification [45, 47, 48] which is an exciting and prevalent area that is related to metric learning described below. It targets to match vehicles across multiple cameras based on their appearance. Such methods allow vehicle tracking in larger regions, which helps in planning the road network. Alternatively, one could use license plate recognition for the same task. However, re-identification does not explicitly require the license plate to be visible and allows a less restricted camera placement (e.g., longer distance, or non-frontal camera angle).

The benefit of type, make and model recognition is that they do not collect any privacy-sensitive information about individual vehicles. The results are purely statistical (e.g., "15% of traffic on this road are buses"), and may thus avoid potential privacy issues. On the other hand, license plate recognition and re-identification systems can also be implemented in a privacy-preserving manner, where only a hash of the recognized license plate or a feature vector representing the appearance is retrieved, rather than a image of the vehicle itself. Nevertheless, their use may be still be less acceptable by the public than a purely statistical approach.

There are two commonly used strategies for type, make and model recognition. First is the straightforward approach that poses the recognition task as a classification problem. For example, vehicle type recognition typically has only a few distinct classes with abundant samples of each class. It is natural to present it as a K-way classification problem [27, 36].

The classification approach may not be feasible for all cases. In re-identification, it is difficult to train a classification model since the large number of identities would make the last fully-connected layer excessively huge. Therefore, this problem is usually approached as a metric learning problem, where a neural network learns a mapping function from images to feature vectors. The identity can be retrieved by comparing the feature vectors against the historical collection of feature vectors, most often applying the nearest neighbor search method. Another significant benefit of the metric learning approach is that it allows addition of new data (e.g., new car models when they are launched) without retraining the entire model. With this in mind, we will describe both approaches, and carry out experimental analyses of their performance.

The rest of this paper is organized as follows. In Section 2, we discuss available datasets and different problem settings of vehicle attribute recognition. Section 3 explains the methodology of our proposed solution to vehicle type and make recognition, and reports the experimental results on the VERI-Wild [48] dataset. Section 4 describes the future research directions. Finally, Section 5 contains the concluding remark.

2 Vehicle Attribute Recognition

In most problems in computer vision, methods can be roughly divided into two categories: older *Hand-crafted feature engineering* methods and newer *Deep learning* approaches. We will discuss these next.

Hand-crafted methods rely on human-engineered feature extraction pipelines to transform the image into a set of features that are robust to variations in both vehicle-specific variables (e.g., scale, location and color), as well as environment variables (e.g., pose, illumination and background). The feature extraction stage is followed by a conventional machine learning (*i.e.*, non-deep learning) classifier, such as nearest neighbor search or Support Vector Machine (SVM) [6].

Deep learning methods differ from the hand-crafted methods in that they do not require human-engineered feature extractors, but instead learn the feature extractors purely from data. This is the main reason for clearly superior accuracy compared to traditional approaches. For a more

in-depth discussion of deep learning, we refer the reader to [15, 19, 26].

2.1 Datasets

Image-based vehicle attribute recognition has been studied extensively, especially since the introduction of deep learning techniques. Table 1 lists commonly used datasets on vehicle attribute recognition, sorted by publication year. In addition to the dataset size, the number of unique types/makes/models is shown when applicable.

Historically, the earliest dataset was collected by Petrovic and Cootes [55], and it concentrated on vehicle model recognition. The dataset contains 1,132 frontal images from 77 distinct vehicle models. Likewise, Clady et al. [5] introduced a set of frontal vehicle images with annotations of models. More recently, the car-types [67] dataset has doubled the number of images among the early datasets, with images captured from different viewpoints.

Later, Peng et al. [54] built up a collection of high-resolution frontal images taken under daylight and night-light, with annotated labels of vehicle type. Both BMW-10 and car-197 datasets were proposed in [39], with the former containing an ultra-fine-grained dataset of 10 BMW sedans and the latter comprising significantly more images from 197 models. Interestingly, the FG3DCar [44] dataset is annotated with additional 64 landmark locations, which makes it feasible to apply 3D-based methods.

In 2015 and 2016, vehicle attribute recognition has seen a surge of interest as many new datasets [13, 27, 41, 64, 80] have been gathered. The most notable datasets are CompCars [80] and BoxCars [64], which contain a large number of images with fine-grained labels of specific vehicle models. The CompCars dataset consists of images collected from either public websites or surveillance cameras, enabling real-world applications that need to address the challenge of significant appearance variations. On the other hand, the BoxCars dataset focuses on traffic surveillance applications and includes a 3D bounding box for each vehicle and the foreground mask.

As might be expected, the dataset scale is continuously increasing. The updated BoxCars116k [65] dataset extends the previous BoxCars [64] to contain almost twice the number of images. The MIO-TCD [51] dataset is a significant milestone pushing vehicle attribute recognition to the next level. The dataset is many times larger than any previous datasets, and it features a localization subset for the detection task and a classification subset for the recognition task.

Among the most recent datasets, the VERI-Wild [48] dataset was initially collected for vehicle re-identification, and images are captured under unconstrained scenarios. Nevertheless, the provided vehicle type and make annotations make it also suitable for recognition purposes.

It stands out to argue that deep neural networks benefit considerably from large-scale datasets. For future research, the MIO-TCD [51] dataset, the VERI-Wild [48] dataset and the CompCars [80] dataset are well suited to vehicle type recognition, vehicle make recognition and vehicle model recognition, respectively.

2.2 Methods for Vehicle Type Recognition

Vehicle type recognition aims at a coarse-grained prediction of vehicle type, with popular categories including sedan, bus and truck. Table 2 summarizes conspicuous hand-crafted

Table 1 Commonly used datasets on vehicle attribute recognition, sorted by publication year.

Dataset	Year	#Image	#Type	#Make	#Model
Petrovic and Cootes [55]	2004	1,132	–	–	77
Clady et al. [5]	2008	1,121	–	–	50
car-types [67]	2011	1,904	–	–	14
Peng et al. [54]	2012	4,924	5	–	–
BMW-10 [39]	2013	512	1	1	10
car-197 [39]	2013	16,185	7	–	197
FG3DCar [44]	2014	300	–	–	30
Liao et al. [41]	2015	1,482	–	8	–
BIT-Vehicle [13]	2015	9,850	6	–	–
CompCars [80]	2015	214,345	12	161	1,687
Huttunen et al. [27]	2016	6,555	4	–	–
BoxCars [64]	2016	63,750	–	27	148
BoxCars116k [65]	2018	116,286	–	45	693
MIO-TCD [51]	2018	648,959	11	–	–
VERI-Wild [48]	2019	416,314	14	149	–

Table 2 Selected hand-crafted and deep learning methods for vehicle type recognition. Algorithms tested on the same dataset are grouped.

Method	Year	Dataset	#Image	#Type	Accuracy	Notes
Petrovic and Cootes [54, 55]	2004	Peng et al. [54]	4,924	5	84.3%	Hand-crafted
Psyllos et al. [54, 57]	2011				78.3%	Hand-crafted
Peng et al. [54]	2012				90.0%	Hand-crafted
Dong and Jia [12]	2013				91.3%	Hand-crafted
Peng et al. [53]	2013				93.7%	Hand-crafted
Dong et al. [13]	2015				96.1%	Deep learning
Kafai and Bhanu [33]	2012	Kafai and Bhanu [33]	845	4	96.6%	Hand-crafted
Petrovic and Cootes [2, 55]	2004	BIT-Vehicle [13]	9,850	6	78.6%	Hand-crafted
Psyllos et al. [2, 57]	2011				70.8%	Hand-crafted
Peng et al. [2, 53]	2013				85.0%	Hand-crafted
Dong et al. [13]	2015				88.1%	Deep learning
Sun et al. [68]	2017				90.1%	Hand-crafted
Yang et al. [80]	2015	Subset of CompCars [80]	52,083	12	63.1%	Deep learning
Huttunen et al. [27]	2016	Huttunen et al. [27]	6,555	4	97.8%	Deep learning
He et al. [22, 58]	2015	MIO-TCD [51]	648,959	11	96.5%	Deep learning
Kim and Lim [37]	2017				97.8%	Deep learning
Lee and Chung [71]	2017				97.9%	Deep learning
Jung et al. [31]	2017				97.9%	Deep learning
Theagarajan et al. [72]	2017				97.8%	Deep learning
Rachmadi et al. [58]	2018				97.9%	Deep learning

and deep learning methods for vehicle type recognition. Note that the accuracies of different methods are not comparable if they are not tested on the same dataset.

2.2.1 Hand-Crafted Methods

Psyllos et al. [57] defined the Region of Interest based on the size and location of the license plate after recognizing the plate with the Sliding Concentric Window segmentation method [8]. A Probabilistic Neural Network [66] is trained on a set of Scale Invariant Feature Transform (SIFT) [49] feature descriptors, and it accelerates inference speed considerably compared with the conventional nearest neighbor classifier.

Peng et al. have done a series of works in vehicle type recognition. In [54], a coarse-to-fine method is proposed to enable fast and accurate license plate localization. A coarse-grained detection is obtained by inspecting the intensity histograms horizontally, while the line segments feature generates finer localization. Eigenvectors are extracted from vehicle front images as the feature representation. K-mean clustering is applied to each vehicle type, and the category of a test sample is in line with the nearest cluster center. Later on, the aforementioned method gets improved in three aspects [53]. First, the color of the license plate is an informative clue to vehicle type, and it is incorporated into the classification pipeline. Furthermore, the coordinates of the vehicle are extracted accurately with a straightforward background-subtraction method. Lastly, the algorithm estimates the vehicle type based on the top ten most similar training samples rather than only the best match.

In [12], two sets of feature embedding are utilized. On the one hand, the SIFT [49] feature analyzes local patterns in images, and it falls into the category of appearance-based features. On the other hand, the relative coordinate between each SIFT keypoint and the mean keypoint of a local region correspond to the structural feature. A Multiple Kernel Learning method is proposed to merge the two feature sets mentioned above and generate a more robust prediction.

Comparably, Sun et al. [68] extracted two sets of feature embedding individually. The global feature set is produced by an improved Canny edge detection algorithm, while the local feature set is extracted from by applying Gabor wavelet kernels on non-overlapping patches of the whole vehicle image. In addition, a two-stage classification strategy is proposed: the first stage model predicts whether the sample is a small vehicle or a large vehicle while the second stage model recognizes the specific vehicle type.

Kafai and Bhanu [33] shifted focus on video-based vehicle type recognition from direct rear-side view for the reason that a vehicle does not necessarily have a front license plate. After detecting a moving vehicle with a Gaussian mixture model, the coordinates of the license

plate are extracted by using either a matched filtering approach [1] which exploits the colored texture in the plate, or a blob detection and filtering method which picks out the best match from candidate blobs. In addition to the license plate, the regions of tail light are located by examining the redness of pixels. A low-level feature set is computed from each frame, e.g., height, width, and angle of tail lights. Subsequently, a Hybrid Dynamic Bayesian Network is implemented by adding an explicit temporal dimension to a standard Bayesian Network, and it generates a probability distribution with respect to the feature vectors.

2.2.2 Deep Learning Methods

Dong et al. [13] adopts a semi-supervised convolutional neural network to classify vehicle type. The neural network consists of convolutional layers, absolute value rectification layers, local contrast normalization layers, average pooling layers, subsampling layers, and a fully-connected layer with softmax activation. A sparse Laplacian filter learning method is proposed to optimize the parameters of convolutional layers with a large number of unlabeled samples. On the contrary, the parameters in the fully-connected layer are learned on labeled samples.

Huttunen et al. [27] compared the performances of a deep learning neural network with a hand-crafted method which employs SVM [6] on SIFT [49] feature. Instead of using either manual or grid search of the optimal setting of neural network's hyperparameters such as input image size, kernel size of convolutional layers and learning rate, the random search strategy [3] reduces computational burden significantly while reaching comparable or even superior performance. The resulting topology outperforms the SVM classifier in terms of accuracy.

Since the introduction of the large-scale MIO-TCD [51] dataset, deep learning has become the predominant approach for vehicle type recognition. Kim and Lim [37] choose a convolutional neural network of moderate size, and the samples are augmented with flipping and rotations. Multiple models are trained while each of them accesses a half portion of the training set, which is randomly selected. Consequently, the classification system produces several predictions for a single test sample. The results are aggregated by a voting process that imposes different weights to each class label so that the problem of imbalanced data is compensated.

Lee and Chung [71] propose an ensemble of 12 local expert networks and 6 global networks. The local expert networks take the GoogLeNet [69] structure, and each network is trained on a subset of training samples. The dataset is split in view of the resolution and aspect ratio of samples. Conversely, the global networks are trained with all training samples and three topologies are used, namely, AlexNet [40], GoogLeNet [69] and ResNet [23]. In the inference procedure, a rule-based gating function [29] selects the prediction from a specific local expert network considering the resolution and aspect ratio of the test sample. The final prediction is generated by merging the predictions of single local expert network and multiple global networks.

Jung et al. [31] train ResNet [23] models on samples augumented by photometric distortions [24] and color modifications [40]. Multiple ResNet-based backbones are optimized simultaneously while their outputs are element-wise added up. A joint fine-tuning method [32] is employed to fine-tune all parameters rather than only the last dense layer. Besides, a mechanism named DropCNN randomly drops the predictions from the aforementioned backbones during training.

In [72], two neural networks are trained independently with the weighted cross-entropy loss function. Both models are based on ResNet [23], and they differ in the number of layers. The logical reasoning is appended to the fully-connected layer to confront the issue of dual-class misclassification. The predictions of different models are combined using weights, which refer to the average values of precision and recall.

Last but not least, Rachmadi et al. [58] introduces a Pseudo Long Short-Term Memory (P-LSTM) classifier for identifying a single image. Unlike the ordinary use cases which involve time-series data, multiple parallel networks extract features from different crops of the input image, and those spatial pyramid features are feed to the P-LSTM classifier in sequence. A fully-connected layer is appended at the end to compute the probabilities of each class label.

2.3 Methods for Vehicle Make and Model Recognition

Vehicle make recognition targets to predict the manufacturer or brand of the vehicle (e.g., Ford, Toyota or Chevrolet). By contrast, vehicle model recognition aims at a more fine-grained prediction of the particular model (e.g., Ford Puma, Toyota Corolla or Chevrolet Volt). The characteristics of type recognition differ from make and model recognition in two aspects. The number of categories in type recognition is significantly smaller than that in make and model recognition. Besides, type recognition tends to have static categories, whereas makes and models change at times. Consequently, type recognition is commonly considered as a more manageable task where a straightforward classification setup and even hand-crafted classifiers can be competitive.

Tables 3 and 4 compile prominent hand-crafted and deep learning methods for vehicle make recognition and vehicle model recognition, respectively. Note that the accuracies of

Table 3 Selected hand-crafted and deep learning methods for vehicle make recognition. Algorithms tested on the same dataset are grouped.

Method	Year	Dataset	#Image	#Make	Accuracy	Notes
Liao et al. [41]	2015	Liao et al. [41]	1,482	8	81.3%	Hand-crafted
Yang et al. [80]	2015	Subset of CompCars [80]	30,955	75	82.9%	Deep learning
Hu et al. [25]	2017				99.3%	Deep learning
Xiang et al. [76]	2019				99.6%	Deep learning

different methods are not comparable if they are not tested on the same dataset.

2.3.1 Hand-Crafted Methods

In the early works, methods are typically constrained to vehicle images captured from the frontal view, and the nearest neighbor search is employed to find the most similar sample. In [55], the license plate is detected by searching for all possible right angle corners and selecting the best candidate while considering the scale and aspect constraints of its rectangle structure. A set of structure mapping methods are investigated to extract the feature vector from the Region of Interest, e.g., raw pixel values, Harris corner detector [21], and square mapped gradients. Clady et al. [5] propose to construct a model from several frontal vehicle images based on oriented-contour points. Given a prototype image, the oriented-contour points matrix is computed by applying a histogram-based threshold process on the gradient orientations. A discriminant function measures the similarity scores between test samples and labels in trained models.

Later methods take advantage of the Deformable Part Model (DPM) [14] algorithm, which is originally proposed to solve generic object detection tasks. Stark et al. [67] suggest that the detected parts indicate the geometry of objects and help in matching the class labels. The vanilla DPM is reformulated as a latent linear multi-class SVM [6], and the consequential structDPM classifier is directly optimized against a multi-class loss function. Liao et al. [41] construct the hypothesis that vehicle parts differ in the discriminative capacity of estimating vehicle attributes. After a DPM-based detector localizes vehicle parts, multiple predictions are generated based on the

Table 4 Selected hand-crafted and deep learning methods for vehicle model recognition. Algorithms tested on the same dataset are grouped.

Method	Year	Dataset	#Image	#Model	Accuracy	Notes
Petrovic and Cootes [55]	2004	Petrovic and Cootes [55]	1,132	77	93%	Hand-crafted
Clady et al. [5]	2008	Clady et al. [5]	1,121	50	93.1%	Hand-crafted
Stark et al. [67]	2011	car-types [67]	1,904	14	93.5%	Hand-crafted
Krause et al. [39]	2013				94.5%	Hand-crafted
Krause et al. [39]	2013	BMW-10 [39]	512	10	76.0%	Hand-crafted
Krause et al. [39]	2013	car-197 [39]	16,185	197	67.6%	Hand-crafted
Krause et al. [38]	2015				92.8%	Deep learning
Hu et al. [25]	2017				93.1%	Deep learning
Xiang et al. [76]	2019				94.3%	Deep learning
Lin et al. [44]	2014	FG3DCar [44]	300	30	90.0%	Hand-crafted
Yang et al. [80]	2015	Subset of CompCars [80]	52,083	431	76.7%	Deep learning
Hu et al. [25]	2017				97.6%	Deep learning
Xiang et al. [76]	2019				98.5%	Deep learning
Jaderberg et al. [30, 81]	2015	Subset of BoxCars [64]	59,742	77	64.3%	Deep learning
Sochor et al. [64]	2016				75.4%	Deep learning
Fu et al. [16, 81]	2017				72.2%	Deep learning
Zeng et al. [81]	2019				81.2%	Deep learning
Lin et al. [43, 65]	2015	Subset of BoxCars116k [65]	90,840	107	69.6%	Deep learning
Simon and Rodner [63, 65]	2015				75.9%	Deep learning
Gao et al. [17, 65]	2016				70.6%	Deep learning
Sochor et al. [65]	2018				84.1%	Deep learning

Histograms of Oriented Gradient (HOG) [7] features of each part. Those predictions are accumulated with a weighting scheme allowing that larger weights are assigned to more influential parts.

Compared with the works explained above, some researchers divert the attention to 3D space. Krause et al. [39] extend two methods to obtain superior object representations in 3D. Established on the basis of 2D Spatial Pyramid [73], each rectified patch is associated with corresponding 3D coordinates. Likewise, the pooling regions in 2D BubbleBank [11] is switched from 2D to 3D. The resulting 3D representations from these two methods are combined with linear SVM [6] classifiers, which are trained in the manner of one-versus-all.

In [44], Lin et al. optimize 3D model alignment and fine-grained classification jointly. To begin with, the DPM method gives a rough estimation of part locations, while a pre-trained regression model further detects the representative landmarks. Each 3D model consists of a collection of 3D points, and the 3D object geometry is obtained by fitting the model to landmark locations in 2D. With hand-crafted features retrieved from each landmark, a multi-class linear SVM [6] classifier predicts the label. Most importantly, the predicted label is fed back to the 3D model to get better alignment results.

2.3.2 Deep Learning Methods

The CompCars [80] dataset contains a considerable number of vehicle images taken from all viewpoints with rich annotations. Yang et al. [80] utilize an Overfeat model [61] which is initialized with pretrained weights on the ImageNet [9] dataset, and fine-tune it for vehicle attribute recognition. The Overfeat model differs from the established AlexNet model [40] in three aspects: (i) it does not contain a contrast normalization scheme; (ii) adjacent pooling regions do not overlap and (iii) smaller stride value is used to get larger feature maps.

Hu et al. [25] surpass the previous work considerably on the strength of a novel spatially weighted pooling scheme. The pooling layer learns spatially weighted masks which assess the discriminative capacity of spatial units, and applies pooling operation to the extracted feature maps of the convolutional layer correspondingly.

Xiang et al. [76] propose a four-stage pipeline that takes the interaction between parts into account. Part detection is implemented using a backbone model truncated at an intermediate layer, while part assembling involves pointwise convolutional layers which gather associated parts into the same feature map. Afterward, topology constraint comprises depthwise convolutional layers and estimates the probability of the topology relationship between related parts. The ending classification uses a fully-connected layer to make predictions.

Sochor et al. are noted for collecting the BoxCars [64] and BoxCars116k [65] datasets. In [64], the recognition performance is boosted by inserting additional supplementary information to the neural network, more specifically, 3D vehicle bounding box, rasterized low-resolution shape, and 3D vehicle orientation. With 3D bounding boxes automatically obtained from surveillance cameras and the rasterized low-resolution shape information, a normalization procedure aligns the original images. Besides, the 3D orientation offers an insight into the viewpoint, which is beneficial.

Later on, the previous method is extended in [65] by proposing a method to estimate 3D bounding boxes in case of such information is unavailable. The directions to the vanishing points are obtained from three classification branches, which generate probabilities for each vanishing point belonging to a specific angle. In addition, two extra data augmentation strategies are integrated. On the one hand, the color of the image is randomly alternated. On the other hand, random crops in the images are filled with random noise.

Zeng et al. [81] devise a framework that learns a joint representation of the 2D global texture and 3D bounding box. The 2D global feature originates from a pre-trained detector that localizes the Region of Interest. The 3D perspective network regresses the 3D bounding box and extracts feature embedding. At last, the feature fusion network merges the set of two features and generates the predictions.

Unlike the aforementioned methods which are explicitly devised for vehicle make and model recognition, some generic image classification algorithms have also been validated, especially from the domain of fine-grained image classification. Jaderberg et al. [30] introduce a differentiable spatial transformer module, which makes the trained models more spatially invariant to the input data. A spatial transformer spatially transforms feature maps, and the manipulation is conditioned on the feature maps itself. The localisation network regresses the transformation parameters. The grid generator produces a set of points where the input feature maps should be sampled. Finally, the sampler samples the input feature maps at the grid points.

In [63], Simon and Rodner present a method that learns part models without the need of acquiring annotations of parts or bounding boxes. The channels in feature maps are treated as a part detector, and a part constellation model is obtained by selecting part detectors that fire at similar relative locations. As a side product, the filtered part proposals can be applied in the data augmentation pipeline.

Lin et al. [43] utilize a bilinear model consisting of two feature extractors. The feature vectors from those extractors

are multiplied using the outer product and pooled to obtain the bilinear vector. The resulting topology is significantly faster than methods that are dependent on detectors, and it is capable of capturing pairwise correlations between the feature channels.

From a different perspective, Gao et al. [17] targeted on reducing the size of feature representation in bilinear models, without compromising the discriminative capacity. Two compact bilinear pooling approaches are investigated to approximate the inner product of two descriptors, namely, Random Maclaurin [34] and Tensor Sketch [56]. The compact pooling methods are differentiable so that the pipeline can be optimized end-to-end.

Last but not least, Fu et al. [16] propose to recursively learn discriminative region attention and region-based feature representation at multiple scales. The classification sub-network provides sufficient discriminative capacity at each scale. The attention proposal sub-network starts from the full image, and iteratively generates region attention from coarse to fine. Meanwhile, the finer scale network takes a magnified region from previous scales in a recurrent manner.

3 Experimental Implementation of a Vehicle Attribute Recognizer

As discussed in Section 2.3, vehicle make and model recognition have been extensively studied, but there is still room for further investigation, especially due to the recently introduced large-scale datasets. Namely, especially people recognition has recently taken significant advances both in facial and full-body recognition, and some recent pipelines have not been experimented on vehicles yet. Therefore, we will next revisit a state-of-the-art method for people re-identification [50] and experiment on its out-of-the-box accuracy in the domain of vehicle attribute recognition. Moreover, the experiment also aims to present the details of implementing a vehicle identification model in a concrete manner.

3.1 Methodology

3.1.1 Model Architecture

The backbone model ResNet50 [23] is initialized with pre-trained weights on ImageNet [9]. The global average pooling layer shrinks the spatial dimensions of feature maps and generates a feature vector for each sample. On the condition that the categorical cross-entropy loss function is applied, a batch normalization [28] layer and a fully-connected layer are appended to the end so that the model predicts the probabilities of classes.

3.1.2 Loss Function

Three loss functions are examined, namely, categorical cross-entropy loss, triplet loss [60] and lifted structured loss [52]. The categorical cross-entropy loss function is widely used in conventional classification problems. The last two falls in the scope of metric learning, which optimizes feature embedding directly in such a way samples within the same class get closer while samples from different classes get further.

3.1.3 Training Procedure

Four strategies are implemented to boost performance. In the early stage of the training, the learning rate starts from a low value and increases gradually as the training proceeds. Such a warmup strategy cracks the distorted gradient issue [20, 46]. Moreover, random erasing data augmentation [83] is utilized to mask out random crops from original images, and it helps the model generalize better. Finally, label-smoothing regularization [70] encourages the model to make less confident predictions, and ℓ_2 regularization induces the model to choose smaller parameters.

3.1.4 Inference Procedure

Given a test sample, the output of the global average pooling layer is retrieved as the feature embedding. The cosine distance function is chosen to measure the distance between two feature vectors. While the entire test set corresponds to the query set, 100 samples are randomly selected from each class label in the training set and those samples constitute the gallery set. Each query sample takes the label of the closest match from the gallery set, and accuracy measures the percentage of cases in which the predicted label is consistent with the ground truth label.

The reason why we adopt the paradigm as mentioned above is that it is also applicable in cases with dynamic content. In the matter of a real-world vehicle model recognition system, automobile manufacturers release new car models regularly. The conventional classification approach, which computes the probabilities of classes that are available in the training set, is doomed to fail on unseen car models. In contrast, building up a gallery set and comparing the test sample with gallery samples might still yield a meaningful prediction. Whenever new car models get released, a certain amount of exemplary images could be added to the gallery set without the need of retraining the neural network itself.

3.2 Experimental Results

Table 5 reports the accuracy of categorical cross-entropy loss, triplet loss [60] and lifted structured loss [52] on

Table 5 Experimental results on the VERI-Wild [48] dataset which contains approximately 0.4 million images.

Method	#Type	Accuracy on Type	#Make	Accuracy on Make
cross-entropy loss		96.8%		95.6%
triplet loss	14	97.4%	149	95.3%
lifted structured loss		97.7%		96.2%

Test performances of categorical cross-entropy loss, triplet loss [60] and lifted structured loss [52] are evaluated on vehicle type and make recognition.

two tasks, *i.e.*, vehicle type and make recognition. It is explicit that lifted structured loss achieves superior performance than the other two loss functions. Therefore, incorporating the latest advances in metric learning, e.g., ArcFace loss [10], might push the boundaries of vehicle attribute recognition even further.

4 Future Research Directions

In this section, we point out four future research directions, namely, few-shot learning, multi-task learning, attention, and edge computing. Those topics are under-developed in the domain of vehicle attribute recognition.

4.1 Few-Shot Learning

New car models hit the consumer market from time to time. This fact poses a challenging problem, especially for any real-world vehicle model recognition system. Without adopting an appropriate solution, the classifier could not identify new car models. In the area of few-shot learning, a model has to recognize new classes with only a few samples from those unseen classes [74]. Setting up a vehicle attribute recognition pipeline in the manner of few-shot learning would make the algorithm more applicable in practice.

4.2 Multi-task Learning

Multi-task learning refers to the learning procedure in which multiples tasks are optimized simultaneously [59]. In ideal cases, the model achieves better generalization by reason of sharing feature representations across relevant tasks [35]. Some vehicle attribute recognition datasets provide several related attributes, e.g., CompCars [80], BoxCars [64] and VERI-Wild [48]. The availability of such annotations makes multi-task learning a feasible topic on vehicle attribute recognition.

4.3 Attention

Following the pattern of the human visual system, the attention mechanism allows the neural network to pay attention to certain salient parts of the input dynamically [78]. In terms of fine-grained image classification, the attention mechanism assists the model in spotting subtle visual differences between distinct categories and obtaining better performance [16, 77, 82]. It is anticipated that automobile manufacturers make incremental modifications to a specific model and introduce a variant model [80]. Therefore, the attention mechanism is beneficial, especially in hard cases, *i.e.*, different vehicle models from the same product series.

4.4 Edge Computing

While the cameras set up on the highway capture video streams, one could deploy the vehicle attribute recognition algorithms directly on edge devices. The concept of edge computing can be of great benefit in shorter response time, lower bandwidth cost, and safer data security [62]. Since edge devices are typically resource-constrained, lightweight models are preferable while balancing speed and accuracy. Possible solutions include parameter pruning and sharing, low-rank factorization, transferred/compact convolutional filters, and knowledge distillation [4].

5 Conclusion

In this paper, we have surveyed the state-of-the-art vehicle attribute recognition algorithms both for coarse-grained (vehicle type) and fine-grained (vehicle make and model) attributes. Since the advent of deep learning techniques, the recognition accuracy has taken significant leaps and is already likely to exceed human-level performance. The unforeseen progress enables new kinds of applications, that earlier low accuracy techniques did not facilitate.

In addition to the hardware resources which support efficient parallelization, another critical enabler for high accuracy is large-scale datasets. We listed an extensive collection of datasets for the three tasks and discussed their strengths and weaknesses. Those datasets contain considerable variations: some of them only have a fixed pose, fixed illumination, or a minimal set of categories; while others are taken “in the wild”, and the challenges brought by the unconstrained environment variables are demanding. In real-world applications, the situation is usually something between these extremes, and we believe that this survey helps in searching for suitable datasets and mixing them for maximum performance.

On the algorithms side, we discussed two typical training setups: classification setup (where each type/make/model is one class) and metric learning setup (where the model seeks to learn a distance function). The former approach is simple to comprehend and easy to implement. However, the challenge is the static nature of the classification model: if a new class appears, the model needs to be adapted. On the other hand, the metric learning approach may have a steeper learning curve, but it can offer increased flexibility and superior performance.

The classification approach is most suitable for cases where the classes are unvarying; for example, the unique vehicle types are unlikely to change any time soon. By contrast, more dynamic cases include vehicle make and model recognition, where new categories appear annually. Metric learning is preferable in such cases, while a test sample is compared to a gallery of examples using the nearest neighbor search method.

Finally, we compared the two approaches quantitatively for vehicle type and make recognition, and discovered that the metric learning approach with properly selected loss function outperforms the classification approach by a clear margin. This novel comparison based on a competitive pipeline is enlightening for practitioners, and it highlights the promising potential of metric learning, which has seen significant advances during recent years.

Acknowledgements This work was financially supported by Business Finland project 408/31/2018 MIDAS.

References

1. Abolghasemi, V., & Ahmadyfard, A. (2009). An edge-based color-aided method for license plate detection. *Image and Vision Computing*, *27*(8), 1134–1142.
2. Bai, S., Liu, Z., Yao, C. (2018). Classify vehicles in traffic scene images with deformable part-based models. *Machine Vision and Applications*, *29*(3), 393–403.
3. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*(1), 281–305.
4. Cheng, Y., Wang, D., Zhou, P., Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. arXiv:1710.09282.
5. Clady, X., Negri, P., Milgram, M., Poulenard, R. (2008). Multi-class vehicle type recognition system. In *IAPR workshop on artificial neural networks in pattern recognition* (pp. 228–239): Springer.
6. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.
7. Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE computer society conference on computer vision and pattern recognition, 2005. CVPR 2005*, (Vol. 1 pp. 886–893): IEEE.
8. Deb, K., Chae, H.U., Jo, K.H. (2009). Vehicle license plate detection method based on sliding concentric windows and histogram. *Journal of Computers*, *4*(8), 771–777.
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L. (2009). Imagenet: a large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition* (pp. 248–255): IEEE.
10. Deng, J., Guo, J., Xue, N., Zafeiriou, S. (2019). Arcface: additive angular margin loss for deep face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4690–4699).
11. Deng, J., Krause, J., Fei-Fei, L. (2013). Fine-grained crowdsourcing for fine-grained recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).
12. Dong, Z., & Jia, Y. (2013). Vehicle type classification using distributions of structural and appearance-based features. In *2013 IEEE international conference on image processing* (pp. 4321–4324): IEEE.
13. Dong, Z., Wu, Y., Pei, M., Jia, Y. (2015). Vehicle type classification using a semi-supervised convolutional neural network. *IEEE Transactions on Intelligent Transportation Systems*, *16*(4), 2247–2256.
14. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645.
15. Chollet, F. (2017). *Deep Learning with Python*, 1st edn. USA: Manning Publications Co.
16. Fu, J., Zheng, H., Mei, T. (2017). Look closer to see better: recurrent attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4438–4446).
17. Gao, Y., Beijbom, O., Zhang, N., Darrell, T. (2016). Compact bilinear pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 317–326).
18. Geiger, A., Lenz, P., Stiller, C., Urtasun, R. (2013). Vision meets robotics: the kitti dataset. *The International Journal of Robotics Research*, *32*(11), 1231–1237.
19. Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep learning*. Cambridge: MIT Press.
20. Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K. (2017). Accurate, large minibatch SGD: training ImageNet in 1 hour. arXiv:1706.02677.
21. Harris, C.G., & Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, (Vol. 15 pp. 10–5244): Citeseer.
22. He, K., Zhang, X., Ren, S., Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916.

23. He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).
24. Howard, A.G. (2013). Some improvements on deep convolutional neural network based image classification. arXiv:1312.5402.
25. Hu, Q., Wang, H., Li, T., Shen, C. (2017). Deep CNNs with spatially weighted pooling for fine-grained car recognition. *IEEE Transactions on Intelligent Transportation Systems*, *18*(11), 3147–3156.
26. Huttunen, H. (2019). Deep neural networks: a signal processing perspective. In *Handbook of signal processing systems* (pp. 133–163): Springer.
27. Huttunen, H., Yancheshmeh, F.S., Chen, K. (2016). Car type recognition with deep neural networks. In *2016 IEEE intelligent vehicles symposium (IV)* (pp. 1115–1120). https://doi.org/10.1109/IVS.2016.7535529.
28. Ioffe, S., & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv:1502.03167.
29. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.
30. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K. (2015). Spatial transformer networks. In *Advances in neural information processing systems* (pp. 2017–2025).
31. Jung, H., Choi, M.K., Jung, J., Lee, J.H., Kwon, S., Young Jung, W. (2017). Resnet-based vehicle classification and localization in traffic surveillance systems. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 61–67).
32. Jung, H., Lee, S., Yim, J., Park, S., Kim, J. (2015). Joint fine-tuning in deep neural networks for facial expression recognition. In *Proceedings of the IEEE international conference on computer vision* (pp. 2983–2991).
33. Kafai, M., & Bhanu, B. (2012). Dynamic bayesian networks for vehicle classification in video. *IEEE Transactions on Industrial Informatics*, *8*(1), 100–109.
34. Kar, P., & Karnick, H. (2012). Random feature maps for dot product kernels. In *Artificial intelligence and statistics* (pp. 583–591).
35. Kendall, A., Gal, Y., Cipolla, R. (2018). Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7482–7491).
36. Khata, M., Shvai, N., Hasnat, A., Llanza, A., Sanogo, A., Meicler, A., Nakib, A. (2019). Novel context-aware classification for highly accurate automatic toll collection. In *2019 IEEE Intelligent vehicles symposium (IV)* (pp. 1105–1110). https://doi.org/10.1109/IVS.2019.8813866.
37. Kim, P.K., & Lim, K.T. (2017). Vehicle type classification using bagging and convolutional neural network on multi view surveillance image. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 41–46).
38. Krause, J., Jin, H., Yang, J., Fei-Fei, L. (2015). Fine-grained recognition without part annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5546–5555).
39. Krause, J., Stark, M., Deng, J., Fei-fei, L. (2013). 3D object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 554–561).
40. Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).
41. Liao, L., Hu, R., Xiao, J., Wang, Q., Xiao, J., Chen, J. (2015). Exploiting effects of parts in fine-grained categorization of vehicles. In *IEEE international conference on image processing* (pp. 745–749): IEEE.
42. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L. (2014). Microsoft coco: common objects in context. In *European conference on computer vision* (pp. 740–755): Springer.
43. Lin, T.Y., RoyChowdhury, A., Maji, S. (2015). Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision* (pp. 1449–1457).
44. Lin, Y.L., Morariu, V.I., Hsu, W., Davis, L.S. (2014). Jointly optimizing 3D modelv fitting and fine-grained classification. In *European conference on computer vision* (pp. 466–480): Springer.
45. Liu, H., Tian, Y., Yang, Y., Pang, L., Huang, T. (2016). Deep relative distance learning: tell the difference between similar vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2167–2175).
46. Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J. (2019). On the variance of the adaptive learning rate and beyond. arXiv:1908.03265.
47. Liu, X., Liu, W., Ma, H., Fu, H. (2016). Large-scale vehicle re-identification in urban surveillance videos. In *IEEE international conference on multimedia and expo* (pp. 1–6): IEEE.
48. Lou, Y., Bai, Y., Liu, J., Wang, S., Duan, L. (2019). VERI-wild: a large dataset and a new method for vehicle re-identification in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3235–3243).
49. Lowe, D.G. (1999). Object recognition from local scale-invariant features. In *The proceedings of the seventh IEEE international conference on computer vision, 1999*, (Vol. 2 pp. 1150–1157): IEEE.
50. Luo, H., Gu, Y., Liao, X., Lai, S., Jiang, W. (2019). Bag of tricks and a strong baseline for deep person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (p. 0).
51. Luo, Z., Branchaud-Charron, F., Lemaire, C., Konrad, J., Li, S., Mishra, A., Achkar, A., Eichel, J., Jodoin, P.M. (2018). MIO-TCD: a new benchmark dataset for vehicle classification and localization. *IEEE Transactions on Image Processing*, *27*(10), 5129–5141.
52. Oh Song, H., Xiang, Y., Jegelka, S., Savarese, S. (2016). Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4004–4012).
53. Peng, Y., Jin, J.S., Luo, S., Xu, M., Au, S., Zhang, Z., Cui, Y. (2013). Vehicle type classification using data mining techniques. In *The era of interactive media* (pp. 325–335): Springer.
54. Peng, Y., Jin, J.S., Luo, S., Xu, M., Cui, Y. (2012). Vehicle type classification using PCA with self-clustering. In *IEEE international conference on multimedia and expo workshops* (pp. 384–389): IEEE.
55. Petrovic, V.S., & Cootes, T.F. (2004). Analysis of features for rigid structure vehicle type recognition. In *BMVC*, (Vol. 2 pp. 587–596).
56. Pham, N., & Pagh, R. (2013). Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 239–247): ACM.
57. Psyllos, A., Anagnostopoulos, C.N., Kayafas, E. (2011). Vehicle model recognition from frontal view image measurements. *Computer Standards & Interfaces*, *33*(2), 142–151.
58. Rachmadi, R.F., Uchimura, K., Koutaki, G., Ogata, K. (2018). Single image vehicle classification using pseudo long short-term memory classifier. *Journal of Visual Communication and Image Representation*, *56*, 265–274.

59. Ruder, S. (2017). An overview of multi-task learning in deep neural networks. arXiv:1706.05098.
60. Schroff, F., Kalenichenko, D., Philbin, J. (2015). Facenet: a unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815–823).
61. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y. (2013). Overfeat: integrated recognition, localization and detection using convolutional networks. arXiv:1312.6229.
62. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L. (2016). Edge computing: vision and challenges. *IEEE Internet of Things Journal*, *3*(5), 637–646.
63. Simon, M., & Rodner, E. (2015). Neural activation constellations: unsupervised part model discovery with convolutional networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1143–1151).
64. Sochor, J., Herout, A., Havel, J. (2016). Boxcars: 3D boxes as cnn input for improved fine-grained vehicle recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3006–3015).
65. Sochor, J., Spanhel, J., Herout, A. (2018). Boxcars: improving fine-grained recognition of vehicles using 3D bounding boxes in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, *20*(1), 97–108.
66. Specht, D.F. (1988). Probabilistic neural networks for classification, mapping, or associative memory. In *IEEE international conference on neural networks*, (Vol. 1 pp. 525–532).
67. Stark, M., Krause, J., Pepik, B., Meger, D., Little, J.J., Schiele, B., Koller, D. (2011). Fine-grained categorization for 3D scene understanding. *International Journal of Robotics Research*, *30*(13), 1543–1552.
68. Sun, W., Zhang, X., Shi, S., He, J., Jin, Y. (2017). Vehicle type recognition combining global and local features via two-stage classification. *Mathematical Problems in Engineering, 2017*.
69. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).
70. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818–2826).
71. Taek Lee, J., & Chung, Y. (2017). Deep learning-based vehicle classification using an ensemble of local expert and global networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 47–52).
72. Theagarajan, R., Pala, F., Bhanu, B. (2017). EDen: ensemble of deep networks for vehicle classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 33–40).
73. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y. (2010). Locality-constrained linear coding for image classification. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 3360–3367): Citeseer.
74. Wang, Y., & Yao, Q. (2019). Few-shot learning: a survey. arXiv:1904.05046.
75. Wieczorkowska, A., Kubera, E., Słowik, T., Skrzypiec, K. (2015). Spectral features for audio based vehicle identification. In *International workshop on new frontiers in mining complex patterns* (pp. 163–178): Springer.
76. Xiang, Y., Fu, Y., Huang, H. (2019). Global topology constraint network for fine-grained vehicle recognition. *IEEE Transactions on Intelligent Transportation Systems*.
77. Xiao, T., Xu, Y., Yang, K., Zhang, J., Peng, Y., Zhang, Z. (2015). The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 842–850).
78. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y. (2015). Show, attend and tell: neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).
79. Xu, Z., Yang, W., Meng, A., Lu, N., Huang, H., Ying, C., Huang, L. (2018). Towards end-to-end license plate detection and recognition: a large dataset and baseline. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 255–271).
80. Yang, L., Luo, P., Change Loy, C., Tang, X. (2015). A large-scale car dataset for fine-grained categorization and verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3973–3981).
81. Zeng, R., Ge, Z., Denman, S., Sridharan, S., Fookes, C. (2019). Geometry-constrained car recognition using a 3D perspective network. arXiv:1903.07916.
82. Zheng, H., Fu, J., Mei, T., Luo, J. (2017). Learning multi-attention convolutional neural network for fine-grained image recognition. In *Proceedings of the IEEE international conference on computer vision* (pp. 5209–5217).
83. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y. (2017). Random erasing data augmentation. arXiv:1708.04896.

Xingyang Ni received his MSc degree in Information Technology from Tampere University of Technology in 2016. Currently, he is pursuing a PhD degree in Computer Vision at Tampere University. He has contributed to various projects on Deep Learning, in both academia and industry. His research interests include person re-identification, feature learning and image retrieval.

Heikki Huttunen received his PhD Degree in Signal Processing at Tampere University of Technology (TUT), Finland, in 1999. Currently he is an associate professor at the Unit of Computing Sciences at Tampere University, and involved in industrial development projects on automated image analysis and pattern recognition. He is an author of over 100 research articles. His research interests include Optical Character Recognition (OCR), Computer vision, Pattern recognition and Statistics.

PUBLICATION

V

Block-optimized Variable Bit Rate Neural Image Compression

C. Aytekin, X. Ni, F. Cricri, J. Lainema, E. Aksu and M. Hannuksela

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2018, 2551–2554

Block-optimized Variable Bit Rate Neural Image Compression

Caglar Aytekin, Xingyang Ni, Francesco Cricri, Jani Lainema, Emre Aksu, Miska Hannuksela
Nokia Technologies
Tampere, Finland
{caglar.aytekin, xingyang.ni.ext, francesco.cricri}@nokia.com
{jani.lainema, emre.aksu, miska.hannuksela}@nokia.com

Abstract

In this work, we propose an end-to-end block-based auto-encoder system for image compression. We introduce novel contributions to neural-network based image compression, mainly in achieving binarization simulation, variable bit rates with multiple networks, entropy-friendly representations, inference-stage code optimization and performance-improving normalization layers in the auto-encoder. We evaluate and show the incremental performance increase of each of our contributions.

1. Introduction

Image compression has traditionally been addressed by transform-based methods such as JPEG [14] and BPG [11]. Recently, neural network based approaches have also been utilized such as hybrid approaches, where neural networks are used together with a traditional codec, or end-to-end learned approaches, where the codec consists solely of neural networks.

Regarding the hybrid approach, several works involve using neural networks as post-processing filters ([5], [6]), to enhance the decoded image. In [8] and [15] both pre-processing and post-processing neural networks are used. In [8] and [15], due to the non-differentiable traditional codec, an end-to-end training cannot be achieved. [15] proposes to utilize alternate training to overcome this issue. In the first stage, the pre-processing network is trained via a differentiable virtual codec. In the second stage, the real codec is used and only the post-processing network is trained.

Regarding the end-to-end learned approach, a typical architecture consists of an auto-encoder (see [9], [12]), where the encoder maps the input image to a low-dimensional tensor, and the decoder reconstructs the image.

The encoder's output, which typically consists of floating-point values, needs to be quantized in order to achieve reasonable compression rates. The quantization operation would provide zero gradients almost everywhere. In order to approximate the quantization, [4] propose to add a random sample from a uniform distribution. [13] uses a random mapping of floating-point values to binary values with a probability derived from the floating-point value.

Training of auto-encoders for data compression needs to account for both decoding quality and compression efficiency. One straightforward training loss for decoding quality is the mean squared error (MSE) between input and output of the auto-encoder. Minimizing the MSE maximizes the peak signal to noise ratio (PSNR), which is a widely used evaluation metric in data compression. However, a model trained with MSE loss tends to result into blurred decoded images. Alternative losses are variational losses [7], adversarial losses [2] and structural similarity loss [12].

Regarding the compression efficiency, [9] proposes to use an adaptive codelength regularization term which encourages structure in the code, so that the arithmetic coder can exploit it for adapting the final codelength to the complexity of the input. In [4] and [12] the authors optimize for rate-distortion performance, where the rate is represented by the entropy.

Other neural network architectures used for image compression include recurrent models, such as in [13].

In this paper, we propose a system for block-based image compression using auto-encoders. In particular, our contributions are:

- Using multiple networks for variable bit rate, with inference-stage code optimization.
- Using L_2 normalization layer as the first layer of decoder, which improves the training and inference performance.
- An entropy-friendly loss designed for block-based neural auto-encoders.
- Fine-tuning each network on a separate sub-set of blocks, according to the blocks' encoding difficulty.
- Interval-preserving binarization noise, which ensures that the noisy signal is in a certain interval to provide consistent input to the decoder during training.

2. Method

In this section, we describe the method used in our end-to-end image compression. Our method is based on fully-convolutional deep auto-encoders and is applied on 32x32 blocks from the image.

2.1. Network Description

Auto-Encoder Network: The encoder part contains five consecutive convolutional blocks. Each block consists of a convolutional layer with stride 2 followed by a parametric rectified linear unit (PReLU). These five blocks are followed by a 1x1 convolutional layer and a sigmoid activation. The output of this layer is the compressed signal and will be referred to as block-codes from now on. The block-codes are 1-dimensional, as the input to the network is of size 32x32 and there exits 5 downsampling convolutions.

The first layer of the decoder is an L_2 normalization layer. It has been shown that mapping auto-encoder representations to the hypersphere surface improves clustering [3]. We also find L_2 normalization beneficial for this work, and we will provide more details about the benefits later in the experimental results. The L_2 normalization layer is followed by five consecutive deconvolutional blocks, each upsampling to double size. Each deconvolutional block consists of a deconvolutional layer followed by PReLU. The five deconvolutional blocks are followed by a final 1x1 convolutional layer with sigmoid activation.

Multiple Networks: The block-code length (number of vector's entries) for the above network is fixed and equal to the number of convolutional kernels in the last layer of the encoder. Setting up a fixed code-length for all blocks can be suboptimal, as blocks may have different content complexity and thus different compression difficulty. To allow for variable bit rate encoding (other than entropy coding), we make use of three separate networks with different code-lengths. We encode/decode each block with the network that provides the smallest bit rate for a target PSNR value.

Deblocking Network: Due to block-based compression, the decoded image contains blocking artefacts. To suppress these artefacts, we employ a fully-convolutional deblocking filter that operates over the entire image. The network's structure is similar to U-Net [10].

2.2. Inference

During encoding, first the image is divided into 32x32 blocks by raster-scan. Each block is encoded by the lowest bit rate neural network (out of three) which satisfies a target PSNR. The output of the encoder is binarized.

We optimize each block-code by optimizing the encoder per block: we keep the weights of the decoder frozen and fine-tune the encoder for a single block. To this end, we set a target PSNR and start optimizing the encoder of lowest bit rate neural network. If this network cannot achieve the target PSNR, we move on to the higher bit rate neural network and optimize its encoder. This process is continued until the target PSNR is achieved and the corresponding block-code is selected as the final one.

We use two-bit indicator signal for each block indicating which neural network was used for encoding that block. All the indicator signals are concatenated and one long indicator vector is obtained for the entire image. The indicator vector is entropy-coded for further bit rate reduction. Similarly, each block-code is concatenated to obtain a long image-code. This image-code is first difference-coded and then entropy-coded. In the end, each image is encoded into three vectors: 1) entropy-coded image-code 2) entropy-coded indicator vector 3) shape of the original image.

During decoding, first the entropy-coded vectors are decoded. Then, the next two bits from indicator vector is read and based on the indicator, the decoder knows which of the three decoder network needs to be used for the current block and therefore the encoding dimension. Then the next bit sequence of same length as this encoding dimension is read from the image-code and is decoded by the selected neural decoder. We repeat the above procedure for all blocks. Next, we combine all blocks to reconstruct the entire image, by using the read shape information. Finally, the reconstructed image is passed through the deblocking network as a post-processing step.

2.3. Training

Binarization Simulation: The block-codes consist of floating-point numbers in the interval [0,1], which need to be binarized in order to achieve a reasonable compression rate. Yet, binarization operation is non-differentiable and cannot be used as is for training the auto-encoder end-to-end. Therefore, during training, we simulate the binarization by adding noise to a value $x[i]$ in the block-code x. The noise is random with a uniform distribution within the interval $[-|nint(x[i]) - x[i]|, |nint(x[i]) - x[i]|]$, where $nint$ denotes the rounding to nearest integer operation and $|.|$ is the absolute value operator. The noise is selected such that the resulting value with additive noise remains in the interval $[0, 1]$. This is to be able to provide a consistent input to the decoder when we use this approximation and when we use the real binarization.

Entropy-Friendly Loss: We concatenate the 1-dimensional block codes from the image, and the resulting image code is entropy-coded to achieve higher compression rates. To make the image code more suitable for entropy coding, we propose the loss in Eq. 1.

$$L_{entropy} = \frac{1}{N-1}\sum_{i=2}^{N}(x_p[i] - x_p[i-1])^2 \qquad (1)$$

In Eq. 1, x_p is obtained by padding the code x with 0 from both sides, where x corresponds to a block-code and N is the number of elements in x_p. This padding is beneficial since in the end we will concatenate all the block codes, thus enforcing both beginning and ends of each block-code to be zero helps achieving a smooth image-code after concatenation.

Training Process: During training, we use MSE-based reconstruction loss L_{rec} and the entropy loss with a regularization parameter λ as follows: $L = L_{rec} + \lambda L_{entropy}$.

Although we simulate the binarization via additive random noise as previously discussed, we found it further beneficial to utilize an alternate training. In each epoch, we first train the auto-encoder end-to-end with binarization simulation over the entire training dataset. Next, we freeze the encoder part, perform actual binarization on the codes and train only the decoder over the entire training dataset.

Since we are going to use each of three neural networks for a different encoding difficulty level, the above training can be suboptimal. In fact, training for example the lowest bit rate neural network with all blocks (including the hardest blocks), would not be consistent with the inference stage, when that network would never been used on hard to encode blocks. To make each network expert to their targeted blocks, we fine-tune each network with the blocks for which that network satisfies the target PSNR.

To make the decoder even more suitable to binarized codes, we keep the encoder frozen, use real binarization and fine-tune each expert decoder on its own training blocks.

The training of the deblocking network is performed separately where input images are the images reconstructed via the inference stage (except the deblocking part) and the ground truth are the original images.

3. Experimental Results

In this section, we quantitatively evaluate our method in CLIC image compression dataset [1]. As an evaluation metric we use the peak signal-to-noise ratio (PSNR). We calculate a single mean-squared-error (MSE) from the entire image dataset and calculate the corresponding PSNR according to Eq. 2. Note that the MSE is calculated on RGB images.

$$PSNR = 20 \log_{10} 255 - 10 \log_{10} MSE \qquad (2)$$

3.1. Implementation Details

All the convolutional or deconvolutional layers have 3x3 kernel size. The number of filters in the first five encoder's layers are 64, 128, 256, 512, 1024. The number of filters in the last layer of the encoder is different in each neural network: 64, 216 and 368 – these determine the number of values output by each encoder. The decoder simply follows the filter sizes for encoder in reverse order. The final layer has 3 filters to convert back to RGB space. The training is performed on 32x32 blocks extracted from images from training dataset with half-overlapping blocks. We refer to the half-overlapping variant as data augmentation training. The regularization parameter for entropy coding loss is selected as $\lambda = 0.001$. All neural networks were trained using a batch size of 256 and Adam optimizer with learning rate 0.001. We have two variants: *NTcodec* where we apply deblocking filter on blocks of size 255x255 for memory efficiency, and *NTcodecFull* where we apply deblocking filter on the entire image.

3.2. Effect of Each Contribution

First, we investigate the effect of each contribution by controlled experiments. In particular, we investigate the effect of noise simulation, L_2 normalization, data augmentation, alternate training and entropy loss. In each of these experiments, one of the above properties were removed and all others were kept fixed. We also compare our full model with a standard architecture where after each convolution and deconvolution layer there is a batch-normalization layer. In this standard model, we remove the introduced L_2 normalization layer, but keep all other components same. We have conducted the experiments only on the 216-bit neural network. We report the obtained validation PSNR in Table 1.

Noise simulation and alternate training have significant effects, as they are crucial for approximating the binarization process. The network with L_2 normalization results into a compression rate of 0.151 bits per pixel (bpp), whereas the one without to 0.134 bpp. The network without L_2 normalization can achieve the same performance (both in bpp and PSNR) with the network with L_2 normalization, however at encoding dimension of 236. As the encoding dimension increases, the training of the network takes longer, moreover the network size increases. Therefore, L_2 normalization has a positive effect in training speed and final network size. Data augmentation has a very minor effect on the performance due to already large number of training blocks and correlated blocks in the data augmentation. The model with batch-normalization behaves similarly to no-normalization network in terms of bit-rate and PSNR, i.e. L_2 normalization achieves similar performance with faster training and lower number of network parameters. Finally, since the entropy-loss acts as a regularization loss, it reduces the final PSNR value. However, the validation set bit rate (after entropy coding) with entropy-loss is 0.151 bits per pixel (bpp) whereas without the entropy-loss it is 0.216 bpp. Therefore, the huge compression rate improvement dominates the slight PSNR decrease.

Next, we investigate the effect of multiple networks, expert neural network training, final decoder fine-tuning, code-optimization and deblocking post-processing. Each experiment is done incrementally to each other in the above

Table 1. Effect of Each Contribution for a Single Neural Network with 216 encoding dimension (on Validation Dataset)

	No Noise Sim.	**No L_2**	**No Data Aug.**	**No Alt. Train.**	**No Entropy Loss**	**Batch-norm**	**Full Model**
PSNR	23.882	26.778	26.946	25.751	27.258	26.882	27.055

Table 2. Effect of Additinal Operations (on Validation Dataset)

	Single NN	**Multiple NN**	**Expert NN**	**Decoder Fine-Tune**	**Deblocking**	**Code Optimize**
PSNR	27.055	27.691	27.779	27.792	28.088	28.929

Table 3. Comparison on Test Set

	JPEG	**BPG**	**OURS**
PSNR	25.612	29.587	27.920
bpp	0.149	0.148	0.148

order. We report the validation PSNRs and the bit rates for each incremental training in Table 2. As we observe, using multiple neural networks provide a decent performance increase whereas expert trainings and decoder fine-tuning has only a minor incremental effect. Deblocking post-processing was aimed to help achieving visually better quality images, yet we also observe that it increases the performance too. Finally, block-wise code optimization greatly improves the performance and achieves a decent PSNR. We would like to note here that the average bit rate for the final model with code optimization is 0.149 bpp, which is below our baseline with single network with no additional processing (0.151 bpp).

Test-set results: Table 3 reports PSNR and bit rates on the test-set for our method and for two traditional codecs (JPEG and BPG).

4. Conclusion

We have proposed an end-to-end block-based auto-encoder system for learned image compression. We have evaluated each building block of our method and have shown that each building block contributes to the performance to a degree. Our novel contributions L_2 normalization, concatenation-enabling entropy-friendly loss, expert neural network fine-tuning and code optimization greatly contribute to our final performance.

References

[1] Workshop and challenge on learned image compression (clic). `http://www.compression.cc/challenge/`. Accessed: 2018-04-26.

[2] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool. Generative adversarial networks for extreme learned image compression. *arXiv preprint arXiv:1804.02958*, 2018.

[3] Ç. Aytekin, X. Ni, F. Cricri, and E. Aksu. Clustering and unsupervised anomaly detection with L2 normalized deep auto-encoder representations. *CoRR*, abs/1802.00187, 2018.

[4] J. Balle, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. In *ICLR*, 2017.

[5] L. Cavigelli, P. Hager, and L. Benini. Cas-cnn: A deep convolutional neural network for image compression artifact suppression. In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

[6] C. Dong, Y. Deng, C. C. Loy, and X. Tang. Compression artifacts reduction by a deep convolutional network. In *International Conference on Computer Vision (ICCV)*, 2015.

[7] K. Gregor, F. Besse, D. Jimenez Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3549–3557. Curran Associates, Inc., 2016.

[8] F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao. An end-to-end compression framework based on convolutional neural networks. *IEEE Transaction on Circuits and Systems for Video Technology*, 2017.

[9] O. Rippel and L. Bourdev. Real-time adaptive image compression. In *International Conference on Machine Learning*, 2017.

[10] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[11] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649–1668, Dec 2012.

[12] L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy image compression with compressive autoencoders. In *International Conference on Learning Representations*, 03 2017.

[13] G. Toderici, D. Vincent, N. Johnston, S. Jin Hwang, D. Minnen, J. Shor, and M. Covell. Full resolution image compression with recurrent neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[14] G. K. Wallace. The jpeg still picture compression standard. *Communications of the ACM*, pages 30–44, 1991.

[15] L. Zhao, H. Bai, A. Wang, and Y. Zhao. Learning a virtual codec based on deep convolutional neural network to compress image. *CoRR*, abs/1712.05969, 2017.

PUBLICATION VI

Clustering and Unsupervised Anomaly Detection with L2 Normalized Deep Auto-Encoder Representations

C. Aytekin, X. Ni, F. Cricri and E. Aksu

2018 International Joint Conference on Neural Networks (IJCNN), 2018, 1–6
DOI: 10.1109/IJCNN.2018.8489068

Clustering and Unsupervised Anomaly Detection with l_2 Normalized Deep Auto-Encoder Representations

Caglar Aytekin, Xingyang Ni, Francesco Cricri and Emre Aksu
Nokia Technologies, Tampere, Finland
Corresponding Author e-mail: caglar.aytekin@nokia.com

***Abstract*—Clustering is essential to many tasks in pattern recognition and computer vision. With the advent of deep learning, there is an increasing interest in learning deep unsupervised representations for clustering analysis. Many works on this domain rely on variants of auto-encoders and use the encoder outputs as representations/features for clustering. In this paper, we show that an l_2 normalization constraint on these representations during auto-encoder training, makes the representations more separable and compact in the Euclidean space after training. This greatly improves the clustering accuracy when k-means clustering is employed on the representations. We also propose a clustering based unsupervised anomaly detection method using l_2 normalized deep auto-encoder representations. We show the effect of l_2 normalization on anomaly detection accuracy. We further show that the proposed anomaly detection method greatly improves accuracy compared to previously proposed deep methods such as reconstruction error based anomaly detection.**

I. Introduction

Cluster analysis is essential to many applications in computer vision and pattern recognition. Given this fact and the recent advent of deep learning, there is an increasing interest in learning deep unsupervised representations for clustering analysis [1], [2], [3], [4]. Most of the methods that perform clustering on deep representations, make use of auto-encoder representations (output of the encoder part) and define clustering losses on them. The focus of previous works have been on the choice of the auto-encoder type and architecture and the clustering loss. In DEC [1], first a dense auto-encoder is trained with minimizing reconstruction error. Then, as a clustering optimization stage, the method iterates between computing an auxiliary target distribution from auto-encoder representations and minimizing the Kullback-Leibler divergence to it. In IDEC [2], it is argued that the clustering loss of DEC corrupts the feature space, therefore IDEC proposes to jointly optimize the clustering loss and reconstruction loss of the auto-encoder. DCEC [4] argues the inefficiency of using dense auto-encoders for image clustering, therefore adopts a convolutional auto-encoder and shows that it improves the clustering accuracy of DEC and IDEC. GMVAE [3] adopts a variational auto-encoder in order to learn unsupervised representations and simply applies K-means clustering on representations.

In this manuscript, we show that regardless of the auto-encoder type (dense or convolutional), constraining the auto-encoder representations to be on the unit-ball, i.e. to be l_2 normalized, during auto-encoder training, greatly improves the clustering accuracy. We show that a simple k-means clustering on the auto-encoder representations trained with our constraint already gives improved accuracy with a large margin compared to baselines with or without additional clustering losses. Motivated by the high performance of our clustering method on deep representations, we propose an unsupervised anomaly detection method based on this clustering. We show that our anomaly detection method greatly improves on other deep anomaly detection strategies such as reconstruction error based ones. We also investigate the effect of l_2 normalization constraint during training on the anomaly detection accuracy and show that it leads to superior results compared to not applying the constraint.

II. Related Work

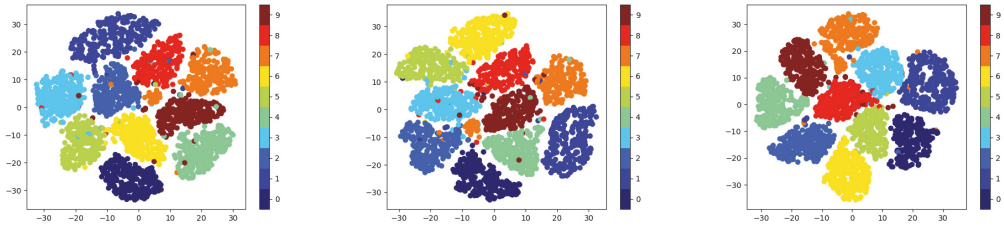
A. Deep Unsupervised Anomaly Detection

Unsupervised anomaly detection tries to find anomalies in the data without using any annotation [7]. Recently, deep learning methods have also been used for this task [5], [6]. These works train auto-encoders on the entire data and use reconstruction loss as an indicator of anomaly. DRAE [5] trains auto-encoders and uses reconstruction error as an anomaly indicator. Moreover, DRAE proposes a method to make the reconstruction error distributions of the normal and abnormal classes even more separable so that it is easier to detect anomalies. AVAE [6] trains both conventional and variational auto-encoders and use reconstruction error as an anomaly indicator.

The general assumption of the above works is that since the anomaly data is smaller in ratio than the normal data, the auto-encoder would not learn to reconstruct it accurately.

The above assumption seems to work in a specific definition of anomaly where the normal samples are drawn from a single class only and anomaly classes have been selected from many other classes [5]. However, the assumption fails in another anomaly type where the normal samples are drawn from multiple classes and anomaly class is sampled from a specific class [6].

In this paper we propose an unsupervised anomaly detection method based on clustering on deep auto-encoder representations and show that it gives a superior performance than reconstruction error based anomaly.



(a) No normalization (b) l_2 normalization after training (c) l_2 normalization during training

Fig. 1: Illustration of t-SNE encoding of auto-encoder representations for MNIST dataset to two dimensions. Best viewed in color.

B. Regularization and Normalization in Neural Networks

Due to the high number of parameters, neural networks have a risk of over-fitting to the training data. This sometimes reduces the generalization ability of the learned network. In order to deal with over-fitting, mostly regularization methods are employed. One of the most widely used regularization technique is weight norm regularization. Here the aim is to add an additional regularization loss to the neural network error, which gives high penalty to weights that have high norms. Both l_1 and l_2 norm can be exploited.

Recently some normalization techniques for neural networks emerged such as [8], [9]. Batch normalization [8], aims to find a statistical mean and variance for the activations which are calculated and updated according to batch statistics. The activations are normalized according to these statistics. In layer normalization [9], the mean and variance are computed from all of the summed inputs to the neurons on a layer on a single training sample. This overcomes the batch-size dependency drawback of batch-normalization. Although these methods were mainly proposed as tricks to make the neural network training faster by conditioning each layer's input, it is argued that they may also have a regularization effect due to their varying estimations of parameters for standardization at each epoch.

In our proposed method, the unit ball constraint that we put on activations is a normalization technique. However, unlike layer or batch normalization, the unit ball constraint is parameter-free as it simply sets the norm of each activation vector to 1. Therefore, it is free from the parameter estimation stochasticity. Yet, it may still act as a regularization method due to its hard constraint on some activations to be of fixed norm. This slightly resembles the l_2 norm regularization. A key difference is that in l_2 norm regularization, the norm is of the weights, but in our case it is applied on the activations. Another key difference is that we fix the activation norms to 1, whereas l_2 norm regularization penalizes to large weight norms and does not fix the norms to any value.

III. Proposed Method

A. Clustering on l_2 Normalized Deep Auto-Encoder Representations

We represent the auto-encoder representations for the input I as $E(I)$ and the reconstructed input as the $D(E(I))$. The representations are generally obtained via several dense or convolutional layers applied on the input I. In each layer, usually there are filtering and an activation operations, and optionally pooling operations. Let f_i be the computations applied to the input at layer i, then the encoded representations for an n-layer encoder are obtained as in Eq. 1.

$$E(I) = f_n(f_{n-1}(...f_1(I))) \tag{1}$$

The reconstruction part of the auto-encoder applies on $E(I)$ and is obtained via several dense or deconvolutional layers. In each layer, usually there are filtering and activation operations and optionally un-pooling or up-sampling operations. Let g_i be the computations applied to the auto-encoder representations, then the reconstructed signal for an m-layer decoder is obtained as as in Eq. 2.

$$D(E(I)) = g_m(g_{m-1}(...g_1(E(I)))) \tag{2}$$

The auto-encoder training is conducted in order to reduce the reconstruction error (loss) given in Eq. 3.

$$L = \frac{1}{|J|}\sum_{j \in J}(I_j - D(E(I_j)))^2 \tag{3}$$

Here, we propose an additional step conducted on the auto-encoder representations $E(I)$. In particular, we apply l_2 normalization on $E(I)$. This corresponds to adding a hard constraint on the representations to be on the unit ball. The loss function with our introduced constraint can then be written as in Eq. 4, where L_c and E_c are loss and encoded representations with our introduced constraint.

(a) Clustering Method

(b) Normality Score Method

Fig. 2: Illustration of proposed methods in inference phase.

$$L_c = \frac{1}{|J|}\sum_{j\in J}(I_j - D(E_c(I_j)))^2, \qquad E_c(I) = \frac{E(I)}{\|E(I)\|_2} \tag{4}$$

We believe that l_2 normalized features (representations) are more suitable for clustering purposes, especially for the methods that use Euclidean distance such as conventional k-means. This is because the distances between the vectors would be independent of their length, and would instead depend on the angle between the vectors. As a positive side effect, enforcing the unit norm on representations would act as a regularization for the entire auto-encoder.

In Fig. 1, we illustrate t-SNE [10] encoding (to 2 dimensions) of the auto-encoder representations of networks with same architecture. All auto-encoders were trained on MNIST [11] dataset training split. The representations corresponding to Fig. 1a are from the auto-encoder that was trained by the loss in Eq. 3. The same representations with l_2 normalization applied after training are illustrated in Fig. 1b. Finally, the representations with l_2 constraint during training , i.e. training with loss Eq. 4, are illustrated in Fig. 1c.

It is observed from the figures that the l_2 normalization during training (Fig. 1c) results into more separable clusters. One example is the distributions of digit 7 in MNIST dataset. Note that the numbers are indicated with color codes where the color bar is available in Fig. 1. It is clearly observed that with no normalization during training (Fig. 1a), digit 7 is divided into 2 parts where the small part is surrounded by 8,9,6 and 2 digits. With normalization applied after training (Fig. 1b) this effect becomes even more evident. So, we clearly observe here that applying normalization after training does not help at all. But, with l_2 normalization constraint during training (Fig. 1c), we see a clear separation of digit 7 as a single cluster from the rest of the numbers. Moreover, it can be observed that the clusters are more compact in Fig. 1c compared to others.

After training the auto-encoder with the loss function in Eq. 4, the clustering is simply performed by k-means algorithm. No more clustering loss is applied. Our clustering method is illustrated in Fig. 2a.

B. Unsupervised Anomaly Detection using l_2 Normalized Deep Auto-Encoder Representations

Here, we propose a clustering based unsupervised anomaly detection. We train an auto-encoder on the entire dataset including normal and abnormal samples and no annotation or supervision is used. The auto-encoder is simply trained with the loss in Eq. 4. After training, the l_2 normalized auto-encoder representations are clustered with k-means algorithm. We assume the anomaly cases to be considerably smaller in number than any normal clusters. Note that this assumption does not put any constraint on the dataset, but it simply follows the general definition of anomaly. Therefore, the centroids obtained by the k-means method can be considered to be representations of normal clusters by some errors that are caused by anomaly samples in the dataset. To each sample i, we assign the normality score v_i in Eq. 5.

$$v_i = \max_j (E_c(I_i) \cdot \frac{C_j}{\|C_j\|_2}) \tag{5}$$

In Eq. 5, C_j is a cluster centroid and $\cdot$ is the dot product operator. Notice that we l_2 normalize the cluster centroids. Since representations $E_c(I_i)$ are already l_2 normalized, $v_i \in [0, 1]$ holds. The measure in Eq. 5 is intuitive considering that we expect high similarities of normal samples to normal classes. Our normality scoring method is illustrated in Fig. 2b.

The normality score can be used for anomaly detection in a straightforward manner. Simply, the abnormal samples can be detected as the ones having $v_i < \tau$, where $\tau \in [0, 1]$ is a threshold.

IV. Experimental Results

A. Clustering

Evaluation Metrics: We use the widely used evaluation for unsupervised clustering accuracy [1] given in Eq. 6.

$$acc = \max_m \frac{\sum_{i=1}^{n} \mathbf{1}\{l_i = m(c_i)\}}{n} \tag{6}$$

In Eq. 6, l_i is the ground truth labeling of sample i, c_i is the cluster assignment according to the one-to-one mapping defined by m, where m ranges over all possible one-to-one mappings between clusters generated by the algorithm and the ground truth labels. The maximization in Eq. 6 can be performed by Hungarian algorithm [12].

We compare the clustering accuracy of auto-encoder representations with and without l_2 normalization constraint. We make this comparison in dense and convolutional auto-encoders. For dense auto-encoder, we use MNIST [11] and for convolutional auto-encoders, we use MNIST [11] and USPS [15] datasets. This is due to the availability of results in the works that use dense and convolutional auto-encoders.

For dense auto-encoder, we use the network structure which is used both in DEC [1] and IDEC [2]. In encoding part there are 4 layers with $500 - 500 - 2000 - 10$ hidden neurons and in decoding part there are $2000 - 500 - 500 - d$ neurons, where d is the dimension of the input. We re-implement the auto-encoder training and with leaky relu [13] activations after each hidden layer except for the last one and trained the auto-encoder end to end for 100 epochs. We select the best model with lowest reconstruction error. As it can be observed from Table I, we obtain a very similar clustering accuracy when we apply k-means on auto-encoder representations compared to the original paper of DEC [1]. Note here that results indicated with * corresponds to our own implementation. Other results for baselines are borrowed from original papers. Table I shows that when we train the auto-encoder with our l_2 normalization constraint on the representations, we achieve a much better clustering accuracy when we apply k-means on the representations. We denote our method as AE-l_2 which stands for auto-encoder with l_2 normalization. Moreover, our clustering accuracy is even better than the methods that define a separately designed clustering loss on the representations (DEC and IDEC).

Next, we make experiments for the convolutional auto-encoder. For this, we make use of the model structure introduced in DCEC [4]. This model consists of 5x5, 5x5 and 3x3 convolutional filters in the encoding layers respectively. There are 32,64 and 128 filters in encoding layers respectively. Convolutions are applied with 2x2 strides and with relu [14] activations. After the convolutional layers, the activations are flattened and there is a dense layer of dimension 10. This is followed by another dense layer and reshaping. Decoder part consists of 64,32 and 1 deconvolutional filters of size 3x3, 5x5 and 5x5 respectively. Relu activations were applied after each convolution, except for the last one. The network was trained for 200 epochs as in the original paper of DCEC [4]. In Table II, we show clustering accuracy of k-means applied on convolutional autoencoder representations. We were able to obtain similar results as in the original paper (DCEC). Note here that results indicated with * corresponds to our own implementation. Other results for baselines are borrowed from original papers. It can be observed from Table II that when we train the convolutional autoencoder with our l_2 normalization constraint on representations, we achieve a much better performance. We denote our method as CAE-l_2 which stands for convolutional auto-encoder with l_2 normalization. Our performance is superior to DCEC which introduces additional clustering loss.

TABLE I: Clustering on Dense Auto-Encoder Representations

	AE* k-means	AE k-means	DEC	IDEC	AE-l_2 k-means
MNIST	81.43	81.82	86.55	88.06	**90.20**

TABLE II: Clustering on Convolutional Auto-Encoder Representations

	CAE* k-means	CAE k-means	DCEC	CAE-l_2 k-means
MNIST	84.83	84.90	88.97	**95.11**
USPS	73.521	74.15	79.00	**91.35**

TABLE III: Comparison of Normalization Methods

	batch-norm	layer-norm	l_2-norm
MNIST	70.67	70.83	**95.11**
USPS	74.95	75.263	**91.35**

l_2 versus Batch and Layer Normalization: Due to l_2 normalization step in our clustering method, we compare it with applying other normalization techniques training. In particular we train two separate networks by using batch [8] and layer [9] normalization, instead of l_2 normalization. All other setup for the experiments are the same. Batch size of 256 is used for all methods in order to have a large enough batch for batch normalization. Our method performs superior to both baselines by a large margin, as the accuracies in Table III indicate. More importantly it is noticed that neither batch nor layer normalization provides a noticeable accuracy increase over the baseline (CAE+k-means). Moreover in MNIST dataset, layer and batch normalization results into a significant accuracy decrease. This is an important indicator showing that the performance upgrade of our method is not a result of a input conditioning, but it is a result of the specific normalization type that is more fit for clustering in Euclidean space.

B. Anomaly Detection

Evaluation Metrics: An anomaly detection method often generates an anomaly score, not a hard classification result. Therefore, a common evaluation strategy in anomaly detection is to threshold this anomaly score and form a receiver operating curve where each point is the true positive and false positive rate of the anomaly detection result corresponding to a threshold. Then, the area under the curve (AUC) of RoC curve is used as an evaluation of the anomaly detection method [7].

Here, we evaluate our method introduced in Section III-B. The evaluation setup and implementation of our method are as follows. In MNIST training dataset, we select a digit class as anomaly class and keep a random 10% of that class in the dataset while the remaining 90% is ignored. We leave the rest of the classes as is. Then, we use the convolutional autoencoder structure in DCEC [4] and train it with our l_2 normalization constraint on representations. Finally, we apply k-means clustering on the representations and keep the centroids. In our experiments we use k=9 for k-means, since we assume that we know the number of normal

classes in the data. For MNIST test dataset, we calculate the auto-encoder representations. As a normality measure for each sample, we calculate the corresponding representation's maximum similarity to pre-calculated cluster centroids as in 5. It should be noted that we repeat the above procedure by choosing a different class to be anomaly class, for all possible classes.

We also evaluate two baselines. In the first baseline, we exactly repeat the above procedure, but without l_2 normalization constraint on representations. In the second baseline, again we train the auto-encoder with l_2 normalization constraint on representations. Then, on the test set, we calculate the reconstruction error per sample and define that as anomaly score. Using reconstruction error based anomaly detection follows the works in AVAE [6] and DRAE [5]. The setups in AVAE and DRAE are different than ours. In AVAE, the training is only conducted on normal data, so the method is not entirely unsupervised in that sense. In DRAE, the anomaly definition is different: only a single class is kept as normal and samples from other classes are treated as anomaly. That setup presents a much easier case and therefore reconstruction error based anomaly detection produces acceptable results. Next, we show that in our setup this is not the case.

For each method we plot a RoC curve via thresholding the normality (or anomaly) score with multiple thresholds. Then, we evaluate the area under the RoC curve (AUC) for measuring the anomaly detection performance. The training and test datasets for all methods are the same. Due to the random selection of 10% of the anomaly class to be kept, performance can change according to the partition that is randomly selected. Therefore, we run the method 10 times for different random partitions and report the mean AUC.

It can be observed from Table IV that our clustering based anomaly detection method drastically outperforms the reconstruction error based anomaly detection for CAE neural network structure.

It is worth noting here an interesting observation from Table IV: for digits 1, 7 and 9, reconstruction error based anomaly detection gets a very inferior performance. This is most evident in digit 1. The reason for this is that the digit 1 is very easy to reconstruct (only 1 stroke) and even though an auto-encoder is trained on much less examples of this digit, it can reconstruct it quite well. This shows a clear drawback of the reconstruction error based anomaly detection. However, in our clustering based method, we achieve a very high accuracy in all the digits.

The effect of our proposed l_2 normalization constraint on representations during training can also be observed from Table IV. In $9/10$ cases, i.e. digits selected as anomaly, anomaly detection with the network trained with l_2 normalization constraint on representations performs much better than the one without. Only in digit 9, we observe an inferior accuracy of our method. Compared to other digits, we observe less performance for digits 4 and 9. We argue that this might be happening due to very similar appearance of these digits in some handwritings. Therefore, the method may confuse these numbers with each other during clustering.

TABLE IV: Anomaly Detection with Auto-Encoder Representations

Anom. Digit	CAE. (recons)	CAE (cluster)	CAE-l_2 (cluster)
0	0.7025	0.7998	**0.9615**
1	0.0782	0.8871	**0.9673**
2	0.879	0.7512	**0.9790**
3	0.8324	0.8449	**0.9382**
4	0.7149	0.4988	**0.7825**
5	0.8359	0.7635	**0.9136**
6	0.6925	0.7896	**0.9497**
7	0.5767	0.7421	**0.9100**
8	0.8912	0.9200	**0.9237**
9	0.514	**0.8944**	0.7495

TABLE V: Anomaly Detection with Auto-Encoder Representations

Anom. (Digit)	AE. (recons.)	VAE. (recons.)	CAE-l_2 cluster
0	0.825	0.917	**0.9615**
1	0.135	0.136	**0.9673**
2	0.874	0.921	**0.9790**
3	0.761	0.781	**0.9382**
4	0.727	**0.808**	0.7825
5	0.792	0.862	**0.9136**
6	0.812	0.848	**0.9497**
7	0.508	0.596	**0.9100**
8	0.869	0.895	**0.9237**
9	0.548	0.545	**0.7495**

In Table V, we compare our method to another method [6] that performs reconstruction error based anomaly detection, but using dense auto-encoders. There is also a variational auto-encoder based version of the method. It should be noted that this method trains auto-encoders only on normal data. This presents a much easier task compared to our case where we also include anomalous samples during training. Thus our case is entirely unsupervised. Still, $9/10$ cases, our method outperforms both variants of the method with a large margin. Only in digit 4, we observe an inferior performance of our method compared to VAE method.

V. Conclusion

In this paper, we have applied a l_2 normalization constraint to deep autoencoder representations during autoencoder training and observed that the representations obtained in this way clusters well in Euclidean space. Therefore, applying a simple k-means clustering on these representations gives high clustering accuracies and works better than methods defining additional clustering losses. We have also shown that the high performance is not due to any conditioning applied on the representations but it is due to selection of a particular normalization that leads to more separable clusters in Euclidean space. We have proposed an unsupervised anomaly detection method on l_2 normalized deep auto-encoder representations.

We have shown that the proposed l_2 normalization constraint drastically increases the anomaly detection method's performance. Finally, we have shown that the commonly adopted deep anomaly detection method based on the reconstruction error performs weak in a definition of anomaly, whereas our method performs superior.

REFERENCES

[1] J. Xie, R. Girshick and A. Farhadi, *Unsupervised deep embedding for clustering analysis*, International Conference on Machine Learning, pp. 478-487, June, 2016.

[2] X. Guo, L. Gao, X. Liu and J. Yin, *Improved deep embedded clustering with local structure preservation*, International Joint Conference on Artificial Intelligence, pp. 1753-1759, June, 2017.

[3] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M- C- Lee, H. Salimbeni, K- Arulkumaran and M. Shanahan, *Deep unsupervised clustering with gaussian mixture variational autoencoders*, arXiv preprint arXiv:1611.02648, 2016.

[4] X. Huo, X. Liu, E. Zheand J. Yin, *Deep Clustering with Convolutional Autoencoders*, International Conference on Neural Information Processing, pp. 373-382, 2017.

[5] Y. Xia, X. Cao, F. Wen, G. Hua and J. Sun, *Learning discriminative reconstructions for unsupervised outlier removal*, Proceedings of the IEEE International Conference on Computer Vision, pp. 1511-1519, 2015.

[6] J. An and S. Cho, *Variational autoencoder based anomaly detection using reconstruction probability*, SNU Data Mining Center, Tech. Rep., 2015.

[7] M. Goldstein and S. Uchida, *A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data*, PloS one, vol 11, no. 4, 2016.

[8] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, International conference on machine learning, pp. 448-456, 2015.

[9] J. L. Ba, J. R. Kiros and G. E. Hinton, *Layer normalization*, arXiv preprint arXiv:1607.06450, 2016.

[10] L. V. D. Maaten and G. Hinton, *Visualizing data using t-SNE*, Journal of machine learning research, pp. 2579-2605, 2008.

[11] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.

[12] H. W. Kuhn, *The Hungarian method for the assignment problem*, Naval Research Logistics, 2(1-2), pp. 83-97, 1955.

[13] V. Nair and G. E. Hinton, *Empirical evaluation of rectified activations in convolutional network*, arXiv preprint arXiv:1505.00853, 2015.

[14] B. Xu, N. Wang, T. Chen and M. Li, *Rectified linear units improve restricted boltzmann machines*, International Conference on Machine Learning, pp. 807-814, 2010.

[15] Y. LeCun, O. Matan, B. Boser, J. D. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jacket and H. S. Baird, *Handwritten zip code recognition with multilayer networks*. International Conference on Pattern Recognition, vol. 2, pp. 35-40, 1990.