Tampere University

Toni Olán

AUDIO-VISUAL SCENE CLASSIFICATION WITH DIFFERENT LATE FUSION COMBINERS

Bachelor's Thesis
Faculty of Information Technology and Communication Sciences
Examiner: Prof. Tuomas Virtanen
June 2021

ABSTRACT

Toni Olán: Audio-Visual Scene Classification with Different Late Fusion Combiners
Bachelor's Thesis
Tampere University
Information Technology
June 2021

Humans use multiple senses when performing a task. Senses are analogous to modalities, which represent a way in which something is experienced or perceived. With the increasing availability of multimodal data, modeling the human behavior with multimodality has gained more popularity on various machine learning tasks. However, in scene classification, the most popular classifiers still rely either only on audio or images.

Fusing correlated modalities right after feature extraction has been shown to achieve higher accuracy than fusing the predictions of unimodal classifiers. However, this later fusion is more flexible, as a system can still output decent results even if a modality is lost. The predictions of the classifiers based on different modalities can be combined in various ways, and the selection of a combination method is still an open issue.

This thesis studies the accuracies of scene classifiers that are based on audio and visual modalities. The performance of a multimodal ensemble classifier is also compared to the unimodal classifiers. The ensemble model consists of all the studied unimodal classifiers and a combiner, which can combine the predictions of the base classifiers in different ways.

Two classifiers are built for both audio and visual modalities. These unimodal classifiers rely on pre-trained models for feature extractions and small trainable neural networks for the classifications. The pre-trained models utilize transfer learning, as they have been trained for recognizing objects instead of scenes. The selected audio models consist of VGGish and L3, whereas the visual ones consist of ResNet and L3.

The experiments show that the classifiers using the same modality perform similarly, obtaining their highest and lowest scores on the same scenes, with some exceptions. Both audio classifiers perform better on certain scenes than the visual classifiers and vice versa, while a classifier based on the visual modality works better on average. However, the chosen pre-trained model has a clear impact on a classifier, as the other classifier reaches lower accuracies in both modalities. The best performing combiners in the experiments are a simple mean combiner and its class-weighted variant. Compared to the base classifiers, the best ensemble classifier obtains a higher average accuracy and only a single scene out of ten obtains a lower classification score.

Keywords: Multimodality, Scene classification, Late fusion, Transfer learning

TIIVISTELMÄ

Toni Olán: Audio-Visuaalinen Ympäristön Tunnistus Eri Myöhäisyhdistämismenetelmillä
Kandidaatin työ
Tampereen yliopisto
Tietotekniikka
Kesäkuu 2021

Ihmiset hyödyntävät useita aistejaan suorittaessaan tehtäviä. Aistit ovat analogisia modaliteettien kanssa, joilla kuvataan tapaa, miten jokin asia koetaan. Ihmismäisen käyttäytymisen mallintaminen multimodaalisuuden avulla on saavuttanut enemmän suosiota eri koneoppimistehtävissä, sillä useammasta modaliteetista koostuva data on nykään helpommin saatavilla. Ympäristön tunnistuksessa suosituimmat luokittimet hyödyntävät kuitenkin edelleen vain joko audiota tai kuvia.

Korreloivien modaliteettien yhdistäminen heti datan tiivistämisen jälkeen on todettu tuottavan parempia tuloksia kuin erillisten yhteen modaliteettiin perustuvien luokittimien tuloksien yhdistäminen. Tämä myöhempi yhdistäminen on kuitenkin joustavampi ratkaisu, sillä sitä hyödyntävä luokitin tuottaa kohtuullisia tuloksia, vaikka jokin modaliteetti ei olisi käytössä. Eri modaliteetteihin perustuvien luokittimien tulokset voidaan yhdistää useammalla eri tavalla, ja yhdistelmätavan valinta on edelleen avoin ongelma.

Tässä työssä tutkitaan eri modaliteetteihin perustuvien luokittimien tarkkuutta ympäristön tunnistamisessa. Luokittimien avulla kootaan myös useampaa modaliteettia hyödyntävä luokitinkokonaisuus, jonka tarkkuutta verrataan yksittäisiin luokittimiin. Yksi luokitin hyödyntää joko audio- tai kuvamodaliteettia. Luokitinkokonaisuus koostuu kaikkien luokittimien lisäksi myös yhdistäjästä, joka voi yhdistää luokittimien tuloksia eri tavoin.

Sekä audion että kuvan luokitteluun rakennetaan kaksi luokitinta. Nämä luokittimet käyttävät esiopetettuja malleja datan tiivistämiseen ja pieniä opetettavia neuroverkkoja itse luokitteluun. Esiopetetut mallit hyödyntävät siirto-oppimista, sillä ne ovat opetettu tunnistamaan esineitä eikä ympäristöjä. Valitut mallit ovat audioluokittimille VGGish ja L3 sekä vastaavasti kuvaluokittimille ResNet ja L3.

Tuloksista huomataan, että samaa modaliteettia hyödyntävät luokittimet käyttäytyvät samanlailla saavuttamalla niiden parhaimmat ja huonoimmat tuloksensa samoissa ympäristöissä muutamalla poikkeuksella. Molemmilla modaliteeteilla luokittimet suoriutuvat paremmin useammassa ympäristössä kuin toisen modaliteetin luokittimet, ja kuvamodaliteettiin perustuva luokitin toimii keskimäärin paremmin. Valitulla esiopetetulla mallilla on kuitenkin selkeä vaikutus luokittimen tarkkuuteen, sillä samaa modaliteettia hyödyntävissä luokittimissa toisen tulokset ovat matalampia. Yhdistäjistä korkeimman tarkkuuden saavuttaa sekä yksinkertainen keskiarvoyhdistäjä että sen luokkapainoarvoinen variaatio. Paras luokitinkokonaisuus saavuttaa suuremman keskiarvotarkkuuden kuin mikään yksittäinen luokitin, ja vain yhden ympäristön tarkkuus kymmenestä on huonompi verrattuna kaikkien luokittimien ympäristötarkkuuksiin.

Avainsanat: Multimodaalisuus, Ympäristön tunnistus, Myöhäisyhdistäminen, Siirto-oppiminen

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

OpenL3	a Python library with L3-based pre-trained models
PyTorch	a Python library for machine learning
ResNet50	a pre-trained residual neural network with 50 layers
TAU	Tampere University
TensorFlow	a Python library for machine learning
TUNI	Tampere Universities
VGGish	a pre-trained audio neural network

1. INTRODUCTION

For us humans, identifying a real-world environment is a trivial task. The decision of which environment was detected is reached within a tenth of a second using our visual senses only [1]. This makes our sight a powerful sense for collecting data from our surroundings.

However, we also use other senses in determining our whereabouts. Hearing, smelling, and feeling the temperature all help with the identification. Often the decision is made with multiple senses, but sometimes just with a single one. Hearing becomes crucial when visibility is low, whether it is due to environmental (fog, dark) or medical (blindness) reasons. Separating environments with similar visual styles but with different soundscapes relies heavily on hearing. Public transportation vehicles, such as an airplane and a train, can have similar looking interiors, especially when crowded, but the sound is often different. We rely solely on visual senses when others cannot be used, for example, when looking outside a window. Even when a sense is temporarily unusable, we can still identify the environment.

Scene classification (recognition, identification) is a process in which a target area or environment consisting of multiple objects and events is categorized into a meaningful class or word, such as a kitchen or a beach. In machine learning, the number of scenes is limited by the number of annotated distinct classes used in the training dataset. By classifying a scene, an object or event in the scene gets context [2]. For example, a person screaming in an amusement park is less worrisome than in an alley.

Many machine learning tasks have started to use multimodal data. Even some scene classification systems have used RGB-D images which have color and depth images as their modalities. [3][4]. Regardless, the state-of-the-art scene recognition systems still rely on a single modality, either image [2] or audio [5]. A small study has been conducted on the performance difference of audio and visual modalities and the different methods of fusing them. [6]. However, the topic still requires more research for more general models.

When using multiple modalities, they have to be fused before the multimodal model's final output. Combining the different modalities early, rather than at the decision level, has been shown to give higher accuracy with correlated modalities [6][7]. However, combining the modalities late is more flexible, as then the model is not dependent on every modality. The model can still output decent results, even when a unimodal sensor breaks and no data is received for the corresponding modality.

The purpose of this thesis is to compare the accuracies of scene classifiers, which are based on either audio or visual modality, and of different combination methods in late fusion. Transfer learning is used to help create the unimodal neural network classifiers.

The remainder of this thesis is structured as follows: Chapter 2 presents the required background information on multimodality and transfer learning. The experiments' base model, which includes the unimodal classifiers and combiners, is introduced in 3. Chapter 4 contains the experiments, including the dataset, training methods and results. Finally, Chapter 5 presents the thesis' conclusions.

2. BACKGROUND

A modality is a way in which something is experienced or perceived. For humans, these modalities are analogous to senses, such as smell and hearing. A modality can be anything perceivable by a sensor or obtainable from a database [8], for example, visual, textual, or aural. Visual refers to sight, and therefore visual data includes different image types. Some modalities give better information than others, and selecting the correct modalities for a task is crucial [8].

Multimodality has been used in various tasks such as speech and emotion recognition, medical image analysis, multimedia event detection [9], biometric identification [10], and object detection [11]. One of the reasons for the increased use of multimodality is intuitiveness. For example, in medical image analysis, combining different image modalities, such as magnetic resonance imaging and photon emission tomography, improves feature visibility and can reveal information a human eye could not see [12]. Another reason is accessibility to multimodal data. Especially multimedia videos have become more accessible due to the possibility to record video and audio simultaneously with the current technology at any time. This accessibility also creates the need of handling multimodal data in different ways [9].

The rest of this chapter gives background information for methods that are often used in multimodal classification tasks. Section 2.1 shortly introduces various methods for fusing different modalities. Section 2.2 provides an overview of transfer learning and the idea behind pre-trained models. Although utilizing transfer learning is not necessary in a classification task, it reduces the training time and complexity of the used neural networks.

2.1 Fusion methods

In multimodal classification, a fusion method is needed to get a single result from an ensemble classifier. In this thesis, the result is handled as a prediction vector. The vector consists of prediction values for every class. A prediction value resembles a probability for a class, but the value is not limited to be a number between 0 and 1. The final output is obtained with the maximum membership rule, where the class with the highest value is chosen. Fusion methods mainly divide into two categories: early and late fusion [8].

Early fusion, also known as feature fusion, joins different modalities together during feature representation [7]. Here, features are defined as properties linked to the data, represented as numbers. In a classification network, the fusion usually takes places between unimodal feature extraction and classification. The combined features are represented by a single, multimodal feature vector. This vector can be used as an input to a sub-network, which makes the final classification decision. Early fusion can benefit from multimodal cross-correlations, in which an event's unimodal features are combined [7]. A semantic example of such a correlation is a video feed of a dog in one modality and barking sounds in the other modality. However, to utilize the correlation, the system requires that the features are correctly aligned in the feature vector [7]. Therefore, there has to be a same extraction interval for each modality. For example, 1 second of audio expects to have an image from that same second. Adding more modalities makes it difficult for the system to learn correlations [8].

Late (decision, semantic) fusion keeps the unimodal features apart and combines the results of the classifiers [7], creating an ensemble classifier. Late fusion is more flexible than early fusion as it can give decent results even if a modality is missing because the modalities do not depend on each other [9]. Late fusion also allows adding more base classifiers or even new modalities to the ensemble without having to train the other classifiers again. The downsides of such an ensemble classifier are the multiple training phases and its incapability to learn multimodal cross-correlations [7].

In late fusion, the results of the base classifiers are combined with a combiner. A combiner is often based on a combination rule, such as mean or median. The combiners used in the experiments are introduced later, in Chapter 3.

Other fusion methods also exist. These include data and hybrid fusion. Data fusion is sometimes considered to be early fusion, as it fuses the raw data. However, the data is rarely aligned, and pre-processing is needed [7]. Hybrid fusion is a mix of early and late fusion, utilizing both of them in a larger network [8]. It can benefit from the strengths of both methods, but requires extensive training. It also faces the same combiner selection problem as late fusion.

2.2 Transfer learning

In transfer learning, knowledge collected during a source task is utilized in solving a similar target task [13]. This reduces the need to re-collect knowledge during the target task. Machine learning benefits from transfer learning by using a neural network model trained for a generic source task to also perform on a target task without training the model from scratch [14]. These trained source models, referred to as pre-trained models, are deep neural network models, which have been trained on large datasets. Their architectures and parameters are publicly available, meant to be utilized with other projects. Some of the benefits of using pre-trained models are the reductions in overfitting, required computational resources, and training time [15].

A pre-trained model is often an early part of a large network where the pre-trained model handles the more complicated computations. This reduces the required training time, as a large part of the network is already trained. A pre-trained model can work either as a feature extractor or as a fine-tuned model.

As a feature extractor, the layer weights are frozen, so the model no longer trains. One or more of the final layers are removed to generalize the model from its source task. The extracted features are either low or high-level features, depending on the number of removed layers [16]. Particularly in simple classification tasks, a feature extracting pre-trained model is usually followed by classifying fully connected layers.

As a fine-tuned model, the final layers are either removed or changed, but the weights are not frozen. During back-propagation, the weights of the layers near the end change more than the earlier ones, adjusting to the high-level features. [16]

A pre-trained model should be used as a feature extractor if the target dataset is small or the network has a large number of parameters [14]. As a fine-tuned model, the network tends to overfit unless the dataset is large or the network is small. Fine-tuning a model is also computationally more expensive, as it requires an extra training session compared to a feature extractor. The experiments of this thesis utilize pre-trained models as feature extractors.

3. BASE MODEL

The goal of the experiments is to compare the scene classification accuracies obtained with unimodal and multimodal classifiers. A unimodal classifier utilizes either audio or visual modality. A multimodal classifier is an ensemble of the unimodal classifiers, which also contains a combiner. For analyzing the modalities, comparisons are also made on the scene level for the classifiers.

Classifiers based on different pre-trained models are created and trained for the experiments. There are two classifiers for both modalities, creating an ensemble classifier of four base classifiers. The layout of the base model consisting of these classifiers can be seen in Figure 3.1.

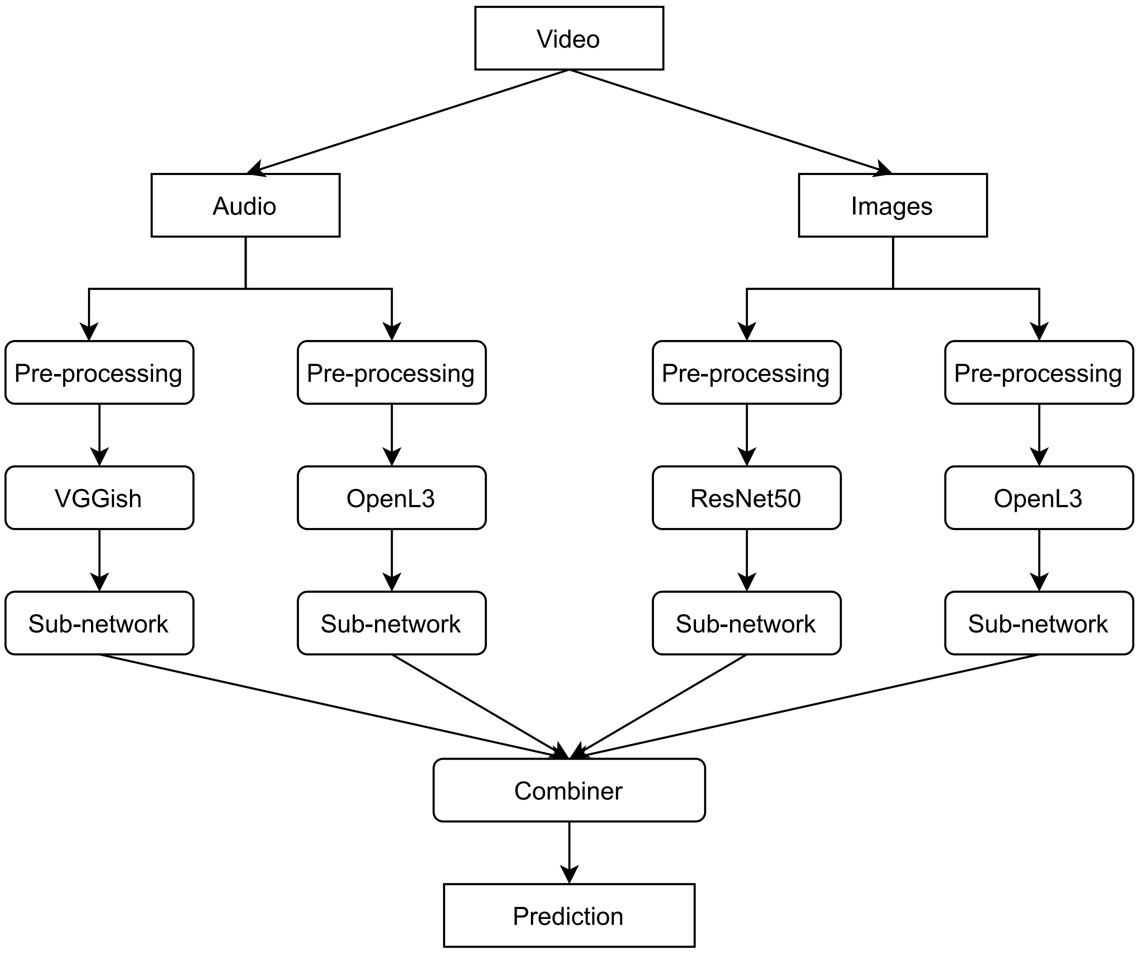


Figure 3.1. *Layout of the base model*

Each base classifier utilizes a pre-trained model as a feature extractor. The models have their own requirements due to their pre-training, and therefore the data needs to be pre-processed for each model separately. In a base classifier, a pre-trained model is followed by a small trainable fully connected sub-network which outputs a prediction vector. The final part of the ensemble network is the combiner, which outputs the final classification result.

The sections of this chapter present the base classifiers in more detail. The combiners of the experiments are also introduced.

3.1 VGGish

VGGish is a pre-trained model designed for audio feature extraction, but it can optionally be fine-tuned. VGGish was first introduced as a modified version of the convolutional neural network model VGG (configuration E) [17]. VGG, named after the development team's name (Visual Geometry Group), focuses on image processing [18], whereas VGGish on audio processing. VGGish was trained with the YouTube-100M dataset introduced in the same paper, and the model was shown to give good results.

The VGGish version used in one of the audio classifiers is a modified version of the VGG configuration A model [18], which has been pre-trained on the YouTube-8M dataset's [19] preliminary version [20]. The model is part of the TensorFlow Model Garden, which collects state-of-the-art models implemented with the TensorFlow machine learning framework [21]. The model consists of 6 convolutional layers and 3 fully connected layers. The output of the final layer is a 128-dimensional embedding [20].

VGGish expects the input to be a log mel spectrogram with 96 frames and 64 mel bands, resulting in an input size of 96 by 64 [20]. The VGGish developers provide a pre-processing code example for a single audio file. In this process, the audio is first converted to a mono-track and re-sampled to 16 kHz. Next, a spectrogram is computed utilizing Short-Time Fourier Transform magnitudes with a 25 ms windows size and 10 ms hop size. The spectrogram is transformed into a log mel spectrogram with 64 mel bins ranging from 125 to 7 500 Hertz. Finally, the spectrogram is framed into non-overlapping windows, resulting in the desired model input size. As each frame is 10 ms wide, VGGish processes the audio in 0.96-second segments. The model outputs a single embedding for every 960 ms in an audio clip, disregarding any residual audio.

The trainable sub-network, which trains on these embeddings, consists of 3 linear layers with 128, 256, and 10 nodes, respectively. The first two layers are activated with ReLU (Rectified Linear Unit) [22]. They are both followed by a Dropout layer with a 0.5 probability. The purpose of the dropout layer is to reduce overfitting by not allowing the nodes to rely on each other [23].

3.2 ResNet50

The selected pre-trained model for one of the visual classifiers is ResNet50. The 50-layered network utilizes residual connections to reduce the degradation issue of larger networks and allow faster convergence [24]. ResNet50 can be found on PyTorch's Torchvision module, which provides models that have been pre-trained on the ImageNet database [25] for addressing different visual tasks [26]. The model consists of an input convolutional layer, 4 unique blocks, or bottlenecks, and a fully connected layer. Each block has 3 convolutional layers. A block repeats a specific number of times, after which a next block in the sequence is used. The model totals 49 convolutional layers. As a feature extractor, the final layer is removed, and the average pooled output of the final convolutional layer is used. This results in a 2048-dimensional embedding for every input image.

The model expects its input image to have 3 color channels, containing values between 0.0 and 1.0 [26]. The image is also expected to be normalized using the ImageNet's mean and standard deviation. Both the width and height of the image has to be at least 224 pixels. Since the model expects an image instead of a video as the input, frame extraction has to be done for the original video. The pre-processing is then applied for these extracted frames, with available PyTorch transformation functions.

The base classifier's sub-network has two linear layers, with 64 and 10 nodes. The input layer uses batch normalization [27], followed by a ReLU activation. No Dropout layer is used with batch normalization, as proposed in the batch normalization paper [27].

3.3 OpenL3

OpenL3 is an open-source Python library consisting of audio and image feature extracting pre-trained models. It is based on the Look, Listen and Learn approach, in which audio and video clip correspondences are trained under self-supervision [28]. The approach led to two unimodal networks, which output well-performing feature embeddings [29]. More models were trained with similar architecture to allow the processing of different type of data, including different sound content and input representation. The ensemble model of this thesis utilizes OpenL3 in an audio classifier as well as in a visual classifier.

The OpenL3 models have been pre-trained on two different subsets of the AudioSet dataset [29]. The full AudioSet originally consisted of over 1.7 million 10-second YouTube videos with 632 different audio classes [30]. The music and environment subsets that OpenL3 trained with had 296K and 195K videos, respectively [29]. The available models follow the L3-Net architecture. They consist of 4 blocks, each having 2 convolutional layers, followed by a max pooling layer [28]. The differences of the unimodal sub-networks are the sizes of the inputs and layer parameters.

The library offers models with combinations of the following settings. The content type can be based either on the music or environment subset. With changes to the final max pooling layers of the unimodal systems, the dimensions of the feature embeddings can be either 512 or 6 144 for audio and 512 or 8 192 for images. The input audio can be represented either as a log linear spectrogram with 257 bands or as a log mel spectrogram with 128 or 256 bands. The image system also requires the audio's input representation, as the two systems were trained simultaneously. The library resizes the images to 244 by 244. The library handles the pre-processing automatically for the supported file types. The audio pre-processing also includes hop size and zero padding, which affect the number of embeddings obtained. [29]

Both audio and visual classifier use the embeddings generated by the environment subset, with 256 mel bands and an output dimension of 512. The audio uses a hop size of 1 second, without padding. These result in a 512-dimensional embedding for each second in audio and every image input.

The architecture of the audio classifier's sub-network is identical to that of the ResNet50 based classifier's sub-network. The only exception is the different input dimensions resulting from the feature extractions. The sub-network of the visual classifier consists of two linear layers, with 128 and 10 nodes. The input layer uses ReLU activation and a Dropout layer with a 0.5 probability.

3.4 Combiners

Different combiners are studied for the late fusion. The combiners expect one or more prediction vectors as their input, and they output a single prediction vector. The final class is obtained with the maximum membership rule. In the event of a tie, the first occurrence is selected, as implemented by PyTorch. In the experiments, the prediction vectors are in descending alphabetical order of the scene names. Therefore, the system prioritizes the classes in alphabetical order, but a tie is rare for most combiners.

Many combiners are based on trivial rules that come from intuition [31]. As these rules work without extra parameters, they are sometimes referred to as fixed rules [10][31]. By adding parameters, such as weights or classifier prioritizing, combiners can be designed for a specific ensemble model. In these case-specific combiners, adding, changing, or removing classifiers requires selecting new parameters. When the parameters are weights, they can be added to a classifier, a classifier's class probabilities, or a meta learner such as a neural network. These can further improve the system's accuracy when the impact of poor performing classifiers is reduced, and well-performing increased [31]. However, the accuracy can also decrease if the weights adjust the predictions poorly. The weights can either be trained or selected according to statistics.

3.4.1 Trivial combiners

A mean combiner takes the mean of class predictions given by the base classifiers. A sum combiner is the same as a mean combiner in this case, as the output vector has the values in the same descending order. In a trimmed version of the mean rule, sometimes referred to as jury rule, the most optimistic and pessimistic class predictions are not used when calculating the mean. A jury combiner requires at least three classifiers.

A maximum combiner selects the highest prediction value of each class to its output vector. If the classifiers' outputs are not normalized, the combiner selects the values of the most over-confident classifiers. A minimum combiner selects the lowest values instead.

A product combiner multiplies the prediction values of each class. If even a single value is negative for a class, the class's product will be zero or less.

A mode combiner is based on the mode rule and is often called majority voting. Unlike the previous combiners, the combiner uses the predicted labels obtained with the maximum membership rule. The label that occurs the most is selected. If there are multiple labels with the same number of occurrences, PyTorch implementation selects the smallest label. The combiner then outputs the label as a one-hot vector for the classification.

A median combiner utilizes the predicted labels similar to the mode combiner. The labels are sorted in ascending order, and the middle label is selected. When there is an even number of classifiers, there will be two labels in the middle. PyTorch selects the smaller label and outputs it as a one-hot vector.

3.4.2 Case-specific combiners

A simple combiner strategy is to add weights to the trivial rules according to the performance of the base classifiers. Each classifier gets a weight which equals to their accuracy on a validation set [32]. The weights are normalized by dividing with the sum of all the accuracies. Following this method, every classifier's class probability can also be given a performance weight. However, statistics regarding every classifiers' class performance is required.

As both the mode and median rules can end up with more than a single label, the decision over the final label has to be made. The classifiers can be prioritized when their validation accuracies are known. After the multiple labels are obtained, each classifier's predicted label is checked in the descending order of accuracy. The first predicted label that corresponds to one of the multiple labels is selected. The prioritized median does not differ from the original median if there is an odd number of classifiers.

A neural network can also work as a combiner, but it requires a separate training session. The model used in the experiments has two linear layers, with 32 and 10 nodes. Batch normalization is applied after the first linear layer, followed by ReLU.

There are also various other combiners that are not represented in this thesis. These include combiners based on custom-defined rules for a specific task and other classifiers such as support vector machine [8]. However, these are not part of the experiments due to time restrictions.

4. EXPERIMENTS

The experiments were implemented using Python, with PyTorch being the main machine learning framework. Two versions of TensorFlow were also used for extracting features with different pre-trained models. The implementations are publicly available on GitHub [1].

Section 4.1 introduces the dataset used in the experiments. The training and testing methods of the classifiers is explained in detail in Section 4.2. Section 4.3 gathers the results and analysis of all the classifiers.

4.1 Dataset

The dataset used in the experiments is a development set of the publicly available TAU Urban Audio-Visual Scenes 2021 [6]. It consists of approximately 10-second audio and video files, which were recorded in 10 different scenes. In total, there are up to 12 291 files. These scenes were recorded in multiples locations of multiple European cities, with 10 cities in the development set.

Of the 10 scenes, 3 are interiors of moving public transportation vehicles (bus, underground metro, tram), 3 indoor buildings (airport, metro station, shopping mall), and 4 outdoor urban areas (park, public square, pedestrian street, street with medium traffic). The number of passengers changes in the vehicles with the seats sometimes being empty and other times filled. The video is also filmed from different parts of the vehicles, sometimes only showing a small object such as a seat or a door. All the other scenes have a varying number of people moving and talking, with the park having the least people.

Both the audio and video were recorded simultaneously on different devices [6]. The audio has a sampling rate of 48 kHz and a bit resolution of 24 bits per sample. The video files were recorded in 1280 by 720 resolution and 29.97 frames per second. Face and registration plate blurring has been applied to the videos before the dataset's release. Any possible sound channels have been removed from the video files.

The dataset comes with a ready-made split for training and testing sets. They share the same scene classes and most of the cities, but the locations are unique. Tables 4.1 and 4.2 demonstrate the statistics of these two sets.

[1]https://github.com/tunasoup/multimodal-scene-classification

Table 4.1. *Training set statistics.*

scene	files	cities	locations	duration (min)
airport	697	8	21	113.2
bus	806	8	38	134.3
metro	705	8	37	117.5
metro station	893	9	32	148.8
park	1006	10	29	167.7
public square	982	10	29	163.7
shopping mall	841	8	22	140.2
street pedestrian	968	10	31	161.3
street traffic	985	10	30	164.2
tram	763	8	36	127.2
total unique	8646	10	305	1441.0

The training set has over 110 minutes worth of recordings for every scene, totaling 24 hours of training data. The scenes were not all recorded in each of the 10 cities. The average number of cities per scene is 8.9. A single city has files from 84 to 105 for a scene, making the distribution of cities close to equal. However, the number of recordings per location is not equal for the scenes. On average, a scene was recorded in a location from 19 to 38 times. On the city level, there are locations with only 6 recordings and one with 72 recordings. The vehicle interior scenes have fewer locations relative to file count as compared to the stationary scenes.

Table 4.2. *Testing set statistics.*

scene	files	cities	locations	duration (min)
airport	281	7	8	46.8
bus	327	8	15	54.5
metro	360	8	16	60.0
metro station	386	9	14	64.3
park	386	10	11	64.3
public square	387	10	12	64.5
shopping mall	387	9	9	64.5
street pedestrian	421	10	14	70.2
street traffic	402	10	12	67.0
tram	308	8	15	51.3
total unique	3645	10	126	607.5

The training:testing ratio is approximately 70:30. The testing data consists of 10 hours of data, and the average number of cities per scene is the same 8.9 as it was for the training data. On the scene level, most of the cities only have a recording from a single location, and the number of recordings per location is between 3 to 81. Both the shopping mall and tram scenes have one city which does not appear on the training data. Also, the airport and tram scenes have a city in the training set that does not appear in the testing set. The scene distribution varies due to the different number of scene recordings and the testing split requiring unique locations. In relative to their recorded size, 33.80 % of the metro data is in testing, and only 27.73 % of the park data, but both still have 386 files. Street pedestrian has the most files (421), whereas airport the least (281).

4.2 Training and testing

To examine and avoid overfitting, 15% of the initial training data is used as validation data. The split is done in the same way as for testing data, where the validation data does not share locations with the training data. However, the city importance was not taken into account in this split either. Consequently, the validation set has one city less for the metro scene than the training set. The training set contains 7 116 files after the split.

The 10-second audio clips are pre-processed according to the requirements of the pre-trained audio models. The pre-processed data is then input into the models. OpenL3 works on 1-second segments, generating ten 512-dimensional feature embeddings for each file. VGGish works on 0.96-second segments, which result in ten 128-dimensional embeddings. However, the final 400 ms of the 10-second audio clips are not used. The feature embeddings are saved, easing the training of the trainable sub-networks as the pre-trained models only need to be used once.

The videos were recorded with minimal humane movement [6], so the filmed backgrounds barely move. The foreground still has some moving objects, such as humans, cars, and breezy trees. For this reason, and to avoid long training, only 2 frames are extracted from every 10-second video clip. The frames are saved in 244 by 244 pixels. Similarly to the audio models, the images are pre-processed accordingly, input into the pre-trained image models, and their embeddings are saved. The embedding dimensions are 2048 and 512 for ResNet50 and OpenL3, respectively.

After the feature embeddings are saved, the sub-networks are trained separately. Their performances are evaluated with the mean loss of the validation data on every epoch. The weights that result in the lowest loss on the validation data are saved. The models initialize with the saved weights when their performance is tested on the test data.

Table 4.3. *Hyperparameters of the trainable neural networks.*

classifier	learning rate	batch size	epoch
audio VGGish	5.5×10^{-5}	32	30
audio OpenL3	8.5×10^{-6}	128	47
visual ResNet50	2.5×10^{-5}	32	26
visual OpenL3	2.5×10^{-5}	32	86

The unimodal classifiers are all trained with the Adam optimizer and cross-entropy loss function. The training data is shuffled on every epoch, and the training is stopped if the model has not obtained a better loss in eight epochs. The hyperparameters of the sub-networks are collected in Table 4.3.

The testing of the base classifiers is done on a file basis. This method differs from the training and validation phases, where each embedding corresponded to a single output. For example, as the VGGish based classifier outputs ten prediction vectors for each file, a mean of the values is taken. For each test file, the base classifiers' mean vectors are stacked and then saved with the ground truth labels. These stacked vectors are used as the input to the combiners.

However, the combiner inputs can also be modified. The vector values are initially logits, also known as log odds. A logit is obtained from a probability with the equation

$$logit(p) = \ln \frac{p}{1-p}, \tag{4.1}$$

where p is the probability. Logit sets a probability to real numbers, from negative infinity to positive infinity. A 0.5 probability equals to a logit with a value of 0. A probability can be obtained with the inverse equation

$$logit^{-1}(a) = \frac{\exp(a)}{\exp(a)+1}, \tag{4.2}$$

where a is a logit value. Both the logit and probability vectors can be normalized with the softmax function so that all the vector values sum to 1. This can prevent a single overconfident classifier from solely determining the final class with some combination methods. The equation for softmax is

$$softmax(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{K} \exp(x_j)}, \tag{4.3}$$

where x is a vector element and K the length of the vector. For logits, the resulting vector contains sum normalized odds. The logits, normalized odds, probabilities, and softmaxed probabilities are all tested as prediction types for the combiners.

4.3 Results

Every classifier's performance is reported as the average of the classifier's scene accuracies. This method takes into account the unequal distribution of the scene files as every scene is considered equal. If the total accuracy was used instead, a classifier with more bias towards street pedestrians than airports would give higher results with the test data.

The F_1-score is used as the metric for comparing the scene-specific performances in the unimodal classifiers. The F_1-score can be calculated with

$$F_1 = \frac{2tp}{2tp + fp + fn}, \tag{4.4}$$

where tp is true positive, fp false positive, and fn false negative. Unlike accuracy, the F_1-score uses false positives and does not give over-optimistic results for biased classifiers. If a classifier always chose the same class for every sample, it would gain the perfect accuracy for that class. However, the score for the class would be low.

4.3.1 Unimodal classifiers

The results of the unimodal classifiers can be seen in Table 4.4. The best-performing ResNet50 based visual classifier achieves an over 20% higher average accuracy than the worst-performing VGGish based classifier. However, the latter is still able to obtain the best score for the street traffic scene. The OpenL3 based audio classifier also achieves the best results for two scenes and outperforms its visual variant on average.

Table 4.4. *F_1 scores of the unimodal scene classifiers.*

	audio		visual	
scene	VGGish	OpenL3	ResNet50	OpenL3
airport	0.374	**0.702**	0.694	0.345
bus	0.557	0.741	**0.755**	0.715
metro	0.519	0.565	**0.881**	0.743
metro station	0.566	0.626	**0.951**	0.783
park	0.888	0.869	**0.894**	0.848
public square	0.533	0.570	**0.744**	0.522
shopping mall	0.599	0.673	**0.846**	0.582
street pedestrian	0.560	0.618	**0.823**	0.670
street traffic	**0.838**	0.821	0.809	0.740
tram	0.510	**0.659**	0.567	0.417
avg. F_1 score	0.594	0.684	**0.797**	0.637
avg. accuracy (%)	59.8	68.7	**79.8**	64.3

The classifiers based on the same modality perform almost in a relatively same manner, reaching most of their respective high and low scores on the same scenes. For example, both visual classifiers obtain a higher score than their average on the same six scenes and struggle particularly with the airport, public square, and tram scenes. The ResNet50 based classifier achieves a higher than average score with the shopping mall scene, but the OpenL3 based one does not. Similar correspondence can be seen with the audio classifiers, except for the airport and bus scenes, where the OpenL3 audio classifier is superior. The modalities also show their strengths on different scenes. When combined, the visual classifiers achieve a higher score for five scenes and the audio classifiers for two scenes. For the remaining three scenes, a similar score is obtained with both modalities.

The confusion matrices of the base classifiers are gathered in Appendix A. In a confusion matrix, a scene's false positives can be seen on the vertical axis and false negatives on the horizontal axis when the diagonal true positives are excluded.

Both audio classifiers confuse street pedestrians with public squares, which are aurally similar. The OpenL3 based audio classifier makes fewer mistakes overall and can distinguish the vehicle interiors rather well. However, it does confuse metros with metro stations. The VGGish based classifier has barely learned the differences of the different vehicle interiors and crowded scenes. However, it can distinguish the park and street traffic scenes without bias and just a small number of misclassifications.

The visual classifiers confuse the interiors of trams as those of buses'. The airports are also confused with shopping malls and public squares as street pedestrians. Overall, the ResNet50 based classifier confuses less, but due to its high bias towards the bus scene, it misclassifies the trams more often. As already noted in the introduction, the interiors of vehicles can be visually quite similar. Airports and shopping malls are also a bit similar, as they both have escalators and moving crowds. The airports also have shops, as these are airports in big European cities. Streets resemble public squares, but they are more narrow and densely populated by objects. Surprisingly, street traffic has a low confusion rate, considering that each extracted frame does not have a car in it.

On average, the classifiers utilizing the visual modality perform better. However, the pre-trained model has a major role in a classifier's performance, as the accuracies of the classifiers utilizing the same modality vary. Just like in a previous scene classification study using the same dataset [6], the OpenL3 library obtains a higher accuracy on the audio side. And even though the remaining VGGish based audio classifier is noticeably worse than the ResNet50 based one, ResNet is deeper than VGGish.

The classifiers could still be improved with various methods. These include fine-tuning VGGish and ResNet, and optimizing the sub-network architectures and hyperparameters. The results of the OpenL3 based classifiers could be improved by extracting increasing the number of embeddings for a file on the audio side [6], and possibly by extracting bigger embeddings. Also, the experiments used only two frames for each video, and there was no data augmentation. All these methods would require substantially more training.

4.3.2 Multimodal ensemble classifier

The results of the trivial combiners are shown in Table 4.5. These combiners do not have parameters, and in tie situations, the scenes are favored in alphabetical order.

Table 4.5. *Average scene accuracies of trivial combiners.*

prediction type	mean (%)	min (%)	max (%)	product (%)	jury (%)	mode (%)	median (%)
logits	**87.4**	83.5	79.3	2.2	85.4	81.4	78.8
normalized odds	86.0	86.4	83.7	**87.4**	84.4	81.4	78.8
probabilities	**86.9**	83.5	79.3	86.6	85.1	81.4	78.8
softmaxed probabilities	**87.0**	83.2	82.5	86.9	84.4	81.4	78.8

Almost all of the combiners perform better than any of the base classifiers with the mean combiner obtaining the highest accuracy using logits. Probabilities and the softmax variant also reach their highest accuracy with the mean combiner. Only the median combiner achieves a lower accuracy than the base classifiers with every prediction type and the max combiner with unnormalized types.

The poor performance of the product combiner using logits is due to the negative values given for below 0.5 probabilities. A single base classifier giving such a probability sets the product negative, and the confident classifiers set this value to the same level as the least likely classes. The mode and median are the same for each prediction type, as they work with the classifiers' predicted labels instead of the probabilities. The mean of logits has the same results as the product of normalized odds because their prediction values are in the same ranking order.

Table 4.6 gathers the results of case-specific combiners. Weights are added to the well-performing mean rule, and the mode and median rules are prioritized. The neural network combiner is trained on the same training data as the base classifiers. Training on the same data results in over-optimistic predictions. Optimally, the training data would be split into more training sets, and the combiner trained with the data that the base classifiers have not yet seen. Similarly to the base classifiers, the training is stopped if the model's loss has not improved in 8 epochs. Using the Adam optimizer, the learning rates for logits and normalized odds are 1.5×10^{-5} and 2.0×10^{-5}. Both probability types have a learning rate of 2.5×10^{-5}. The batch size is 32 for each.

Table 4.6. *Average scene accuracies of the case-specific combiners. The accuracy weighted mean (AWM) combiner weighs every classifier, whereas the F_1 weighted mean (FWM) combiner weighs every class of every classifier. The mode and median combiners prioritize well-performing base classifiers. The neural network (NN) is the only one of the combiners that is not based on a combination rule.*

prediction type	AWM (%)	FWM (%)	mode (%)	median (%)	NN (%)
logits	87.3	**87.4**	82.4	80.8	86.6
normalized odds	86.3	**86.6**	82.4	80.8	85.6
probabilities	86.9	**87.2**	82.4	80.8	86.4
softmaxed probabilities	86.8	**87.0**	82.4	80.8	86.6

Adding accuracy weights to the classifiers does not noticeably increase or decrease the accuracy of a mean combiner in these experiments. Weighing the class probabilities achieves the highest results of the evaluated case-specific combiners. However, this accuracy increase is not noticeable either. The neural network combiner performs slightly worse than the mean combiners. The trivial mean combiner and the class weighing variant both achieve the highest accuracy of 87.4% with logits, which is over 7% higher than the best unimodal classifier.

Prioritizing the mode and median improves the combiners, with the median combiner now surpassing the base classifiers. Consequently, all of the tested combiners outperformed the base classifiers. However, some of them required case-specific parameters or normalized inputs.

Excluding the combiners that require normalization, the accuracies of the combiners using different prediction types is small. The ranking of the prediction types may not be the same for a different model or dataset. The topic would require more research to determine if there is a notable difference when using the different prediction types.

The confusion matrix of the ensemble classifier using a trivial mean combiner can be seen in the figure 4.1. Compared to the base classifiers, the ensemble achieves higher F_1 scores for all the scenes except for the metro station. The ResNet50 based visual classifier obtains a score of 0.951, whereas the ensemble 0.928. The ensemble model has more true positives on the particular scene, but it misclassifies some of the airports as metro stations due to the intervention of the other base classifiers. The performance of the crowded scenes has improved, as only public squares suffer from being classified as street pedestrians.

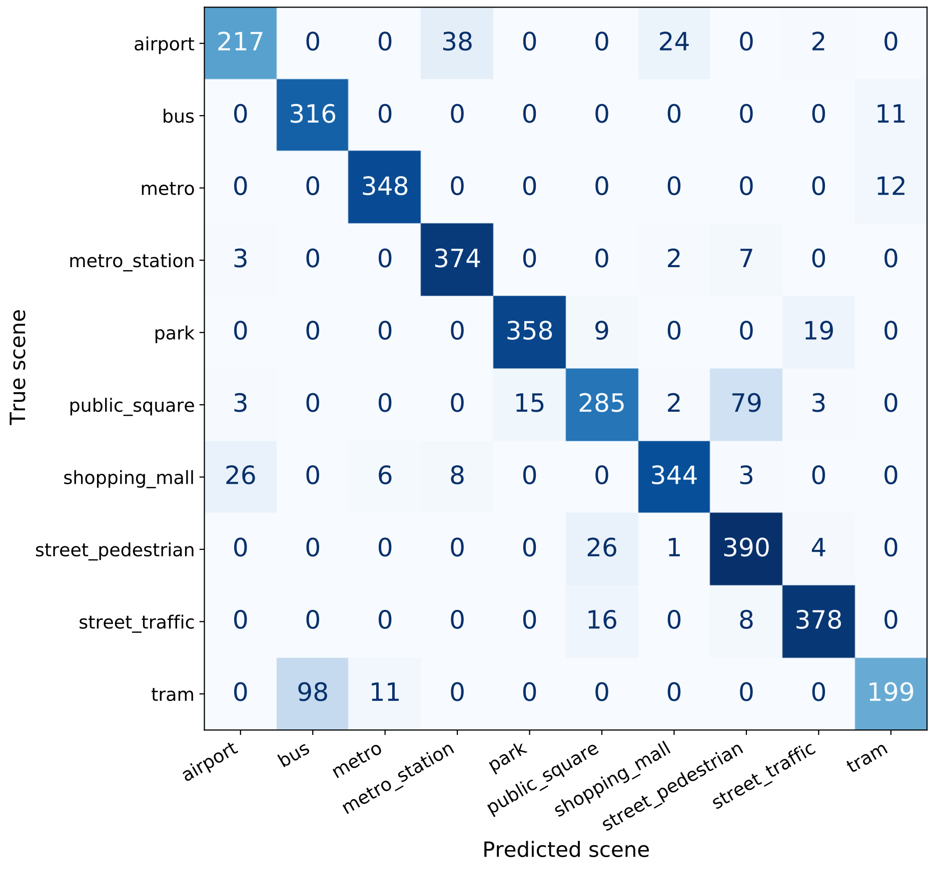


Figure 4.1. *Confusion matrix of the best ensemble classifier using a trivial mean combiner with logits.*

The best model still struggles with the tram scene and often classifies a tram as a bus. The model has fewer true positives on the tram scene as the VGGish based audio classifier, but unlike the base classifier, it does not have a bias towards it and achieves a higher score. However, the tram scene's score of 0.751 remains the lowest ensemble score, as no base classifier learned to recognize the scene well.

The best multimodal model represented in these experiments could still be improved. The combiner weights could be trained exhaustively, and the classifiers changed, removed, or even more added. However, to better evaluate the model's accuracy, the TAU Urban Audio-Visual Scenes 2021 evaluation dataset can be used for testing once it is published. Unlike the development set's testing data, the evaluation dataset consists of unseen cities instead of just unseen locations.

5. CONCLUSIONS

In this thesis, a multimodal ensemble neural network utilizing late fusion was built for scene classification. The ensemble classifier consisted of multiple unimodal base classifiers and a combiner that combines the classifier results. The unimodal classifiers were based on audio and visual modalities, with both modalities having two classifiers. Each base classifier used a pre-trained model as a feature extractor, with a small trainable subnetwork using the features to make the classifications. The results of all the classifiers and different combination methods were studied.

The experiments show that scene classifiers utilizing the same modality perform similarly, obtaining most of their highest and lowest scores on the same scenes. The individual strength of the modalities is also shown, as classifiers of both modalities perform better on certain scenes. However, the chosen pre-trained model has a clear impact on the classifier. In both modalities, the average accuracy of the other classifier is noticeably lower. In the experiments, utilizing a visual modality over audio is better on average, but this may not hold in general, as the architectures of the pre-trained models differ.

The tested combination methods were based on trivial rules and on more case-specific methods that utilized fixed parameters. Different input types were also tested for the combiners, including logits, probabilities, and their normalized versions. The input type's effect on the accuracy was only noticeable with combination rules that prefer normalization, such as the maximum rule. All the combiners were able to outperform the unimodal classifiers. However, some of them required adding parameters or normalizing the inputs. The best combination method was a simple mean of the base classifier outputs. The same accuracy was also reached with its class-weighted variant, where each classifier's class probability was adjusted with a weight.

The best multimodal ensemble model achieved an average accuracy of 87.4%, which is over 7% higher than the best unimodal classifier. The model also achieved better results on nine scenes out of ten when compared to all of the unimodal classifiers. The model could be improved further, but the study already shows that different combination methods can increase the classification accuracy of an audio-visual scene classifier.

REFERENCES

[1] Intraub, H. Rethinking visual scene perception. *Wiley Interdisciplinary Reviews: Cognitive Science* 3.1 (2012), pp. 117–127. ISSN: 1939-5078.

[2] Xie, L., Lee, F., Liu, L., Kotani, K. and Chen, Q. Scene recognition: A comprehensive survey. *Pattern Recognition* 102 (2020), pp. 107205–. ISSN: 0031-3203.

[3] Wang, A., Cai, J., Lu, J. and Cham, T.-J. Modality and component aware feature fusion for RGB-D scene classification. *2016 IEEE Conference on Computer Vision and Pattern Recognition* 2016- (2016), pp. 5995–6004. ISSN: 1063-6919.

[4] Zhang, H., Patel, V. M. and Chellappa, R. Hierarchical multimodal metric learning for multimodal classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition* 2017- (2017), pp. 2925–2933. ISSN: 1063-6919.

[5] Heittola, T., Mesaros, A. and Virtanen, T. Acoustic scene classification in DCASE 2020 Challenge: Generalization across devices and low complexity solutions. *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2020 Workshop* (Nov. 2020), pp. 56–60.

[6] Wang, S., Mesaros, A., Heittola, T. and Virtanen, T. A curated dataset of urban scenes for audio-visual scene analysis. *2021 IEEE International Conference on Acoustics, Speech and Signal Processing* (2021), pp. 626–630. DOI: 10.1109/ICASSP39728.2021.9415085.

[7] Gadzicki, K., Khamsehashari, R. and Zetzsche, C. Early vs Late Fusion in Multimodal Convolutional Neural Networks. *2020 IEEE 23rd International Conference on Information Fusion* (2020), pp. 1–6. DOI: 10.23919/FUSION45008.2020.9190246.

[8] Atrey, P. K., Hossain, M. A., El Saddik, A. and Kankanhalli, M. S. Multimodal fusion for multimedia analysis: A survey. *Multimedia Systems* 16.6 (2010), pp. 345–379. ISSN: 0942-4962.

[9] Baltrusaitis, T., Ahuja, C. and Morency, L.-P. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41.2 (2019), pp. 423–443. ISSN: 0162-8828.

[10] Gawande, U. and Golhar, Y. Biometric security system: A rigorous review of unimodal and multimodal biometrics techniques. *International Journal of Biometrics* 10 (Jan. 2018), p. 142. DOI: 10.1504/IJBM.2018.091629.

[11] Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W. and Dietmayer, K. Deep multi-modal object detection and semantic segmentation for autonomous driving: datasets, methods, and challenges. *IEEE*

Transactions on Intelligent Transportation Systems 22.3 (2021), pp. 1341–1360. ISSN: 1524-9050.

[12] James, A. P. and Dasarathy, B. V. Medical image fusion: A survey of the state of the art. *Information Fusion* 19.1 (2014), pp. 4–19. ISSN: 1566-2535.

[13] Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: `10.1109/TKDE.2009.191`.

[14] Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. How transferable are features in deep neural networks?: *Advances in Neural Information Processing Systems* 4.January (2014), pp. 3320–3328. ISSN: 1049-5258.

[15] Lei, H., Han, T., Zhou, F., Yu, Z., Qin, J., Elazab, A. and Lei, B. A deeply supervised residual network for HEp-2 cell classification via cross-modal transfer learning. *Pattern Recognition* 79 (2018), pp. 290–302. ISSN: 0031-3203.

[16] Mehdipour Ghazi, M., Yanikoglu, B. and Aptoula, E. Plant identification using deep neural networks via optimization of transfer learning parameters. *Neurocomputing (Amsterdam)* 235 (2017), pp. 228–235. ISSN: 0925-2312.

[17] Hershey, S., Chaudhuri, S., Ellis, D. P. W., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., Slaney, M., Weiss, R. J. and Wilson, K. CNN architectures for large-scale audio classification. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing* (2017), pp. 131–135. ISSN: 1520-6149.

[18] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations* (2015). arXiv: `1409.1556 [cs.CV]`.

[19] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B. and Vijayanarasimhan, S. YouTube-8M: A large-scale video classification benchmark. (2016). arXiv: `1609.08675 [cs.CV]`.

[20] Ellis, D., Heshey, S., Jesen, A. and Plakal, M. VGGish. *GitHub Repository* (Aug. 18, 2020). URL: `https://github.com/tensorflow/models/tree/master/research/audioset/vggish` (visited on 03/21/2020).

[21] Kim, J. and Li, J. *Introducing the model garden for TensorFlow 2*. Mar. 2020. URL: `https://blog.tensorflow.org/2020/03/introducing-model-garden-for-tensorflow-2.html` (visited on 03/21/2020).

[22] Nair, V. and Hinton, G. E. Rectified linear units improve Restricted Boltzmann machines. *27th International Conference on Machine Learning* (2010), pp. 807–814.

[23] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. ISSN: 1532-4435.

[24] He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image rsecognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778. ISSN: 1063-6919.

[25] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[26] *Torchvision models*. Dec. 2020. URL: `https://pytorch.org/vision/0.8/models.html` (visited on 03/21/2020).

[27] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning* 1 (2015), pp. 448–456.

[28] Arandjelovic, R. and Zisserman, A. Look, Listen and Learn. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 609–617. ISSN: 1550-5499.

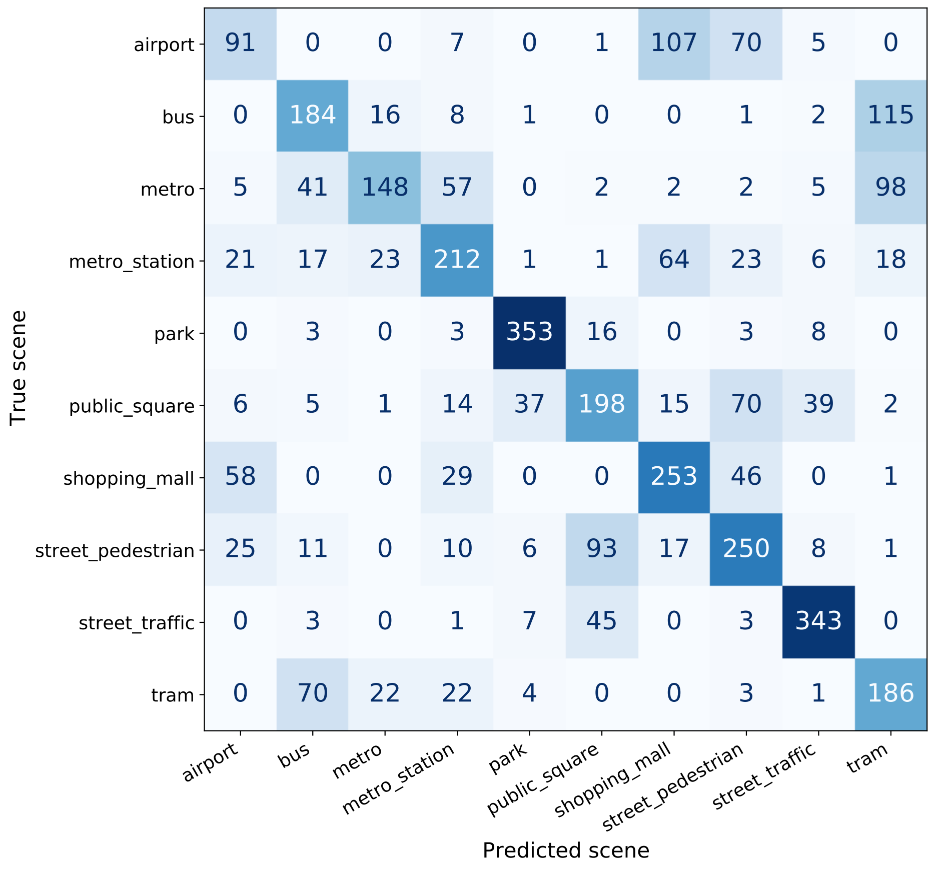
[29] Cramer, J., Wu, H.-H., Salamon, J. and Bello, J. P. Look, Listen, and Learn more: Design choices for deep audio embeddings. *2019 IEEE International Conference on Acoustics, Speech and Signal Processing* (2019), pp. 3852–3856. ISSN: 2379-190X.

[30] Gemmeke, J. F., Ellis, D. P. W., Freedman, D., Jansen, A., Lawrence, W., Moore, R. C., Plakal, M. and Ritter, M. Audio Set: An ontology and human-labeled dataset for audio events. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing* (2017).

[31] Kuncheva, L. I. *Combining pattern classifiers: Methods and algorithms*. 2nd ed. Hoboken, New Jersey: Wiley, 2014, pp. 111–185. ISBN: 1-118-91456-2.

[32] Rokach, L. *Pattern classification using ensemble methods*. Series in Machine Perception and Artificial Intelligence ; v. 75. Singapore ; World Scientific, 2010, p. 67. ISBN: 1-282-75785-7.

APPENDIX A: CONFUSION MATRICES



True scene \ Predicted scene	airport	bus	metro	metro_station	park	public_square	shopping_mall	street_pedestrian	street_traffic	tram
airport	91	0	0	7	0	1	107	70	5	0
bus	0	184	16	8	1	0	0	1	2	115
metro	5	41	148	57	0	2	2	2	5	98
metro_station	21	17	23	212	1	1	64	23	6	18
park	0	3	0	3	353	16	0	3	8	0
public_square	6	5	1	14	37	198	15	70	39	2
shopping_mall	58	0	0	29	0	0	253	46	0	1
street_pedestrian	25	11	0	10	6	93	17	250	8	1
street_traffic	0	3	0	1	7	45	0	3	343	0
tram	0	70	22	22	4	0	0	3	1	186

Figure A.1. *Confusion matrix of VGGish based audio classifier*

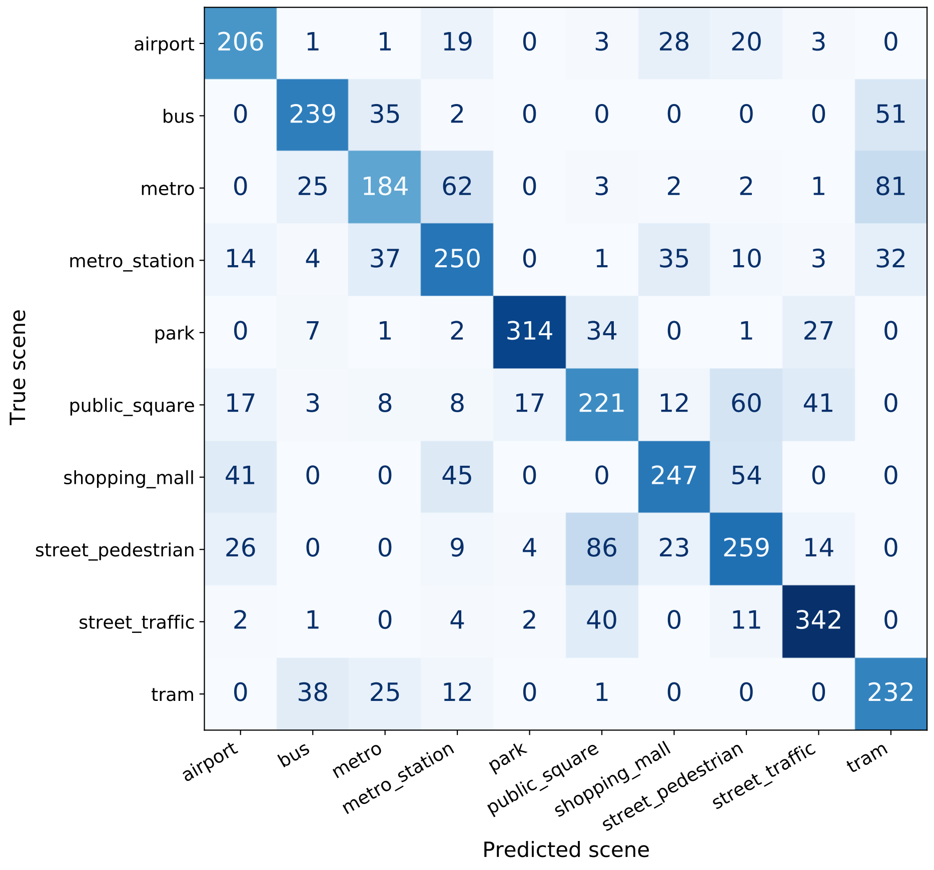
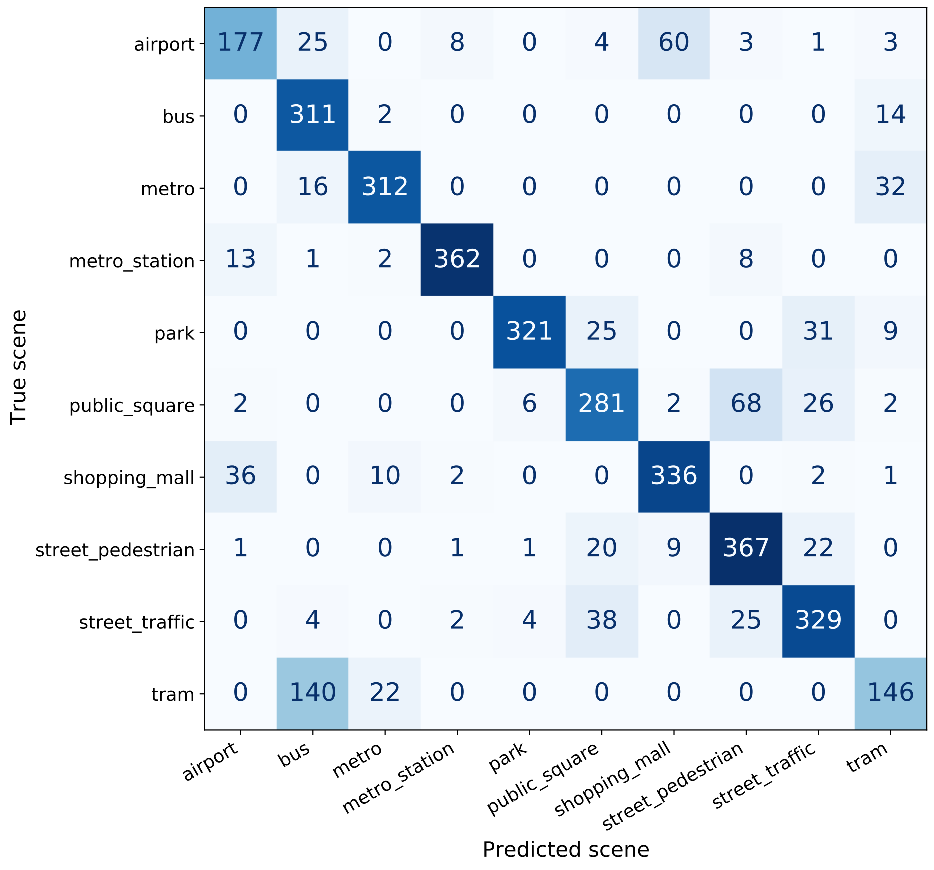
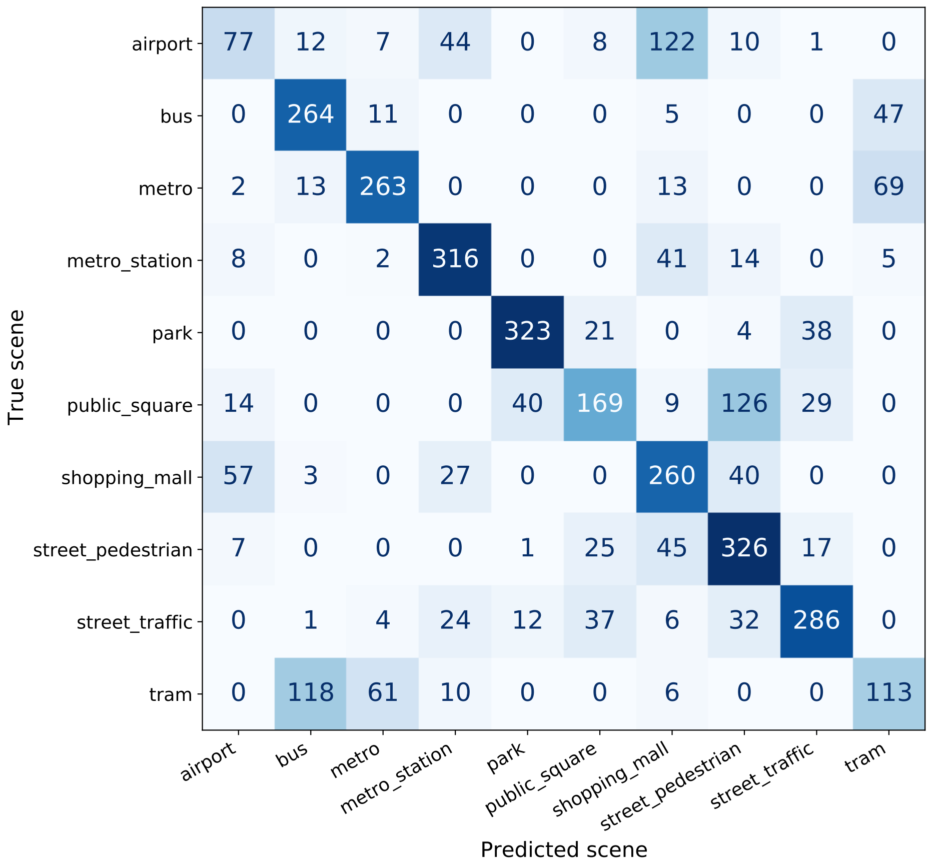


Figure A.2. *Confusion matrix of OpenL3 based audio classifier*
]



	airport	bus	metro	metro_station	park	public_square	shopping_mall	street_pedestrian	street_traffic	tram
airport	177	25	0	8	0	4	60	3	1	3
bus	0	311	2	0	0	0	0	0	0	14
metro	0	16	312	0	0	0	0	0	0	32
metro_station	13	1	2	362	0	0	0	8	0	0
park	0	0	0	0	321	25	0	0	31	9
public_square	2	0	0	0	6	281	2	68	26	2
shopping_mall	36	0	10	2	0	0	336	0	2	1
street_pedestrian	1	0	0	1	1	20	9	367	22	0
street_traffic	0	4	0	2	4	38	0	25	329	0
tram	0	140	22	0	0	0	0	0	0	146

Figure A.3. *Confusion matrix of ResNet50 based visual classifier*



True scene \ Predicted scene	airport	bus	metro	metro_station	park	public_square	shopping_mall	street_pedestrian	street_traffic	tram
airport	77	12	7	44	0	8	122	10	1	0
bus	0	264	11	0	0	0	5	0	0	47
metro	2	13	263	0	0	0	13	0	0	69
metro_station	8	0	2	316	0	0	41	14	0	5
park	0	0	0	0	323	21	0	4	38	0
public_square	14	0	0	0	40	169	9	126	29	0
shopping_mall	57	3	0	27	0	0	260	40	0	0
street_pedestrian	7	0	0	0	1	25	45	326	17	0
street_traffic	0	1	4	24	12	37	6	32	286	0
tram	0	118	61	10	0	0	6	0	0	113

Figure A.4. *Confusion matrix of OpenL3 based visual classifier*