Tampere University

Mikko Suominen

IMAGE CLASSIFICATION IN THE FOURIER DOMAIN

Bachelor's thesis
Faculty of Information Technology and Communication Sciences
Tarkastajat: Prof. Joni Kämäräinen
May 2024

ABSTRACT

Mikko Suominen: Image classification in the Fourier domain
Bachelor's thesis
Tampere University
Bachelor's Programme in Computing and Electrical Engineering
May 2024

Artificial intelligence (AI) has recently gained a lot of attention in the media. The main driving force for emerging AI tools such as ChatGPT has been deep learning enabled by modern computational hardware. Deep learning is based on artificial neural networks (ANN) which are computational models inspired by the structure of the brain. Image classification is a task that finds extensive use for neural networks.

Fourier transformation is a process that converts a signal into sinusoidal components. In this thesis, it is studied how transforming the images to the Fourier domain and optimizing an ANN for these transformed inputs affects the ANN's performance.

This study is conducted by studying the recent publications on this topic and examining the presented approaches to creating a neural network model for image classification in the Fourier domain. Then a total of 4 ANN models are defined and tested with a dataset containing images with 10 possible labels. The performance of each model with Fourier-transformed inputs is compared to the same model's performance without the transformation.

The study concludes that there is not enough evidence to support the usefulness of FFT in neural networks for this task, with classification accuracy as the evaluation metric. ANN's accuracy scores with Fourier-transformed inputs were consistently approximately 10% smaller than those with original inputs. However, the ability of ANN's to utilize transformed inputs without a significant performance drop suggests that these transformed images contain exploitable features. There remains a possibility that a more effective method for feature extraction exists.

Keywords: neural network, deep learning, artificial intelligence, computer vision

TIIVISTELMÄ

Mikko Suominen: Kuvien luokittelu Fourier tasossa
Kandidaatintutkielma
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Toukokuu 2024

Tekoäly (AI) on saanut viimeaikoina runsaasti huomiota mediassa. Syväoppiminen on modernin laskentatehon mahdollistama tekoälyn menetelmä, jolla on ollut merkittävä rooli edistyksellisien AI-työkalujen kuten ChatGPT:n kehityksessä. Syväoppiminen perustuu hermoverkkoihin (ANN), jotka ovat ihmisen aivoista inspiraatiota saaneita laskennallisia malleja. Kuvien luokittelu on tehtävä, jossa pystytään runsaasti hyödyntämään neuroverkkoja.

Fourier muunnoksella voidaan muuntaa signaali sen sinimuotoisin komponentteihin. Tässä työssä tutkitaan, miten kuvien muuntaminen Fourier tasoon ja neuroverkon optimoiminen näille muunnetuille syötteille vaikuttaa kyseisen verkon suorityuskykyyn.

Tämä työ toteutetaan tutkimalla viimeaikaisia julkaisuja aiheesta ja perehtymällä esitettyihin lähestymistapohin luoda neuroverkkomalli kuvien luokitteluun Fourier-tasossa. Sen jälkeen, määritetään ja testataan 4 neuroverkko mallia. Mallit koulutetaan ja testataan aineistolla, joka sisältää kuvia 10:stä mahdollisesta kategoriasta. Jokaisen mallin suorituskykyä Fourier muunnetuilla syötteillä verrataan saman mallin suorituskykyyn alkuperäisillä syötteillä.

Työn lopputuloksena ei ole riittävästi näyttöä, että FFT parantaisi neuroverkkojen luokittelutarkuutta. Tarkkuudet Fourier-muunnetuilla syötteillä olivat johdonmukaisesti noin 10% alhaisempia verrattuna tarkkuuksiin alkuperäisillä syötteillä. Kuitenkin neuroverkkojen kyky hyödyntää Fourier-muunnettuja kuvia ilman suurempaa pudotusta tarkkuudessa viittaa siihen, että muunnetut syötteet sisältävät hyödynnettäviä piirteitä. Ei ole poissuljettu, että toisenlaisia menetelmiä hyödyntymällä, muunnettuilla syötteilä voisi saavuttaa paremman tarkkuuden.

Avainsanat: neuroverkko, syväoppiminen, tekoäly, konenäkö

CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial intelligence
ANN	Artificial neural network
CNN	Convolutional neural network
FFT	Fast Fourier Transform
GPU	Graphical processing unit
MLP	Multilayer perceptron

1. INTRODUCTION

Artificial intelligence (AI) is a field in computer science that studies methods for a computer system or machine to complete tasks that require human intelligence. [1] AI has recently gained a lot of attention in the media and a notable contributor to this is a company called OpenAI, whose revolutionary products such as ChatGPT have demonstrated the vast potential of AI tools.

Computer vision is a specific subset within the domain of AI, dedicated to tasks related to visual perception, such as image analysis and video understanding. Computer vision is a field of AI that finds extensive real-world applications. It has applications in a wide range of fields such as medicine, automotive industry, production industries, consumer electronics, and surveillance. [2]

The most mundane example of a computer vision application would be Apple Face ID which is used to verify the user in Apple iPhones [3]. Other examples of computer vision applications would be detecting lung cancer from medical scans [4], quality assurance in the production industry [5], autofocus in camera systems [6] or gun detection in surveillance systems [7].

Image classification is a computer vision related task of categorizing images into predefined classes using a function that inputs an image and outputs the corresponding category label. Each image is assigned to exactly one of the predefined classes. The ANNs were originally developed mainly for this application. [2, p. 240, p. 349] In the deep learning approach, the function is called a model. The model has to be trained with a set of images with known labels. After sufficient training, the model can be used to predict labels for unlabeled images belonging to one of the predefined classes.

The model contains a fixed number of parameters which will be adjusted in the training process. Training begins with a model in which parameters are initialized based on some rule. [2, p. 283] A training sample is inputted into the model and the model outputs probabilities for each possible label. The model's parameters are then adjusted based on how far off this output was from the ideal output. The ideal output is, that the probability for the correct label is 1.0.

Ideally, when the process of inputting a training sample into the model and adjusting the parameters is repeated enough, the model learns to classify the training samples

correctly. The model should now have learned the features of the classes [2] and the model is ready to be used to classify the unseen data.

The process is visualized in the figure 1.1

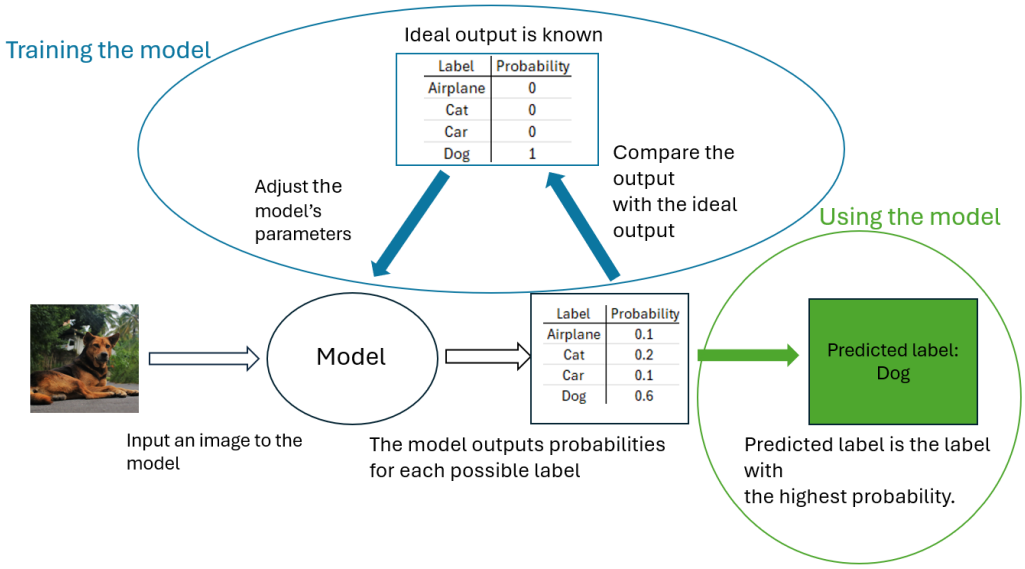


Figure 1.1. *Visualization of the processes within an ANN-based image classification system. In this example, the set of possible labels is {Airplane, Cat, Car, Dog}. The model can be improved by adjusting the parameters with labeled training images using the training process. After sufficient training, the model can be used to predict labels for unlabeled images.*

There has been recently published two bachelor theses which explored the feasibility of training artificial neural networks (ANN) originally designed for regular RGB images to classify Fourier-transformed versions of these images. The consensus from these studies was that convolutional neural networks (CNNs) struggle to excel in this particular task. [8] [9] This raises the question of what kind of architecture might be more suitable for image classification in the Fourier domain, or is it even possible to achieve similar performance as with the spatial domain architectures?

The Fourier transform is a linear operation used in many signal processing applications especially in telecommunications. As the transform does not lose any data, and the original signal can be retrieved with its inverse Fourier transform. Due to linearity, data preservation, and the ability of neural networks to fit any data, it makes sense that similar performance can be achieved with Fourier-transformed data as with normal images.

The objective of this thesis is to provide insights into addressing this question. In this thesis we attempt to find out what kind of elements would make the neural network work with Fourier images. The study is conducted by reviewing the literature on how Fourier-transformed images can be utilized with neural networks. Then in the experiments section four convolution-based architectures are trained and tested with both spatial and spectral

inputs. The results are compared to gain insight into how the Fourier-transform affects the network's performance.

Chapter 2 introduces architectures and computer vision applications that have been claimed to achieve promising results by utilizing the Fourier transformation. Chapter 3 introduces the basics of image classification with deep learning. Chapter 4 explains the test setup, Chapter 5 presents the results, and Chapter 6 concludes the study.

2. RELATED WORKS

The breakthrough in image classification using neural networks occurred in 2012 when AlexNet, developed by Krizhevsky et al. [10], secured first place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a top-5 error rate of 15.3%. This represented a remarkable improvement of over 10 percentage points compared to the second-best model [11]. Subsequently, convolutional neural networks (CNNs) remained the state-of-the-art method for computer vision tasks until 2020 when vision transformers were introduced. While vision transformers outperform CNNs in terms of accuracy, it's worth noting that CNNs remain a popular choice due to their ease of implementation.

Fourier analysis is a powerful tool for feature extraction, albeit with a specific set of applications. While it excels at capturing repetitive patterns, it loses the spatial relations [12] [13]. The Fourier transform offers the advantage of expediting specific tasks but presents challenges for others. For instance, in the context of medical images, a majority of the examples are negative, with only a few being positive. Healthy images tend to share many similarities, whereas the diseased samples may exhibit subtle differences [14]. These subtle differences are often hard to discern in the spatial domain, but they significantly impact the entire image when viewed in the frequency domain. Some research efforts leverage this property, such as the use of Fourier transformation in autoencoder anomaly detection [15] or removal of rain from images using Fourier transforms [12]. This has practical applications in fields like autonomous vehicles and surveillance.

There are some releases exploring the application of Fourier transform layers to optimize neural network efficiency by reducing the number of operations [16] [17] [18]. These methods aim to approximate convolutional neural network (CNN) operations in the Fourier domain, with a primary focus on utilizing the convolution theorem which states that convolution is equivalent to pointwise multiplication in the Fourier domain.

In the 2014 published article Fast Training of Convolutional Networks through FFTs [19], a faster training speed was achieved when the convolutional layer was replaced with FFT operation for both kernels and inputs, followed by pointwise multiplication, which was finally returned to spatial domain with inverse transformation.

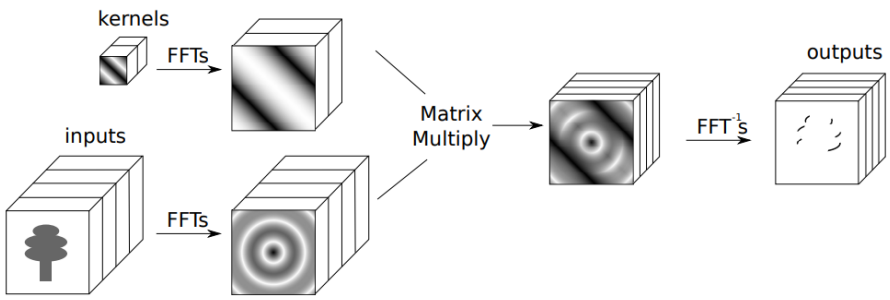


Figure 2.1. *Visualization of the spectral domain equivalent of the convolution. Unmodified figure from the original paper[19].*

This paper proves that the convolution theorem can be applied to neural networks. But this was not the ready solution to implement a backbone for spectral domain images yet since there was a multiple Fourier and inverse Fourier transformation throughout the process.

In 2017 Pratt et al. [16] introduced the Fourier Convolutional Neural Network (FCNN). Where it was claimed that the whole training process was implemented in the Fourier domain. The multiple switching to the spectral domain and back was avoided by performing a spectral domain equivalent max pool operation. The non-linearity layer which would have required the input to be in the spatial domain is possibly left out of the architecture since there was no mention of non-linearity in the paper. In addition, it was not shared how the input was flattened before the fully connected layers.

The model was tested in the paper with MNIST and CIFAR-10 datasets, but as the focus of the study was the computation time the test accuracies were not presented. The training accuracy in the training phase was over 90% for the MNIST dataset but less than 30% on CIFAR-10, which seems quite low. These accuracies could however be expected if the network didn't include any non-linearities.

In 2019, Ayet et al. released a paper Spectral-based convolutional neural networks without multiple spatial-frequency domain switchings [17] where they proposed spectral ReLU activation function, approximating spatial domain's ReLU in the spectral domain. The paper also introduced batch normalization adapted for the spectral domain. According to the paper, these findings would theoretically make creating a proper network working fully on a spectral domain possible. In the paper implementation however inverse Fourier transformation was applied before the fully connected layers thus the implementation was not in the Fourier domain from end to end. Similarly to the previous papers focus of this paper was to reduce the computation time, and it was not likely to be optimized for accuracy. The network still reached over 99% classification accuracy for the MNIST dataset, which is decent.

In 2022 András Fülöp and András Horváth released an article End-to-End Training of

Deep Neural Networks in the Fourier domain. [18] In the article they managed to substitute each element of the CNN with a suitable spectral solution and they claimed to be the first to introduce an approach where the whole training is implemented in the Fourier domain. To achieve this they used their nonlinear function which they called FReLU $f : \mathbb{C} \to \mathbb{C}$, was expressed as:

$$f(a+ib) = \begin{cases} a+ib & \text{if } \sqrt{a^2+b^2} > \alpha, \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

where α is a tunable parameter.

The maxpooling layer was substituted with a spectral pooling introduced in 2015 by Rippel et al. [20]. In the spectral pooling layer the input is transformed to the Fourier domain and shifted so that the DC component is in the center. The result is cropped maintaining the center and then inverted back to the spatial domain. This process is visualized in the figure 2.2.

Figure 2.2. *Spectral pooling visualization from the original source [20].*

The convolution layer was simply replaced with point-wise multiplication. Batch normalization wasn't included in this architecture. Before proceeding to flatten the input after the final convolutional layer and forwarding it to the fully connected layers, a magnitude function

$$f_{\text{abs}^2}(a+ib) = a^2 + b^2 \tag{2.2}$$

was applied.

This architecture was tested in the paper with the MNIST dataset and it's accuracy score was on average 91.93% while the time domain reference architecture scored 98.75%. It was concluded in the paper that the proposed architecture can achieve similar classification performance as the time-domain reference.

The paper presented impressive-sounding conclusion regarding Fourier backbone networks for image classification tasks. But this result should be interpreted cautiously. The complexity of MNIST dataset's handwritten numbers is much smaller when compared to natural images such as images of animals. There is no guarantee that the architecture would also perform at all on harder classification tasks. The MNIST dataset will also be tested in the experimental section and it will be shown that the over 98% test accuracy easily be achieved for Fourier-transformed presentation with a simple convolutional network designed for spatial images.

When the architecture was tested with the code shared with the paper [18] and CIFAR10-dataset containing 10 classes of natural images such as animals and vehicles, the training accuracy struggled to surpass 30%. This is a significantly weaker performance that would be achieved by simply inputting the Fourier images into a regular convolutional network. This will be discussed further in the results and analysis section.

It is worth mentioning that the focus of these architectures discussed in the related works section was to improve efficiency to enable the utilization of trained network classification capabilities on low-end hardware such as smartphones. The efficiency question was left out while studying these in this thesis.

3. IMAGE CLASSIFICATION WITH DEEP LEARNING

Deep learning is a method in machine learning that utilizes artificial neural networks (ANN). ANN's form a class of models constructed by combining various mathematical operations, both linear and non-linear [21]. This construction results in a function capable of fitting highly complex data [22].

3.1 Multilayer perceptron

The simplest ANN architechture is a multilayer perceptron (MLP), which consists of multiple nested perceptrons. Perceptron is a simple machine learning algorithm for binary classification invented in 1943 by Warren McCulloch [23]. Perceptron works by first taking a dot product with input vector and weight vector and then inserting the result to a sign function. Output will be either -1 or 1.

$$y_{pred} = 2sgn(\mathbf{x} \cdot \mathbf{w}) - 1 \tag{3.1}$$

This is visualized in the figure 3.1.

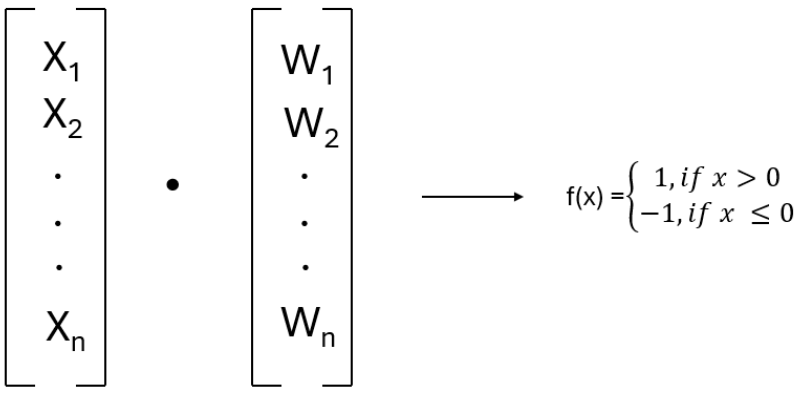


Figure 3.1. *Visualizing binary classification with the preceptor. The output of the preceptor is either 1 or -1 depending on whether the dot product is above zero or not.*

A perceptron can also be visualized with a graph as presented in figure 3.2.

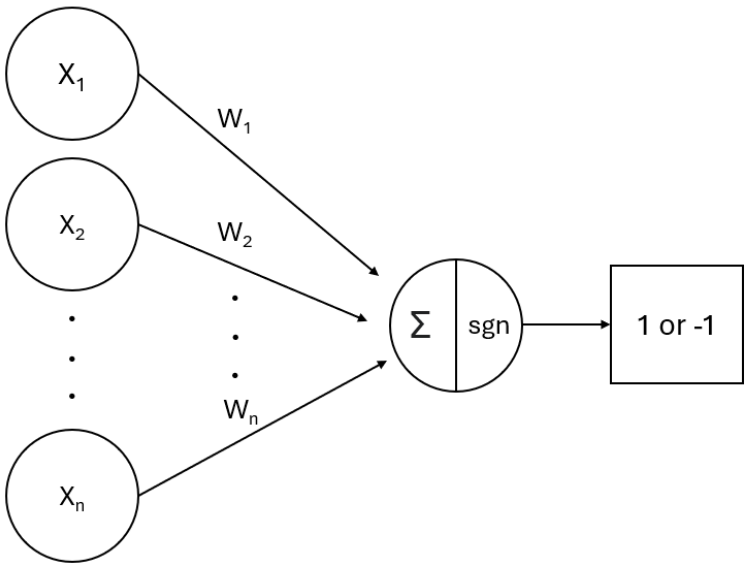


Figure 3.2. *Another visualization for binary classification with the preceptor. Graph visualization is clearer when visualizing nested perceptors.*

Before the perceptron can be used for classification, the suitable weight vector has to be found through the training process. For this, training data is required. Training the perceptron starts by first choosing arbitrary weights and then iteratively test the classifier with training data. If misclassification with the current weights is found, update the weights vector with the update rule.

$$w^{t+1} = w^t + lr \cdot (y_i x_i) \tag{3.2}$$

where lr is the learning rate, a tunable parameter that affects how drastically the weight is adjusted in each iteration.

This is very intuitive. For example, if the classification was falsely +1, the weight is too large and it will be reduced (y_i is -1 in this case).

The MLP can now be constructed by organizing multiple perceptrons hierarchically. A layer is a set of perceptrons that receive the same inputs. Each layer receives the inputs from the previous layer, and after processing, passes the input to the next layer. This is called nesting. MLP constructed by nesting perceptrons is visualized in figure 3.3.

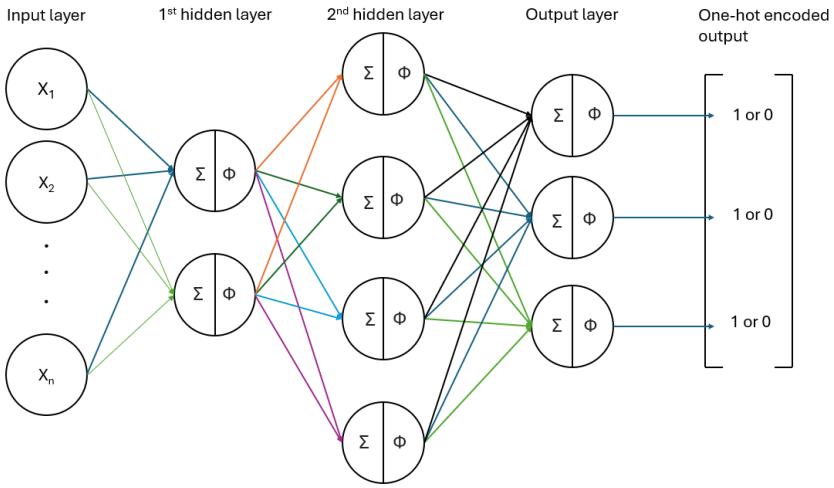


Figure 3.3. *Graph visualization of MLP.*

The function can be extended to fit more complex data and instead of binary classification, multi-class classification is possible. The number of perceptrons in the output layer determines the number of classes.

The output is now a vector of R^n dimensions, where n is the number of classes. The final label is one-hot encoded to this vector.

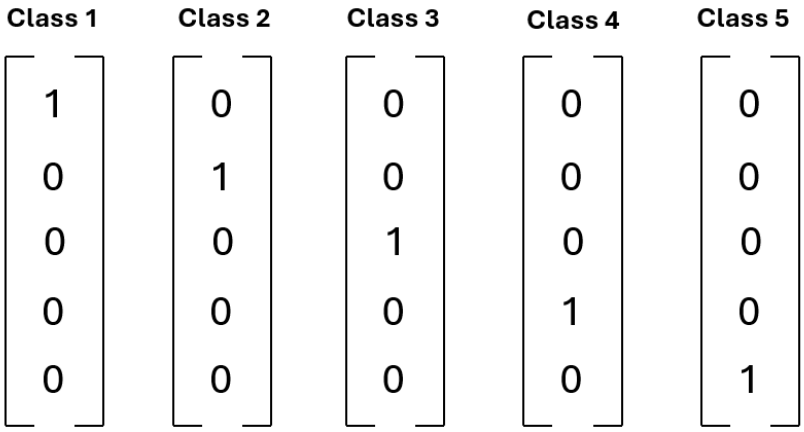


Figure 3.4. *One-hot encoded class labels.*

In a perfect scenario, there is exactly one non-zero element with a value of 1, which index tells which class the input belongs.

3.1.1 Training the MLP

Finding optimal parameters for the MLP weights is not as simple as training a single perceptron. When there is more than one perceptron it is no longer trivial which perceptron's weights cause the output to be incorrect. The MLP can be trained with gradient descent. In gradient descent in each training iteration partial derivate of the loss function in terms of the weight to be updated is calculated and the weight will be updated with the formula

$$w^{t+1} = w^t - lr \cdot \frac{\partial L}{\partial w^t} \tag{3.3}$$

To make this viable. sign activation function has to be replaced with some other non-linear function, because sign activation is not derivable in terms of the weights thus gradient based algorithms for training the network cannot be utilized. Rectified Linear Unit (RELU) has proven to be an effective choice for the activation function.

3.1.2 Classifying an image with MLP

The standard form for representing digital images involves three concatenated matrices, with each matrix representing a different color channel: red, green, and blue. This data structure is commonly referred to as a tensor. In image classification using a Multilayer Perceptron (MLP), the typical approach is to create a one-dimensional feature vector. This involves flattening the arrays of the color channels and appending them to the feature vector.

Process of classifying an image with a MLP is visualized in the figure 3.5.

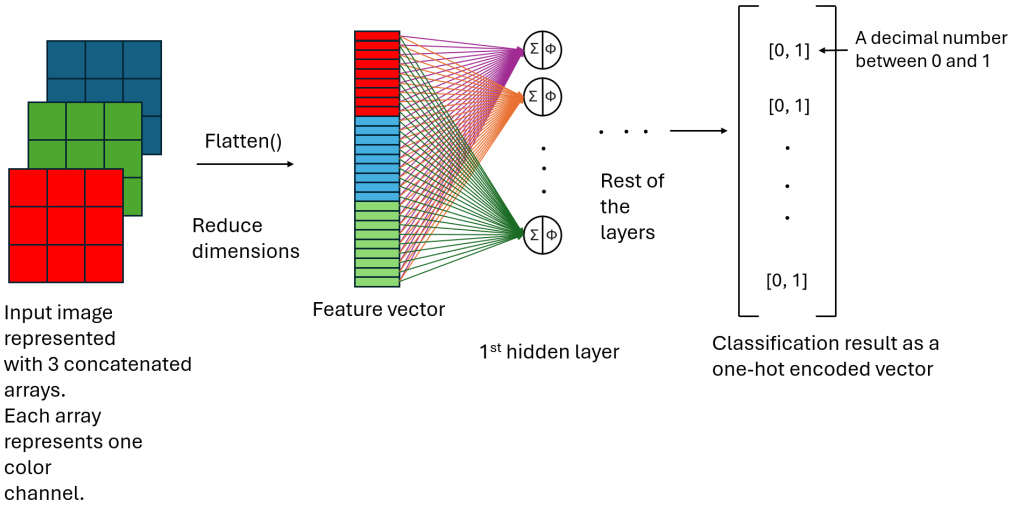


Figure 3.5. *Image classification pipeline with the MLP*

While constructing the feature vector by flattening the input tensor to 1-Dimensional vector

works for some degree, there are some major flaws with this approach. The spatial relationships between pixels are not taken into account. For instance, in the image of a cat, the spatial arrangement of distinctive features, such as ears, eyes, and tail carries a lot of useful information about the image. In addition, small irrelevant variations in the input, for example, moving the focused object to a different location within the image can make the model incapable of outputting the correct classification.

3.2 Convolutional Neural Network

Convolutional neural networks (CNN) solve the problems related to spatial relationships and shifted inputs. [21] Instead of creating the feature vector by simply flattening the input tensor, the feature vector is created by first filtering the input with learnable filters and then flattening the filtering result. These filters are 2-dimensional arrays called kernels. The filtering happens with convolution. Convolution for 2D matrices can be defined as [2]

$$g(i,j) = f * h = \sum_{k,l} f(i-k, j-l)h(k,l) \tag{3.4}$$

Kernels extract the relevant information inside the sliding window, where pixels are usually highly correlated. [21]

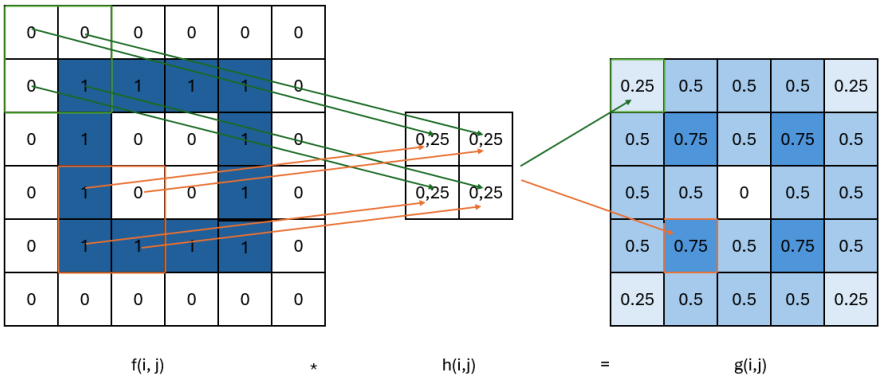


Figure 3.6. *Example of convolution operation. In this example the input is filtered with average filter resulting to a blurred version of the input.*

Like perceptors in MLP, multiple filters can be added in series or parallel. An example of CNN with three convolutional layers and three hidden layers is presented in figure 3.7.

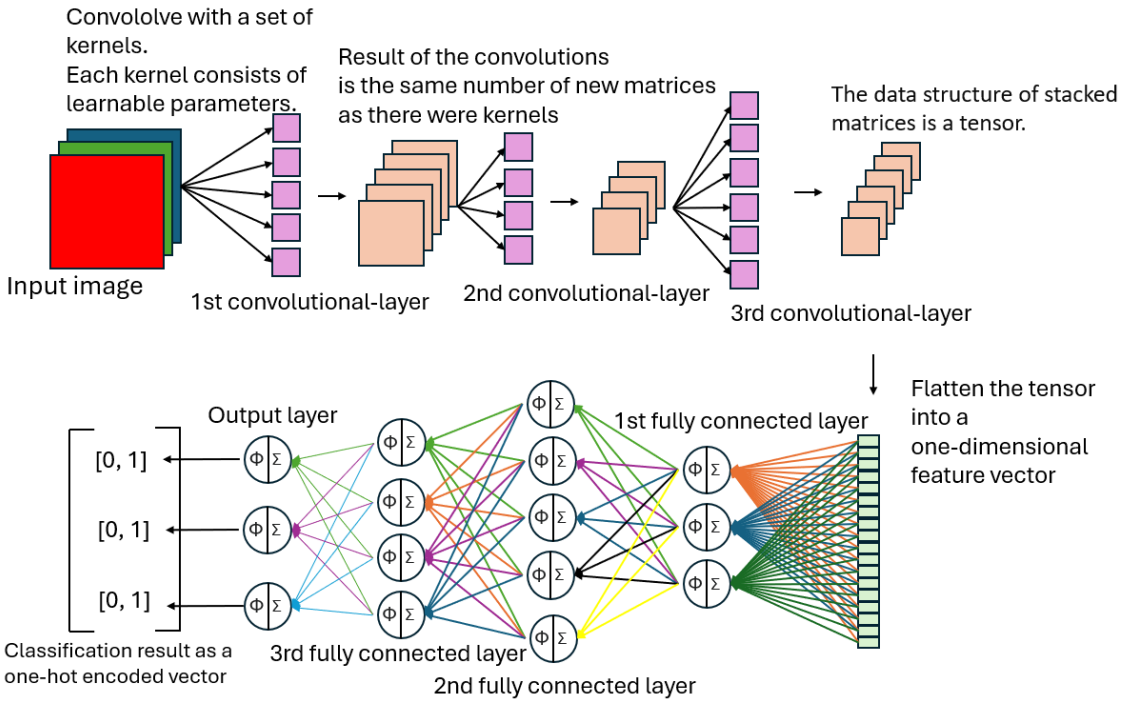


Figure 3.7. *A CNN with three convolutional layers and three hidden layers. The convolutions are used to create a feature vector which is then inputted to a structure similar to MLP.*

Max pooling

The max pooling method was first discussed in Riesenhuber's paper Hierarchical Models of Object Recognition in Cortex [24] released in 1999. In the paper the question of how some features could be robustly detected was approached from the neuroscientific standpoint. The authors claimed a MAX-like operation is applied in the visual cortex and is a key mechanism for object detection in the brain.

Max pooling is a common layer in the CNN where a square window is similarly slid across the input as in the convolution the convolutional kernel is slid. Now, the maximum value is picked instead of multiplying and summing the elements inside the window. This can be expressed as

$$f_{maxpool}(i,j) = \max_{k,l} f(i-k, j-l) \tag{3.5}$$

The benefit of the max pool layer is that it removes the less distinctive features from the representation, reducing the size of the tensor passed to the next convolutional layer. This ultimately reduces the number of trainable parameters making the training more efficient. Max pooling also improves the network shift invariancy and reduces the risk of overfitting [2].

4. EXPERIMENTAL SETUP

The tests were conducted with the Ubuntu 22.04.4 operating system equipped with NVIDIA RTX4060Ti GPU. The code was written with Python 3.11 and TensorFlow 2.16 library with Keras interface. All the code used in this study is available in my GitHub repository[1].

For the experiments, two datasets were utilized, namely CIFAR-10 and MNIST. CIFAR-10 [25] contains 60000 colored images split into 10 classes: (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The dataset is ready split to 10000 test and 50000 training images, and there is 10000 training images for each class. The resolution of the images is 32x32. The primary focus of this study was the results acquired with the CIFAR-10 dataset.

MNIST dataset [26] contains 70000 grey-scale images of handwritten digits ranging from 0 to 9. 10000 images are set for testing and 60000 for training. The complexity of the MNIST dataset is significantly lower than the CIFAR-10:s as the images are in black and white format and smaller 28x28 resolution. Digits can also be expected to be more distinctive from each other when compared to cats and dogs for example. The main purpose for testing MNIST dataset was to gain understanding of the other studies also conducted with MNIST dataset.

Two different main architectures are tested. The first is a simple convolutional network with 130 thousand parameters and the second is a more sophisticated Xception network with 2.7 million parameters. The modified versions of these, where the augmentation layer is placed before the input layers are also tested making the total number of architectures 4. Each architecture is tested with both original and Fourier-transformed CIFAR-10 datasets making the total number of CIFAR-10 models 8.

All models are trained for 120 epochs using the Adam optimizer and a learning rate scheduler. The learning rate is adjusted to 10^-5 after 72 epochs, and to 10^-6 after 96 epochs. Validation accuracy is tested after each epoch. Training data is split into training and validation sets with an 80-20 split.

[1] `https://github.com/mikkosuo/convolution-on-fourier-images`

Simple convolutional network

The simple neural network constructs from 3 convolutional layers followed by 3 fully connected layers. The structure of the simple convolutional neural network is presented in table 4.1.

Table 4.1. *List of layers and the input and output shapes for 32x32x3 shaped input. Convolution and max-pool layers are colored, as they are key components of the architecture.*

Layer index	Name	Input shape	Output shape
1	Input	32x32x3	32x32x3
2	BatchNormalization	*doesn't affect shape*	
3	Convolution (32@3x3)	32x32x3	30x30x32
4	Max-Pool	30x30x32	15x15x32
5	BatchNormalization	*doesn't affect shape*	
6	Convolution (64@3x3)	15x15x32	13x13x64
7	Max-Pool	13x13x64	6x6x64
8	BatchNormalization	*doesn't affect shape*	
9	Convolution (128@3x3)	6x6x64	4x4x128
10	Max-Pool	4x4x128	2x2x128
11	Flatten	2x2x128	512
12	Dense (64)	512	64
13	Dense (32)	64	32
14	Dense (10) [Output-layer]	32	10

This is visualized in the figure 4.1.

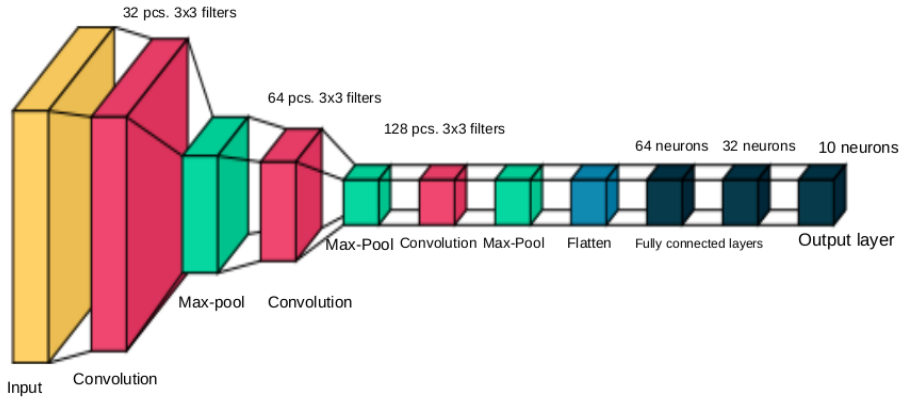


Figure 4.1. *Visualization of the simple convolutional neural network.*

Total number of parameters is 130 thousand.

Xception

Xception is an architecture published 2016 by Francois Chollet [27], the creator of the Keras library. A smaller version of this architecture [28] is selected to represent more complex architecture in this study because it is fairly simple to implement and the implementation is available in the keras documentation [28].

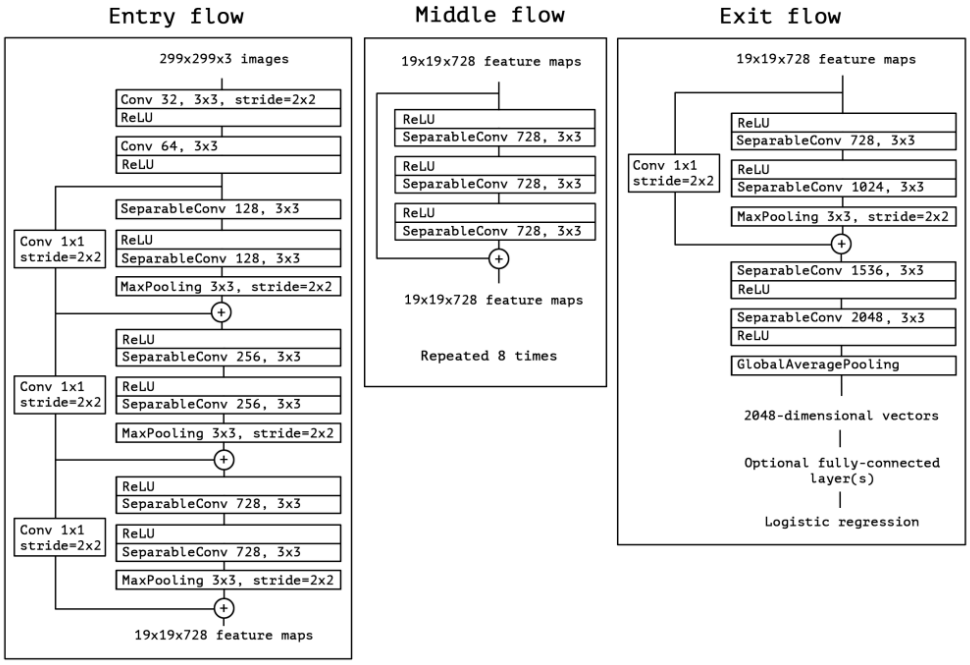


Figure 4.2. *Xception architecture. Figure from the source [27].*

Xception is based on depthwise separable convolutions and residual connections also known as skip connections. Depthwise separable convolutions extract features with lesser computations.

Skip connections enable the reuse of the earlier layers' features in the deeper layers. Residual connection makes the training easier since there are multiple paths for the information to flow. Skip connections also improve generalization by ensembling. [29]

The total number of trainable parameters in the smaller version of Xception is 2.7 million which is over 20 times more than in the simple convolutional neural network.

Data augmentation

The test accuracy, which indicates the network's ability to generalize, can be significantly lower than the training accuracy. This is called overfitting which means that the model has learned training set specific features that do not apply to all possible instances of the

class.

Reasons for overfitting could be that the training set is too small making it hard for the model to differentiate between irrelevant elements and relevant features in the image. In addition, even if the training set's size seems sufficient, the training examples could not be distributed properly to present the whole class. For example, if we train a binary classifier for cats and dogs, and in the training set, all cats face left and all dogs face right, the classifier will struggle with a test image of a dog facing left.

Overfitting can be lessened by creating new artificial training examples from the existing training data, for example, by applying geometric and color transformations, cropping, rotating, or flipping [30]. These techniques can be called dataset augmentation. [2]

Figure 4.3 visualizes the flip-and-rotate augmentation used in the augmented versions of the simple convolutional network and the Xception architectures.

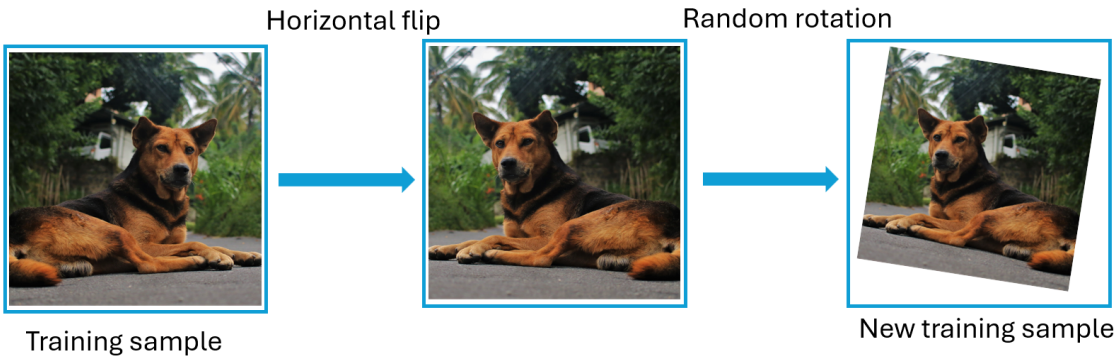


Figure 4.3. *Example of creating a new training sample with flip-and-rotate augmentation technique.*

Augmentation can be applied directly to the dataset, or there could be an augmentation layer before the network's input layer. In the experiments section, the augmentation is implemented by adding an augmentation layer directly after the input layer. In the augmentation layer, the input is flipped with 50% possibility and rotated a random amount of degrees in the range $[-11, 11]$.

2D-discrete Fourier transformation

Fourier transform can be used to analyze the frequency spectrum of an image. Two-dimensional Fourier transformation in discrete domain is defined as

$$H(k_x, k_y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) e^{-j2\pi \left(\frac{k_x x}{M} + \frac{k_y y}{N} \right)} \tag{4.1}$$

where M and N are the width and height of the image. [2]

2D-Fourier transformation is implemented using tensorflow's libraries. The transformed image cannot be directly used as an input to the ANN because it contains complex values that are not supported by the Keras' interface's input layer. To overcome this, the real and imaginary parts are concatenated. The size of the last dimension increases from 3 to 6. This is visualized in figure 4.4.

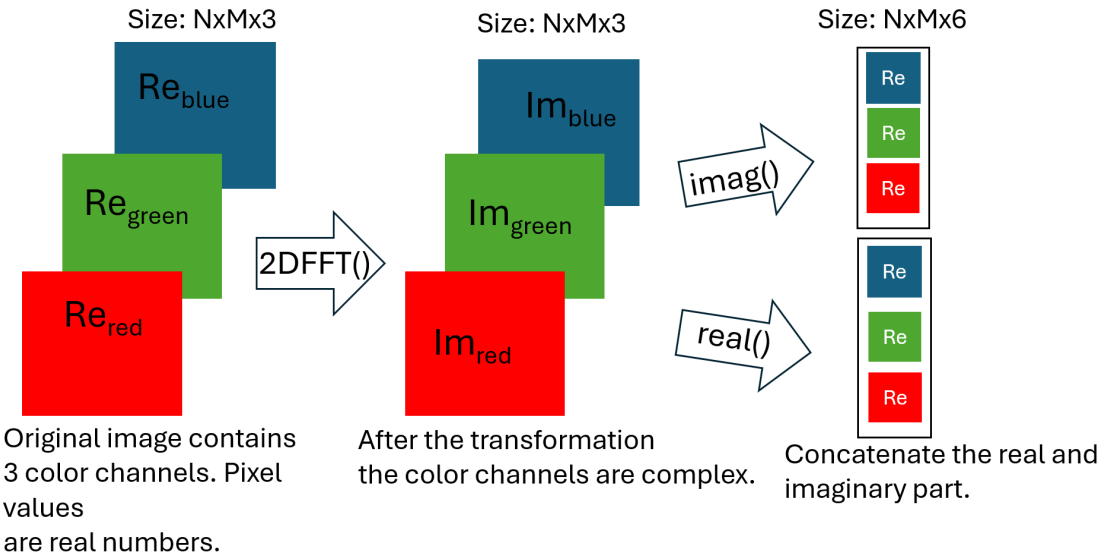


Figure 4.4. *Applying the Fourier transform to the input. The imaginary parts are concatenated along a new dimension since ANNs created with Keras interface do not support complex tensors.*

Implementation can also be understood through a single line of code:

```
X_train = tf.concat([real(fft2d(X_train)), imag(fft2d(X_train))], axis=3)
```

which was used to achieve the Fourier-transformed input for the ANN.

5. RESULS AND ANALYSIS

This section is divided into three parts. First, the training results will be analyzed, followed by a test result analysis. Then finally, the findings will be discussed.

In total, 16 different models were trained, 8 for the CIFAR-10 dataset and 8 for the MNIST. The main focus will be on the CIFAR-10 dataset. With the CIFAR-10 dataset, on the same architecture, the accuracy scores were consistently around 10 % lower on the spectral set compared to the spatial.

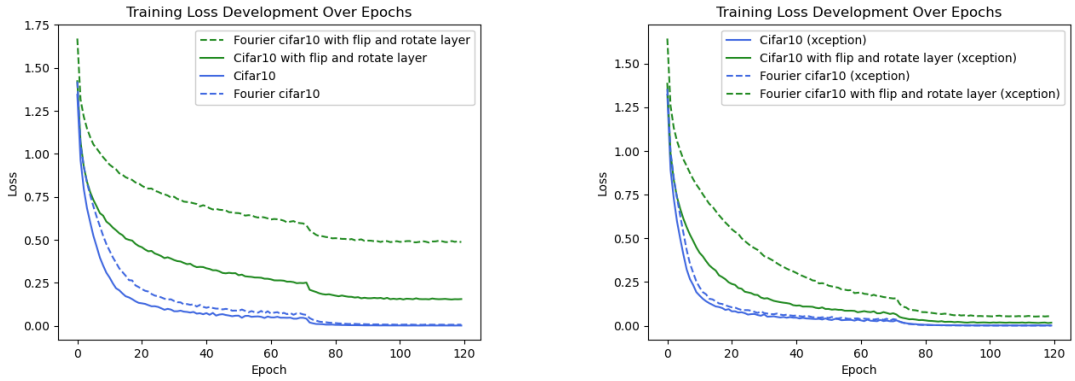
The MNIST dataset was tested only for reference to understand the scores in the related works section's papers. There was not much to compare, as the test accuracy of all networks reached the 98-99 % range.

5.1 Training results

Both simple convolution and Xception architectures were able to learn both spatial and spectral images. On both architectures, the training losses and accuracies are close to one another when there is no flip and rotation augmentation. On the versions with an augmentation layer, the spatial images got noticeably better training accuracies and losses compared to the spectral images.

There are no differences in the two architectures' training behavior on the versions without the augmentation layer. On the versions with the augmentation layer Xception's training accuracies eventually got close to 1.0 whereas the simple convolution network struggled. From this, it can be concluded that the more sophisticated Xception network is more invariant to flipping and rotation compared to the simple convolution network.

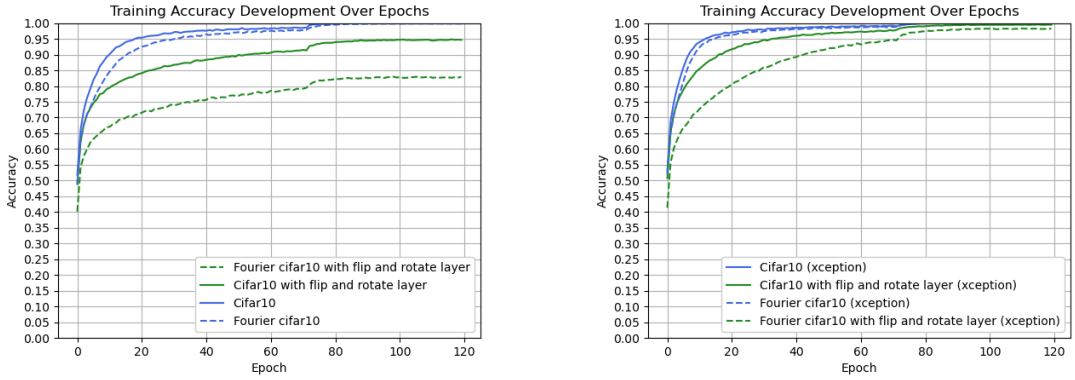
In the next three figures visualizing the training process, dashed lines represent the spectral networks and solid lines represent the spatial networks. Augmented versions are marked with green and non-augmented with blue.



Simple convolution *Xception*

Figure 5.1. *Side-by-side comparison of simple convolution and Xception networks' training losses. The difference is, that the loss in the Xception's case got closer to zero on the augmented networks (green lines).*

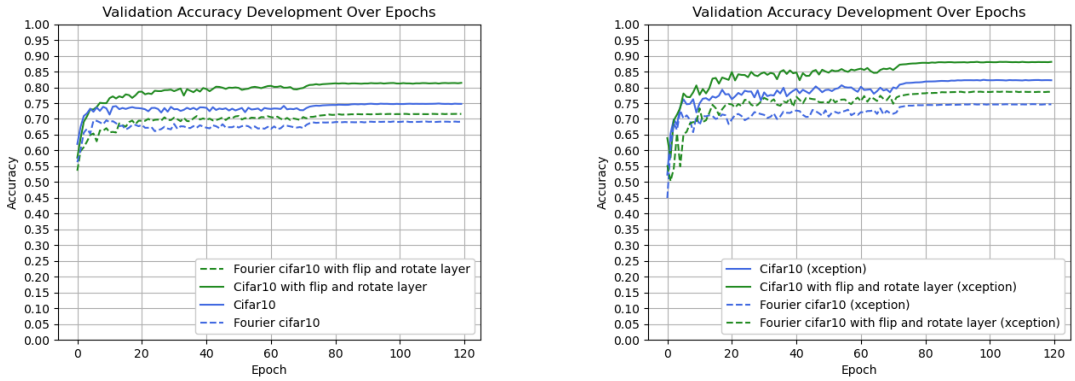
Similar phenomena is observable when examining the development of the training accuracy over epochs. In the case of the spectral images, the flip-and-rotate layer-equipped network was never able to come close to the one without that layer.



Simple convolution *Xception*

Figure 5.2. *Side-by-side comparison of simple convolution and Xception networks' training accuracies. The Xception network achieved higher training accuracies when comparing the augmented versions (green lines).*

Validation accuracy gives a picture, of what kind of accuracies can be expected with the unseen test data. The validation plots were similar in both the spectral and spatial cases. The only difference is that the spectral network's accuracy is consistently slightly lower and the augmentation didn't yield as much improvement.

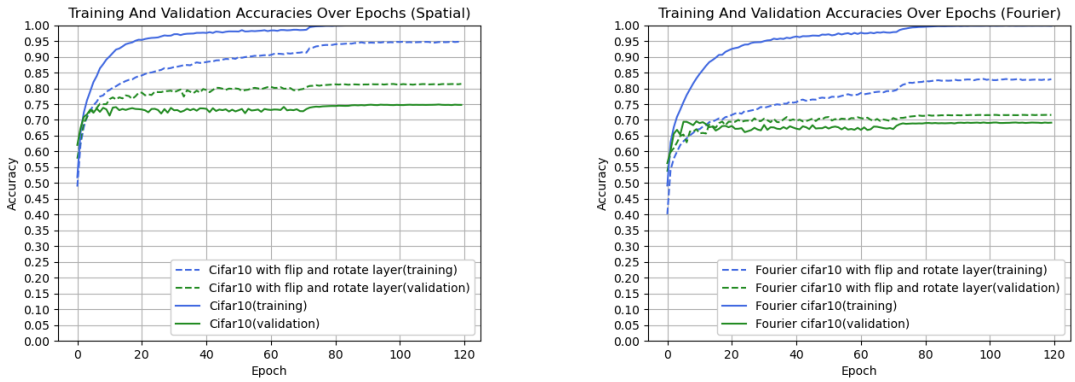


Simple convolution *Xception*

Figure 5.3. *Side-by-side comparison of simple convolution and Xception networks' validation accuracies. Xception's accuracies are slightly higher.*

The training results can be summarized with a side-by-side comparison of the spatial and spectral versions of the same base architecture. Contrary to the previous figures, all training accuracies are marked with blue lines, and the validation accuracies with green. Augmented networks are marked with dashing.

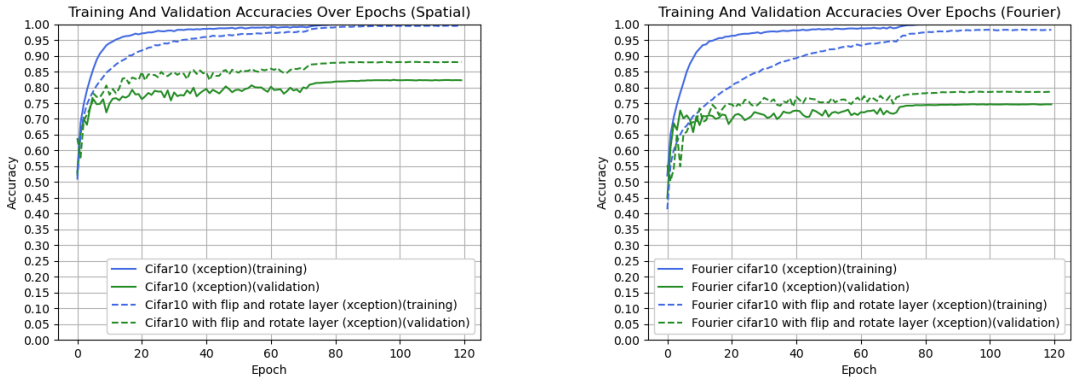
Figure 5.4 presents the simple convolution network's training and validation accuracies.



Spatial (simple conv.) *Spectral (simple conv.)*

Figure 5.4. *Comparison of simple convolution's spatial and spectral networks training and validation accuracies. Plots are similar with the exception that the spectral's validation accuracies are slightly lower.*

Figure 5.5 presents the Xception's training and validation accuracies.



Spatial (Xception) *Spectral (Xception)*

Figure 5.5. *Comparison of spatial and spectral networks training and validation accuracies. Plots are similar with the exception that the spectral's validation accuracies are slightly lower.*

In conclusion, simple convolution and Xception behaved quite similarly. The only difference was that the Xception reached noticeably higher training accuracies than the simple CNN when comparing the versions with the augmentation layer. Validation accuracy on the spectral images was consistently lower compared to the same network with spatial inputs.

MNIST results for the reference

Both the spatial and spectral versions of the MNIST network reached over 98% over validation accuracy during the training on the simple convolution base architecture. Spatial version was slightly better but the difference is less than 1%.

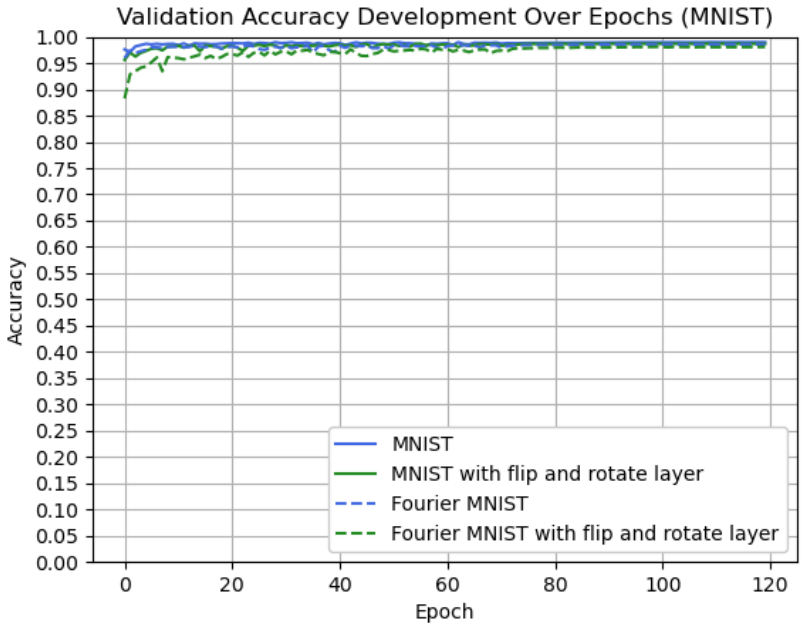


Figure 5.6. *Validation accuracies of the simple convolutional network during training on the MNIST dataset.*

With this result, it can be concluded that when the complexity of the dataset is low, it makes no difference for the standard CNN whether the data is in the spatial or the spectral domain.

5.2 Test results

As could be expected from the validation accuracies, the classification accuracies on the test set follow the same pattern: Spectral versions consistently have approximately 10% lower accuracy scores and the augmentation doesn't yield as big improvement as for the spatial version.

Table 5.1. *Accuracy scores on the training sets. The differences between the spatial and spectral network scores are bolded. The best scores are marked with green and the worst with red.*

		Database					
		CIFAR-10			MNIST		
Network		Spatial	Spectral	**Diff**	Spatial	Spectral	**Diff**
Simple conv	no augment	74.02 %	67.32 %	**-9.05 %**	99.00 %	98.49 %	**-0.52 %**
	augment	80.78 %	72.33 %	**-10.46 %**	98.85 %	98.24 %	**-0.62 %**
Xception	no augment	81.39 %	73.87 %	**-9.24 %**	99.50 %	98.94 %	**-0.57 %**
	augment	87.65 %	78.25 %	**-10.72 %**	99.35 %	99.02 %	**-0.57 %**

The test results analysis proceeds with examining the confusion matrices. The presented confusion matrices (in the appendix A) have been normalized over the true conditions (rows), meaning that the sum of each row equals one and the range of possible values in each element is [0, 1]. This makes it easier to see the portions of the predicted labels for each class. The differences in the class-wise accuracies between the two networks can

be distinguished by subtracting the two matrices from each other. The following difference tells how many percentage units higher or lower the spatial network's accuracies are.

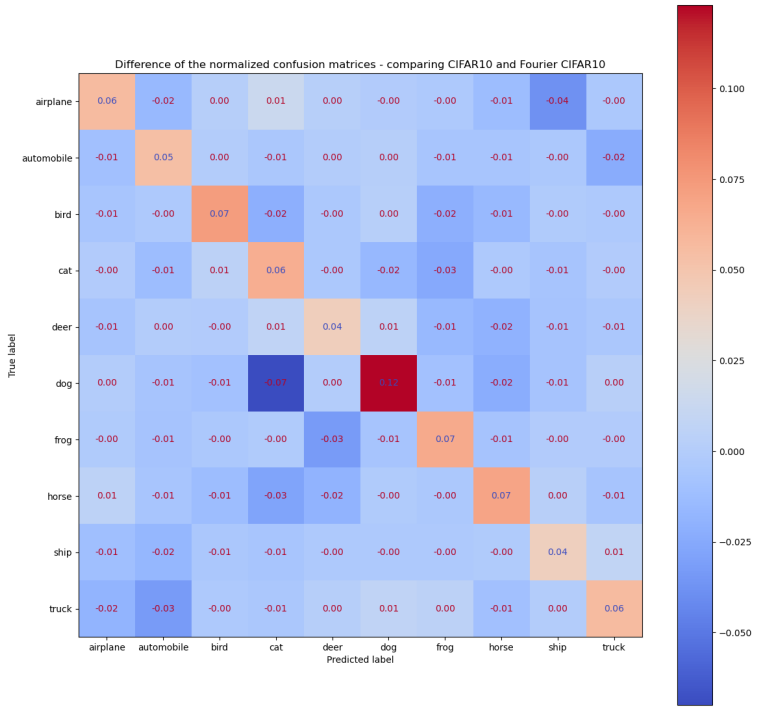


Figure 5.7. *Difference of the CIFAR-10's and Fourier CIFAR-10's confusion matrices. The number represents the difference in the accuracy score's percentage units.*

The spatial network was consistently better at classifying in all classes. The biggest difference is in the cat and dog classes. Both networks struggled with these classes and the most common misclassifications were to classify a dog as a cat, or a cat as a dog. The spectral network struggled significantly more with this. The augmentation layer didn't affect the arrangement, however, it improved the spatial network more.

All of the confusion matrices can be found in appendix A and the more difference matrices can be found in appendix B. The class-wise performance is similar in all cases; cats and dogs get confused with each other, and they are harder classes, and augmentation improves the performance of all networks.

5.3 Discussion

The experimental results indicated, that convolutional neural networks can extract distinctive features from Fourier-transformed images on multi-class classification tasks. This

result is coherent with the conclusion of the previous bachelor thesis of the same topic [8] [9].

However, the validation and test accuracies on all architectures were consistently lower with the Fourier images. This indicates that the features extracted from the Fourier inputs were less distinctive than the features extracted from the spatial inputs.

A second finding against the spectral representation usefulness in image classification is that despite a few releases attempting to solve the problem of classifying spectral images [16] [17] [18], none of them showed sufficient evidence that Fourier transform aids the feature extraction. Quite the opposite, since the accuracy scores on the Fourier-transformed MNIST dataset, were worse than the spatial reference in all papers.

In addition, MNIST is a poor choice for benchmarking in this case since the complexity of this particular dataset is significantly lower than the other common datasets. The MNIST images are in black-and-white format, and there isn't much in-class variation. Figure 5.6 shows that even a novel CNN can learn to classify both spatial and spectral MNIST datasets with differences in accuracy being less than 0.5%.

These points raise the question of why it is so hard to find usage for the Fourier transform in deep learning based image classification task, meanwhile, the Fourier transform is crucial in other fields of signal processing such as audio processing or telecommunications.

One reason for these unpromising results might be that CNNs by design learn to transform the input and eventually extract the distinctive features. Fourier transform is also a process of extracting features. If the Fourier transform was useful, the CNN should in the theory learn to process the inputs in the Fourier-like fashion without needing this transform to be applied as a pre-process. There is no intuition on applying a feature reduction process before CNN.

Another way to gain insight into this question is to examine how visual and auditory signals are interpreted differently from a biological standpoint. Let's analyze the last processing layer of these signals before they are converted to nerve signals in the brain. In the visual system, the nerve impulses are generated by photoreceptor cells, which are positioned in a matrix-like structure inside the retina. Photoreceptor cells are tuned to a certain direction of the photon by way of their location in the retina.

In the auditory system, the nerve impulses are generated by the hair cells which are located inside a cochlea. The hair cells are tuned to a certain frequency based on their location in the cochlea. The cochlea is a tube-like structure where the length of propagation of the sound wave increases as the frequency of the sound wave decreases.

Now, when comparing nerve signals, the one produced by the visual system carries information, on what intensities were detected in which parts of the field of view. The one

produced by the auditory system carries information, on what frequencies were detected in which time window of the auditory input stream. The visual system nerve signal is a spatial representation of the original signal whereas the auditory system's is in a spectral representation.

The nature of the auditory signal requires it to be somehow transformed into a spectral domain before it can be interpreted inside the human brain, whereas, the visual signal has to remain in the spatial domain. With this in mind, it feels unintuitive to apply Fourier transform to images before classifying them.

6. CONCLUSION

In conclusion, the findings indicate that transforming images into the Fourier domain does not yield a performance boost on neural networks on multiclass classification tasks, if the performance metric is classification accuracy. This was confirmed with four different architectures; in all the cases, the spectral version was behind by the same degree. There were no significant differences in the classwise accuracies either, spectral version was always slightly behind. The harder classes in the spatial domain were also harder in spectral. There were no cases or classes where the spectral version was better than the spatial.

The difference between the accuracies on the spatial networks compared to the spectrals' was found to be dependent on the complexity of the dataset. On the MNIST dataset, the difference was merely 0.5% whereas with the CIFAR-10 the difference was around 10%. This aligns with the intuition that Fourier transformation makes the features slightly harder to extract but does not completely lose them for the convolutional network. On the MNIST dataset, the distinctiveness of the features is so high that even after making them harder to extract even the simplest network can achieve an almost perfect score.

The hardest classes to classify for all networks were cats and dogs. The main reason for this was that these classes got easily confused with each other. The easiest class was the automobile.

REFERENCES

[1] Russell, S. and Norvig, P. Artificial Intelligence: A Modern Approach. Global edition. Harlow: Pearson Education, Limited, 2016, pp. 1–2.

[2] Szeliski, R. *Computer Vision: Algorithms and Applications, 2nd ed.* Springer, 2022.

[3] *An On-device Deep Neural Network for Face Detection*. Accessed: 10.11.2023. URL: https://machinelearning.apple.com/research/face-detection.

[4] Bari, M., Ahmed, A., Sabir, M. and Naveed, S. *Mehran University Research Journal Of Engineering & Technology* 38.2 (2019), pp. 351–360. URL: https://search.informit.org/doi/10.3316/informit.291574439064900.

[5] *Valmet toimittaa konenäköjärjestelmän ITC:lle Intiaan*. Accessed: 10.11.2023. URL: https://www.valmet.com/fi/media/uutiset/lehdistotiedotteet/2020/valmet-toimittaa-konenakojarjestelman-itclle-intiaan/.

[6] *Sony animal eye-autofocus*. Accessed: 10.11.2023. URL: https://www.sony.com/et/electronics/animal-eye-af.

[7] *AI Gun Detection Software*. Accessed: 10.11.2023. URL: https://actuate.ai/ai-security/gun-detection.

[8] Sassali, N. Deep Learning with Fourier Transformed Images. (2023). URL: https://trepo.tuni.fi/handle/10024/149045.

[9] Tötterström, S. Frequency Domain Image Classification with Convolutional Neural Networks. (2023). URL: https://trepo.tuni.fi/handle/10024/148933.

[10] Krizhevsky, A., Sutskever, I. and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. Burges, L. Bottou and K. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[11] Russakovsky, O., Deng, J., Huang, Z., Berg, A. C. and Fei-Fei, L. Detecting avocados to zucchinis: what have we done, and where are we going?: *International Conference on Computer Vision (ICCV)*. 2013.

[12] Sharma, P., Basavaraju, S. and Sur, A. Deep learning-based image de-raining using discrete Fourier transformation. *The Visual Computer* 37 (Aug. 2021), pp. 1–14. DOI: 10.1007/s00371-020-01971-w.

[13] Duan, H., Liu, Y., Yan, H., He, Q., He, Y. and Guan, T. Fourier ViT: A Multi-scale Vision Transformer with Fourier Transform for Histopathological Image Classification. *2022 7th International Conference on Automation, Control and Robotics Engineering (CACRE)*. 2022, pp. 189–193. DOI: 10.1109/CACRE54574.2022.9834158.

[14] Yang, T. A Lightweight Method for Fast Deep Learning Inference in Medical Image Classification Based on Fourier Transform and Signal Processing. *2022 16th ICME International Conference on Complex Medical Engineering (CME)*. 2022, pp. 278–281. DOI: 10.1109/CME55444.2022.10063278.

[15] Lappas, D., Argyriou, V. and Makris, D. Fourier Transformation Autoencoders for Anomaly Detection. *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2021, pp. 1475–1479. DOI: 10.1109/ICASSP39728.2021.9415010.

[16] Pratt, H., Williams, B., Coenen, F. and Zheng, Y. FCNN: Fourier Convolutional Neural Networks. *Machine Learning and Knowledge Discovery in Databases*. Ed. by M. Ceci, J. Hollmén, L. Todorovski, C. Vens and S. Džeroski. Cham: Springer International Publishing, 2017, pp. 786–798. ISBN: 978-3-319-71249-9.

[17] Ayat, S. O., Khalil-Hani, M., Ab Rahman, A. A.-H. and Abdellatef, H. Spectral-based convolutional neural network without multiple spatial-frequency domain switchings. *Neurocomputing* 364 (2019), pp. 152–167. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2019.06.094. URL: https://www.sciencedirect.com/science/article/pii/S0925231219310148.

[18] Fülöp, A. and Horváth, A. End-to-End Training of Deep Neural Networks in the Fourier Domain. *Mathematics* 10.12 (2022). ISSN: 2227-7390. DOI: 10.3390/math10122132. URL: https://www.mdpi.com/2227-7390/10/12/2132.

[19] Mathieu, M., Henaff, M. and LeCun, Y. *Fast Training of Convolutional Networks through FFTs*. 2014. arXiv: 1312.5851 [cs.CV].

[20] Rippel, O., Snoek, J. and Adams, R. P. *Spectral Representations for Convolutional Neural Networks*. 2015. arXiv: 1506.03767 [stat.ML].

[21] Hadji, I. and Wildes, R. P. What Do We Understand About Convolutional Networks?: *CoRR* abs/1803.08834 (2018). arXiv: 1803.08834. URL: http://arxiv.org/abs/1803.08834.

[22] Hornik, K., Stinchcombe, M. and White, H. Multilayer feedforward networks are universal approximators. *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[23] Mcculloch, W. and Pitts, W. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics* 5 (1943), pp. 127–147.

[24] Riesenhuber, M. and Poggio, T. Hierarchical models of object recognition in cortex. *Nature Neuroscience* 2.11 (Nov. 1999), pp. 1019–1025. ISSN: 1546-1726. DOI: 10.1038/14819. URL: https://doi.org/10.1038/14819.

[25] URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[26] *MNIST-dataset*. URL: https://www.kaggle.com/datasets/hojjatk/mnist-dataset.

[27] Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR* abs/1610.02357 (2016). arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357.

[28] *Smaller version of the Xception architecture*. URL: https://keras.io/examples/vision/image_classification_from_scratch.

[29] He, K., Zhang, X., Ren, S. and Sun, J. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[30] Shorten, C. and Khoshgoftaar, T. M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* 6.1 (July 2019), p. 60. ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

APPENDIX A: CONFUSION MATRICES

A.1 CIFAR-10 (simple conv. without aug.)

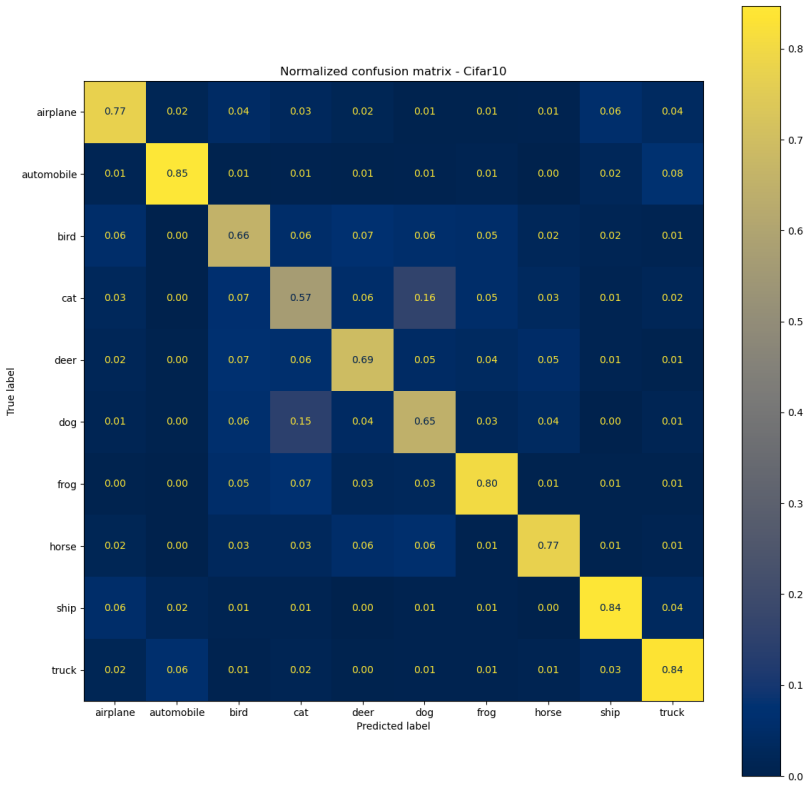


Figure A.1. *Normalized confusion matrix of CIFAR-10 results with a simple CNN without augmentation layer*

A.2 Spectral CIFAR-10 (simple conv. without aug.)

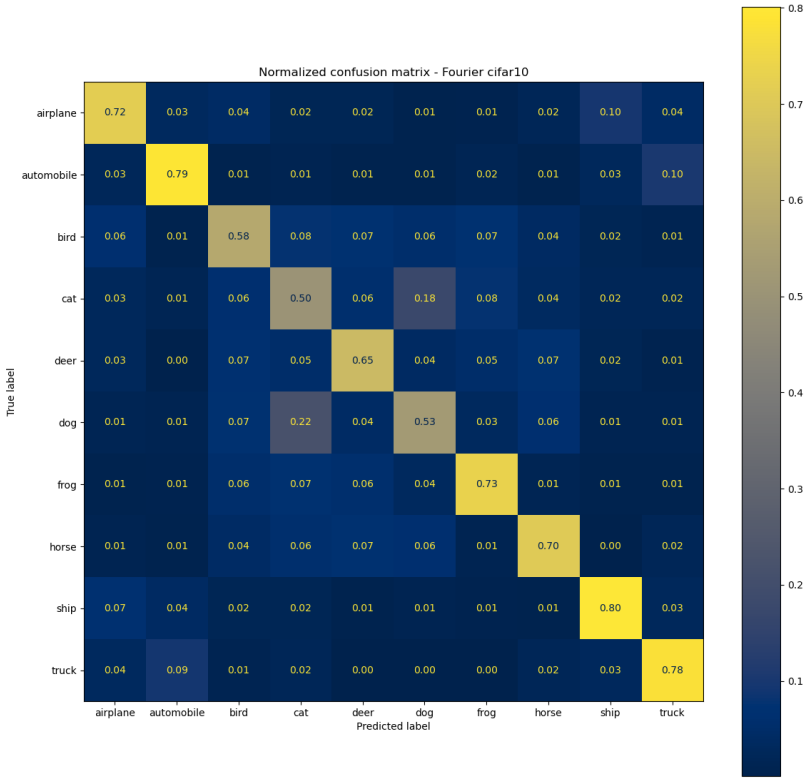


Figure A.2. *Normalized confusion matrix of spectral CIFAR-10 results with a simple CNN without augmentation layer.*

A.3 CIFAR-10 (simple conv. aug.)

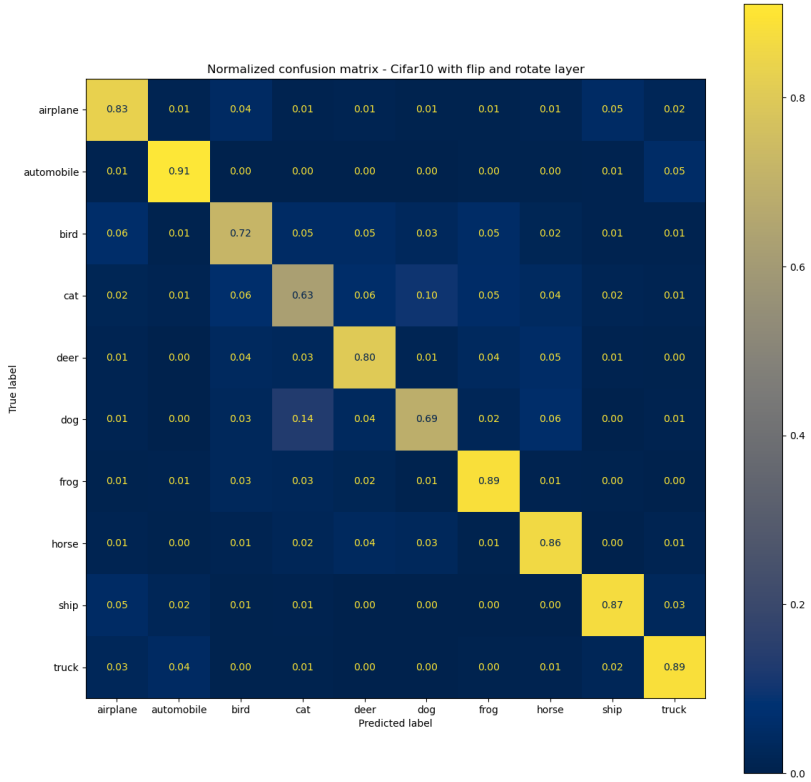


Figure A.3. *Normalized confusion matrix of CIFAR-10 results with a simple CNN with augmentation layer.*

A.4 Spectral CIFAR-10 (simple conv. aug.)

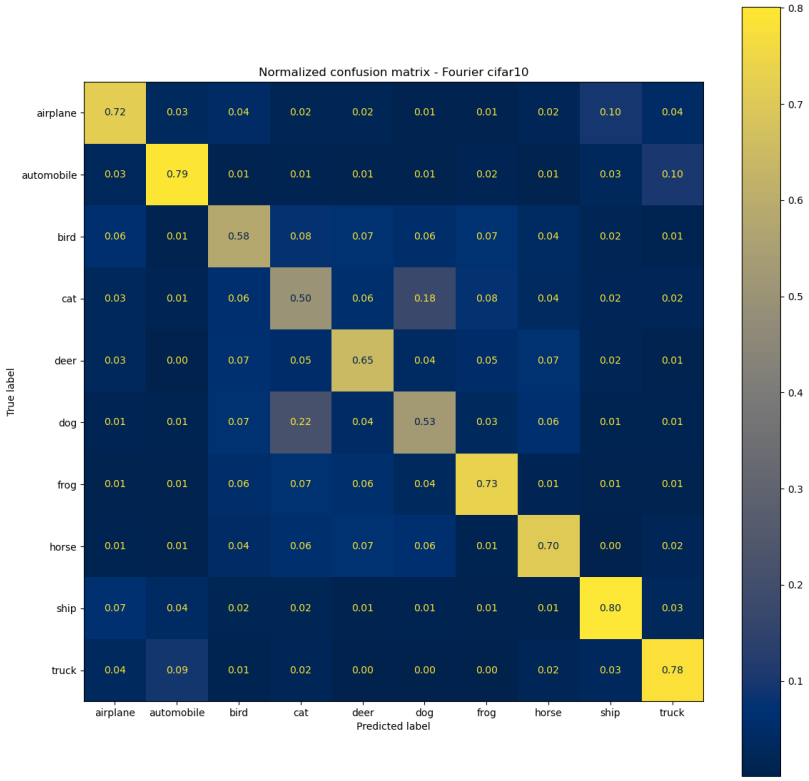


Figure A.4. *Normalized confusion matrix of spectral CIFAR-10 results with a simple CNN with augmentation layer.*

A.5 CIFAR-10 (Xception without aug.)

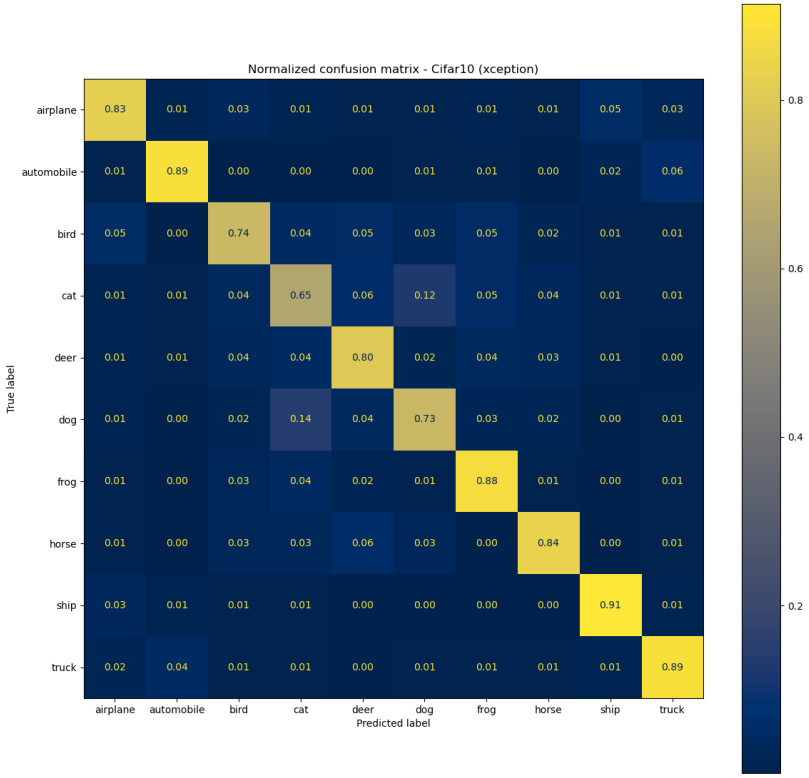


Figure A.5. *Normalized confusion matrix of CIFAR-10 results with Xception network without the augmentation layer.*

A.6 Spectral CIFAR-10 (Xception without aug.)

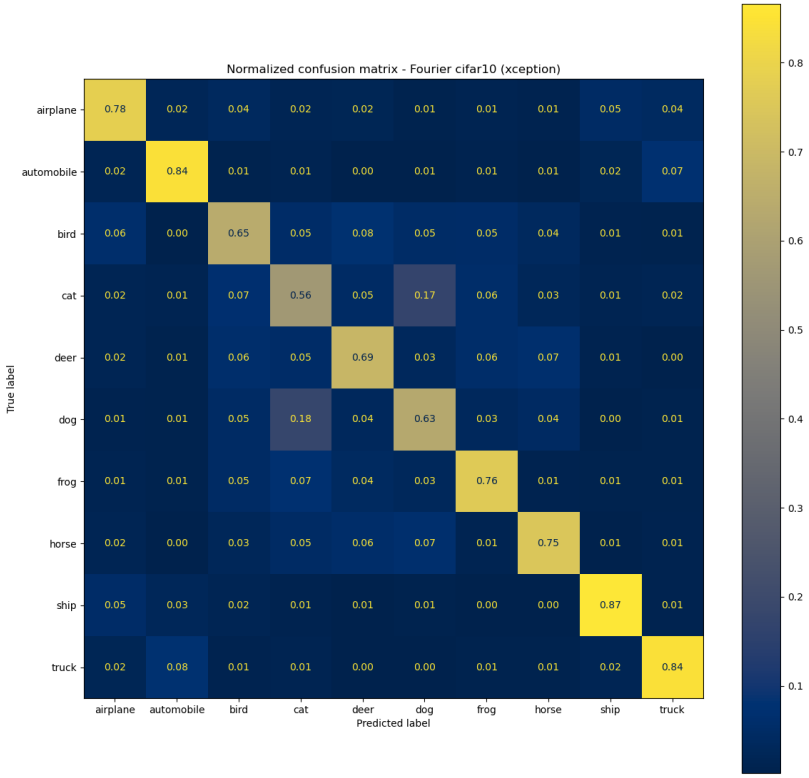


Figure A.6. *Normalized confusion matrix of spectral CIFAR-10 results with Xception network without the augmentation layer.*

A.7 CIFAR-10 (Xception aug.)

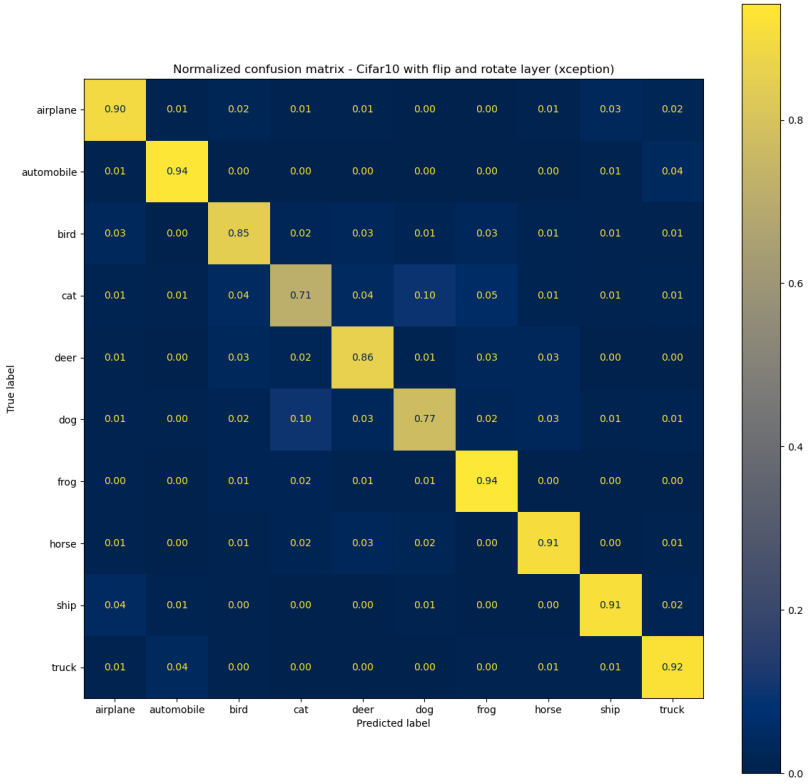


Figure A.7. *Normalized confusion matrix of CIFAR-10 results with Xception network with the augmentation layer.*

A.8 Spectral CIFAR-10 (Xception aug.)

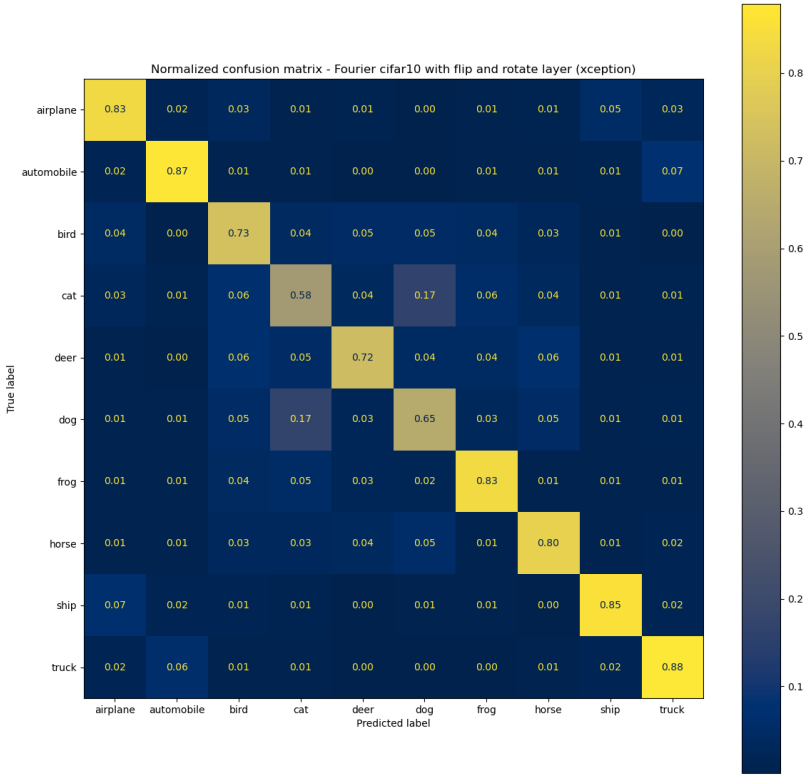


Figure A.8. *Normalized confusion matrix of spectral CIFAR-10 results with Xception network with the augmentation layer.*

APPENDIX B: DIFFERENCE MATRICES

B.1 Comparing CIFAR-10 and Fourier CIFAR-10 (Simple conv.)

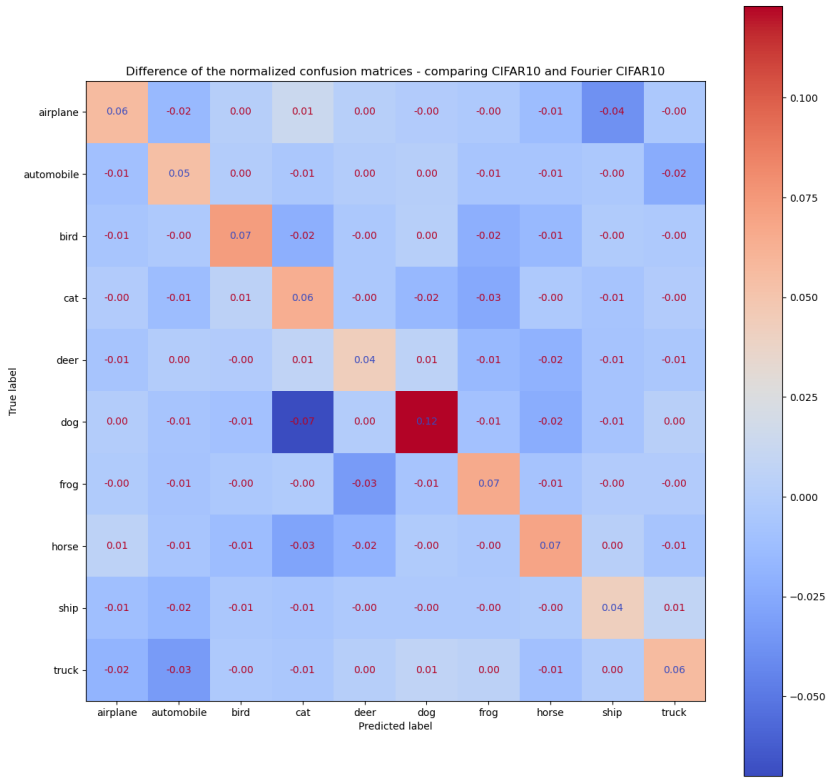


Figure B.1. *Result matrix by subtractingA.2 from A.1*

B.2 Comparing CIFAR-10 and Fourier CIFAR-10 (Simple conv. aug.)

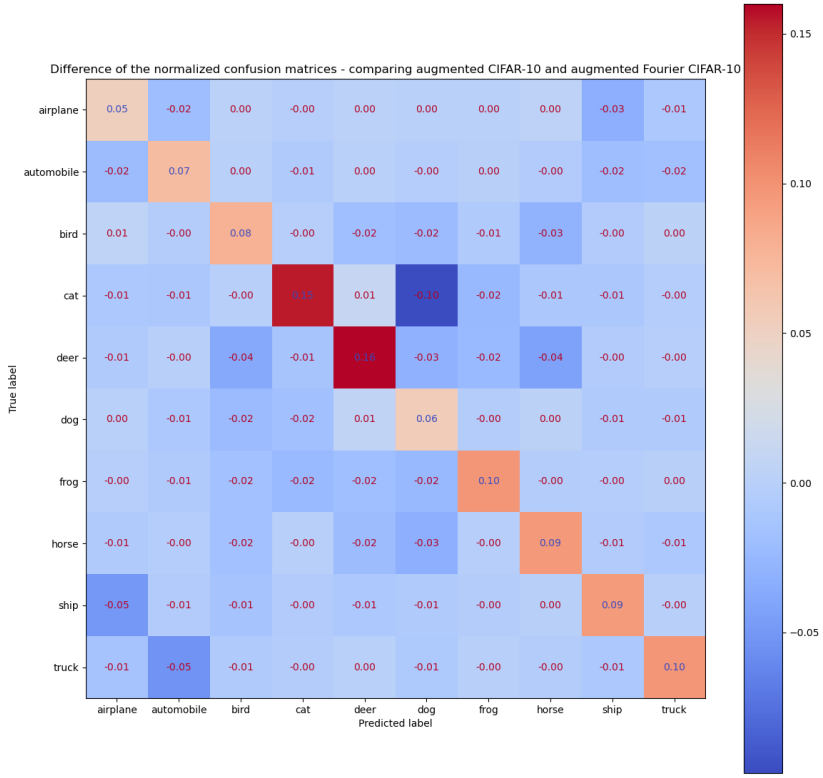


Figure B.2. *Result matrix by subtractingA.4 from A.3*

B.3 Comparing CIFAR-10 and Fourier CIFAR-10 (Xception)

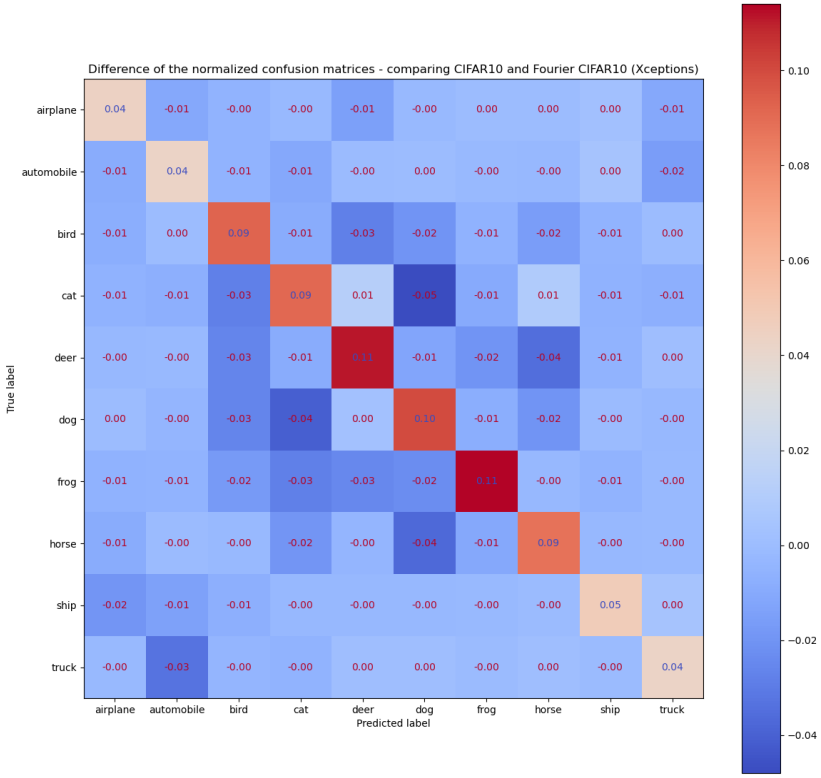


Figure B.3. *Result matrix by subtractingA.6 from A.5*

B.4 Comparing CIFAR-10 and Fourier CIFAR-10 (Xception aug.)

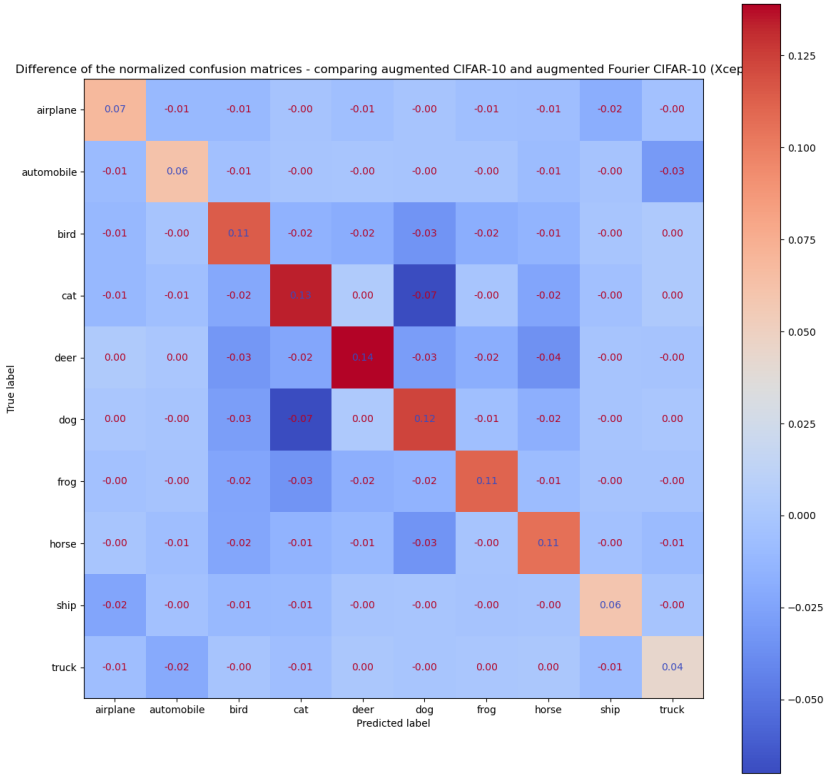


Figure B.4. *Result matrix by subtractingA.8 from A.7*