Tampere University

Sophie Tötterström

FREQUENCY DOMAIN IMAGE CLASSIFICATION WITH CONVOLUTIONAL NEURAL NETWORKS

Bachelor's Thesis
Faculty of Information Technology and Communication Sciences
Examiner: Prof. Joni Kämäräinen
May 2023

ABSTRACT

Sophie Tötterström: Frequency Domain Image Classification with Convolutional Neural Networks
Bachelor's Thesis
Tampere University
Bachelor's Programme in Computing and Electrical Engineering
May 2023

The purpose of this thesis was to explore image classification in the frequency domain using convolutional neural networks. Image classification is a common application of machine learning where image samples are categorized into classes. This application is valuable in many fields, and can be performed using various machine learning methods.

Neural networks are layered structures of computational units. They can be trained to perform classification. Convolutional neural networks add convolution operations to this layered structure.

Traditionally, samples used for image classification have red, green, and blue color channels which display information in the spatial domain. The frequency domain representation of these images captures crucial information about geometric features, which may improve the performance of neural networks.

To conduct this experiment, convolutional neural network models were built following a selected baseline architecture. The models were built for both color images and images transformed into the frequency domain using a Fourier transform operation. These models were then trained and evaluated to compare their performance in relation to accuracy.

While the models required similar amounts of training, the ones using frequency domain images were only able to achieve half the prediction accuracy of their color image counterparts. The results demonstrate that the benefits of convolutional neural networks are lost when classifying images that are transformed into the frequency domain.

Keywords: machine learning, convolutional neural networks, Fourier transform, image classification

TIIVISTELMÄ

Sophie Tötterström: Taajuustason kuvien luokittelu konvoluutioneuroverkkojen avulla
Kandidaatintyö
Tampereen yliopisto
Tieto- ja sähkötekniikan kandidaattiohjelma
Toukokuu 2023

Tämän työn tavoitteena oli tutkia kuvien luokittelua taajuustasossa konvoluutioneuroverkkojen avulla. Kuvien luokittelu on yleinen koneoppimisen sovellus, missä kuvanäytteet kategorisoidaan luokkiin. Kuvien luokittelu voidaan toteuttaa eri menetelmien avulla ja sitä voidaan soveltaa monilla eri aloilla.

Neuroverkot ovat kerroksittaisia rakenteita, jotka koostuvat useista laskennallisista yksiköistä. Ne voidaan opettaa suorittamaan luokittelua. Konvoluutioneuroverkkojen rakenteeseen on lisätty konvoluutiolaskutoimituksen suorittavia laskennallisia yksiköitä.

Perinteisesti kuvanluokittelussa käytetään kuvanäytteitä, jotka koostuvat punaisesta, sinisestä ja vihreästä värikanavasta. Kuvien taajuustasollinen esitys tallentaa keskeistä tietoa geometrisistä ominaisuuksista, mikä saattaa edesauttaa neuroverkkojen suorituskykyä.

Tämän työn toteutus alkoi rakentamalla konvoluutioneuroverkkomalleja valitun lähtömallin arkkitehtuurin pohjalta. Mallit rakennettiin värikuville sekä taajuustasoon Fourier-muunnoksen avulla muunnetuille kuville. Tämän jälkeen mallit opetettiin luokittelemaan ja niiden suorituskykyä vertailtiin.

Spatiaali- ja taajuustason kuville rakennettujen mallien oppiminen kesti yhtä pitkään, mutta taajuustason luokittelijat saavuttivat vain puolet niiden verrokkien kuvanluokittelutarkkuuksista. Tulokset osoittavat, että konvoluutioneuroverkkojen hyödyt menetetään, kun luokittelu suoritetaan taajuustasoon muunnetuilla kuvilla.

Avainsanat: koneoppiminen, konvoluutioneuroverkot, Fourier-muunnos, kuvien luokittelu

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

CNN	Convolutional Neural Network
DFT	Discrete Fourier Transform
FFT	Fast Fourier Transform
IFT	Inverse Fourier Transform
NN	Neural Network
ReLU	Rectified Linear Unit
RGB	red-green-blue

1. INTRODUCTION

Image classification is a machine learning application in which an image sample is assigned a class label based on features in the image. It is a common and well-understood task with a myriad of valuable uses. Neural networks are a combination of layers of computations that are carefully weighted and biased to be especially adept in performing classification [1]. Traditionally, image classification is performed on grayscale or color images.

Fourier transform transforms an input signal into a representation of the signal's frequency content. When performed on two-dimensional images, the transform's output is the frequency content of the intensity function of the input image [2]. The resulting frequency-domain representation contains important information about the original image's spatial features and is useful in many image processing tasks [3].

The purpose of this thesis is to study if convolutional neural networks can be applied to frequency domain image classification, and how accurate they may be in this case. As the Fourier transform reveals information that is not directly available in the original images, perhaps the frequency domain could improve the classification accuracy of a properly trained neural network.

First, related works are explored and background is explained to offer the required information that this research builds on. This includes the fundamentals of image classification using neural networks, and key concepts around Fourier transforms applied to images. With this foundation, this research describes the process of implementing and training the neural network models that will be evaluated. The evaluation of these models includes comparing the accuracy of the experimental model, trained on frequency domain images, to a control model that was trained with typical color images.

Background related to image classification and the Fourier transform, as well as previous works, are discussed in Chapter 2. The theory related to the machine learning methods, such as neural networks, and other necessary methodology for the experiment is described in Chapter 3. The experimentation process and the associated results are outlined in Chapter 4. Finally, the research is summarized in the Chapter 5, including considerations and improvement suggestions for future research.

2. BACKGROUND

Classification is a task where a computer is asked to categorize given inputs into classes [1]. In image classification, a class label is assigned to an input image. This classification can be performed using a variety of different methods, with the most common methods employing neural networks (NN). Typically, neural networks use images with red, green, and blue color channels, or RGB images, as their input. When an RGB image has a Fourier transform applied to it, it produces the frequency and phase content of the image. The image is transformed from the spatial into the frequency domain.

Chapter 2.1 discusses the Fourier transform further. This transform method has been applied to a multitude of problems. The frequency content of an image is useful in many image classification methods. Chapter 2.2 compiles results from previous works regarding classification in both the spatial and frequency domains.

2.1 Fourier Transform

The basic premise of the Fourier transform is that any function can be represented as a sum of sine and cosine components. When applied to a given signal, it provides a way to describe the frequency composition of the signal. As an example, transmission signals like radio or audio signals are functions over time, placing them in the time domain. Applying a Fourier transform decomposes these time-domain signals into the frequency domain. Despite being most commonly used, the Fourier transform is not the only way to convert signals into the frequency domain: notably the Laplace and Z-transforms may be used [4].

While less intuitive, images can also be considered signals. Rather than the time domain like transmission signals, images exist in the spatial domain. Since each pixel of an image has an intensity value, the image can be represented as a function and used as input to a Fourier transform operation. Similarly to the Fourier transform applied to inputs in the time domain, the transform on spatial domain input will also produce the frequency composition as output. [5]

The frequency domain representation of an image will appear different from the spatial domain representation. Each pixel in the frequency domain image represents a specific frequency present in the spatial domain [5]. Despite this different appearance, the geometric characteristics of the spatial domain image are accessed in the frequency domain image. Since the frequency composition of an image represents the change of the pixel values in the spatial domain image, geometric features will be directly mapped to a specific frequency domain representation. For example, an edge in the spatial domain, defined by a high contrast between 2 pixels, will be represented in the frequency domain as a single peak. In this way, the geometric features of the spatial domain are maintained in the frequency domain.

The continuous Fourier transform is generalized from the complex Fourier series [4] and is represented by the following equation

$$F(k) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ikx}\,dx = \mathcal{F}_x[f(x)](k), \tag{2.1}$$

where the transform of the function $f(x)$ at frequency k is its Fourier transform $F(k)$, and $\mathcal{F}_x[f(x)](k)$ is the Fourier transform operation [4, 6].

Many mathematical operations, such as image filtering [3], are computationally and conceptually more simple in the frequency domain [2]. Additionally, frequency domain signals can be transformed back to the original domain. This is done with an Inverse Fourier transform (IFT). The IFT defined using the notation from equation (2.1) is

$$f(x) = \mathcal{F}_k^{-1}[F(k)](x), \tag{2.2}$$

where $f(x)$ is the original continuous function, and $F(k)$ its Fourier transform [6].

2.1.1 Discrete Fourier Transform

While signals can be represented mathematically as continuous functions, most real-world applications use sampled signals with functions that are discrete. Similarly, pixels in an image have discrete values. For these situations, the discrete version of the Fourier transform is used instead of the equation (2.1).

The Discrete Fourier Transform (DFT) transforms a signal into the frequency domain. DFT is defined by the following equation

$$F(k) = \sum_{n=0}^{N-1} f(n)e^{-2\pi ink/N}, \tag{2.3}$$

where $k \in \mathbb{Z}$ is the wave number, $f(n)$ is a discrete-time signal with signal length N, and $F(k)$ is its N-point discrete Fourier transform [4].

The 2-dimensional DFT of a square image with the size $N \times N$ is given by

$$F(k,l) = \frac{1}{N^2} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} f(n,m) e^{-2\pi i (kn/N + lm/N)}, \tag{2.4}$$

where $f(n,m)$ is the spatial domain image, $N \times N$ the size of the image matrix thus N is the length or width of the image, and $F(k,l)$ the DFT for frequency (k,l) [5].

2.1.2 Fast Fourier Transform

Several methods for calculating the DFT are described in section 2.1.1. Calculating the DFT of a $N \times N$-matrix described with equation (2.4) requires $O(N^2)$ computations. However, such a matrix only has N distinct values. Algorithms that take advantage of this to reduce the computational complexity of the DFT computation are known as Fast Fourier Transform (FFT) algorithms. The computational complexity is generally reduced to $O(N \log_2(N))$. [5]

Multiple FFT algorithms have been developed, with one of the most common being the Cooley-Tukey algorithm [7]. It effectively divides the DFT into smaller problems which are recursively combined into a solution to increase computational efficiency. The algorithm exploits the symmetry of the matrix by only considering distinct values, and trigonometric constant coefficients described as "twiddle factors" [7]. The FFT implementation used in this thesis is based on the Cooley-Tukey algorithm.

2.1.3 Fourier Spectrum

The frequency information of a signal over a range is its spectrum [2]. The frequency content of an image is retrieved by computing its DFT using equation (2.4). The result of this is complex-valued. One composition of this result is the signal's magnitude and phase spectra.

In order to represent the given spatial domain sample in the frequency domain, important frequency characteristics are defined. These characteristics provide a way to visualize the DFT without accounting for complex numbers. The magnitude of a DFT image point is calculated with the equation

$$|F_(k,l)| = \sqrt{(\mathrm{Re}(F_(k,l))^2 + (\mathrm{Im}(F_(k,l))^2}, \tag{2.5}$$

where Re and Im are the real and imaginary parts of the DFT value $F_(k,l)$ of the specified pixel [8].

The phase of a DFT is defined as

$$\arg(F_(k,l)) = \arctan \frac{\mathrm{Im}(F_(k,l))}{\mathrm{Re}(F_(k,l))}, \tag{2.6}$$

where similarly Re and Im are the real and imaginary parts of the DFT value $F_(k,l)$ of the specified pixel.

The magnitude represents the intensity of each frequency of an image, thus it contains the changes in the spatial domain. The phase represents the locations of these features. [5]

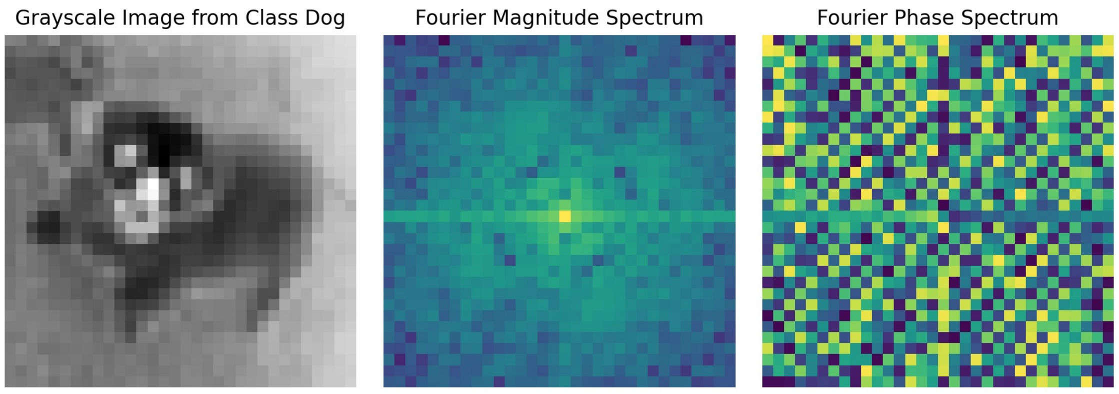


Figure 2.1. *A grayscale image of a dog and the computed magnitude and phase spectra.*

Figure 2.1 exemplifies how the magnitude and phase spectra might look for a given 32×32 grayscale image. Unlike for grayscale images, the magnitude and phase spectra of an RGB image need to be calculated separately for each color channel.

2.2 Previous Works on Image Classification

Image classification involves assigning labels to objects based on previous training samples. More complex classification tasks include classification when there are multiple objects in one image or a situation where the object is partially occluded. There are multiple different ways to create a classifier, with the most common involving supervised learning of neural networks. This will be discussed further in chapter 3. Image classification has applications in multiple different fields, including autonomous vehicles [9], and healthcare [10].

2.2.1 Image Classification in the Spatial Domain

In image classification, the input images are commonly either grayscale or red-green-blue (RGB) images. Grayscale images are effectively a matrix where each value corresponds to the intensity between white and black. Since the grayscale image is directly represented by a single matrix of "pixel" values, it is considered to have one channel. RGB images however consist of three channels; one for the intensity of each of the three colors for any given pixel.

ImageNet [11] is a large-scale hierarchical image database with more than 14 million images and 20 000 classes. When the convolutional neural network AlexNet [12] was published in 2012, it became the state-of-the-art solution for the classification of RGB image samples in ImageNet. Later, more accurate and deeper neural networks [13] were developed for the task. However, these networks introduced a new problem: classification accuracy degrades as more layers are added. To solve this problem, residual neural networks were introduced [14]. The latest ImageNet classification solutions process data in parallel [15], and are transformers [16], which consider the significance of the input data instead of weighing all samples equally.

2.2.2 Image Classification in the Frequency Domain

Current high-performing image classification systems rely on several operations to extract information from spatial domain images making them computationally expensive. Since the frequency domain representation of an image contains information about an image's geometry, as discussed in section 2.1, the idea of using the frequency domain transformed images has been proposed to increase training efficiency.

Convolutional neural networks utilize multiple convolution operations, which are discussed in section 3.2. To reduce the number of convolution operations, previous works suggest that performing image classification in the Fourier-transformed frequency domain may only require one convolutional operation [17]. Other works have introduced the concept of a Fourier convolutional neural networks with a similar structure, which would offer benefits in NN training speeds while achieving high accuracy [18, 19]. Stuchi et al. [8] have also proposed a faster training process that relies on extracting local and global features from images, transforming them to the frequency domain, and then using the magnitude of these blocks to train a NN. The proposed training process was not able to achieve superior results compared to AlexNet but did reduce the number of required parameters, increasing the speed of the training due to the omitted convolutional layers.

3. METHODOLOGY

Artificial neural networks are machine learning techniques that loosely model the structure of the human brain [20]. They are introduced in section 3.1. Neural networks work exceptionally well for many tasks, including image classification [21, 22], which is why they were selected for this thesis. Convolutional neural networks, discussed in section 3.2, were selected specifically based on previous works on the topic described in section 2.2. Finally, the selected baseline model, AlexNet, will be discussed in section 3.3.

3.1 Neural Networks

Artificial neural networks perform many small computations through multiple layers of the network [1]. The structure of neural networks (NNs) is inspired by a biological brain, where information regarding stimuli passes through connected neurons [20]. Each computation in artificial neural networks is performed by an individual artificial neuron, often only referred to as a neuron [1]. These single computational units work in unison to combine simple computations to perform complex tasks.

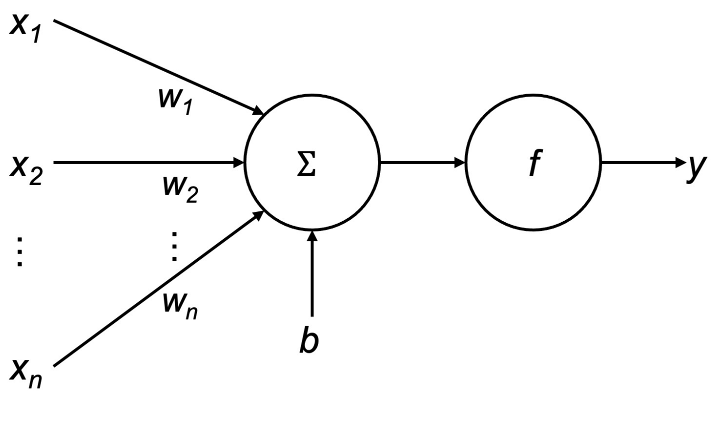


Figure 3.1. *An artificial neuron.*

The structure of an artificial neuron is presented in Figure 3.1 where $x_1, x_2, ..., x_n$ are the input values, $w_1, w_2, ..., w_n$ the weights, b is the bias, a constant added to offset the result [20], f the activation function, and y the output. The weights can be considered to demonstrate how influential the given input is. The neuron calculates the weighted sum of the inputs, adds the bias, and uses the result as an input for the activation function.

The activation function acts as a threshold, defining whether the given inputs meet the threshold requirements to activate the neuron [3].

The purpose of optimizing the weights for each input is to minimize the error in prediction [20]. This will maximize the prediction accuracy. Supervised learning is a NN training strategy where correct predictions are provided, and the weights in each input are optimized to minimize a selected loss function [20, 23].

A NN is formed by combining multiple neurons into a layer, and joining layers together by connecting one layer's neurons to the next layer's neurons. The input and output layers exist on the edges of the NN structure, with one or more layers existing between them. Since a neuron's input maps linearly into its output, it can only solve linear problems. Combining neurons allows a model to solve non-linearly separable problems. [3]

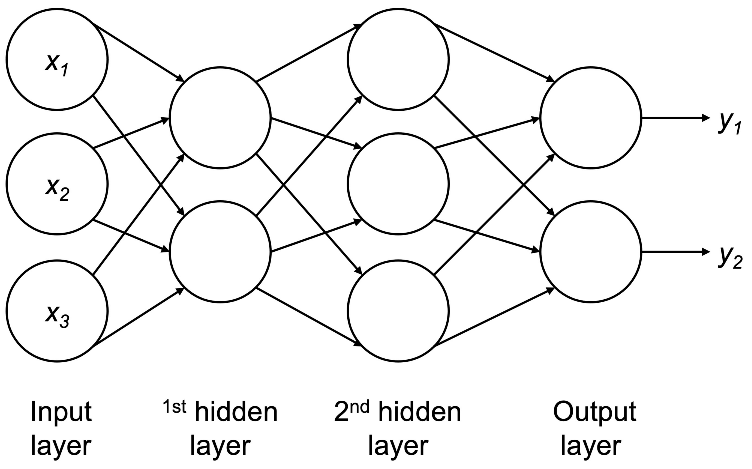


Figure 3.2. *A simple fully-connected multi-layer NN.*

A simple example of a multi-layer NN is portrayed in Figure 3.2. The layers between the input layer and the output layer are known as hidden layers [3]. When all of the neurons of a layer are connected to all of the neurons of the next layer, the NN is described as fully-connected. The NN in Figure 3.2 is an example of a fully-connected neural network.

3.1.1 Back-propagation

Figure 3.2 illustrates a feed-forward NN where the input passes through the system and gives an output. None of the neurons' outputs are given as inputs to neurons in previous layers. This is forward propagation [1]. Back-propagation is the opposite, where the outputs of neurons are given as inputs to previous layers.

An ideal NN has optimal values set for the weights in the network to perform a given task accurately. The basic principle of the back-propagation algorithm is to determine the relationship between each weight and the total loss function, and tune the weights to minimize the loss computed at each output layer [3]. Back-propagation offers the ability to optimize the weights further as the loss is minimized.

Training a NN relies heavily on back-propagation and having a sufficient amount of diverse training data. All samples to be classified need to be divided into a training, validation, and test data set. The training data is used to optimize the weights of the NN, using a back-propagation algorithm. After the NN model has been trained with training data, the parameters are further tuned using separate validation data and back-propagation. Finally, test data is used to evaluate the model. When classifying the samples from the test data, the weights in the NN are no longer modified, allowing the NN performance to be evaluated.

3.1.2 Activation functions

A crucial element of designing a NN for a given task is the selection of the activation functions [20]. For example, AlexNet employs specific activation functions in each layer of the NN to yield the best results when performing image classification for multiple classes [12].

A crucial activation function used in AlexNet is the rectified linear unit (ReLu) [24], which produces linear output for positive input values, but zero when the input is negative. This improves the model's ability to generalize with a large variety of data by ensuring linear separability [20]. ReLu also speeds up the training process, as the output of the activation function simply depends on the sign of the input [12]. ReLu follows the equation

$$\mathrm{ReLu}(x) = \max(0, x) = \begin{cases} x & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases}, \tag{3.1}$$

where x is the weighted sum of the inputs.

For k-class classification, AlexNet uses softmax on a later layer of the network, where each unit outputs the probability of each class [20]. Softmax calculates the probability distribution of the input vector with respect to the outputs $\vec{x}$, and it is defined to be

$$\mathrm{softmax}(\vec{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \forall i \in \{1, ..., k\}, \tag{3.2}$$

where k is the number of classes. [1, 23]

3.1.3 Batch Normalization

Normalization is the process of scaling the samples to a specific range, and specifically standardization rescales the data to have a mean of zero and standard deviation of one. As demonstrated in Figure 3.2, NNs perform activation function operations in multiple layers, where the size of the output for each layer can vary [3]. Batch normalization makes the training of the NN faster and improves its performance [25]. If z is the activations of one layer, batch normalization is performed

$$z_{bn} = \frac{(z - \mu_z)\alpha_z}{\sigma_z}, \tag{3.3}$$

where μ_z is the mean, σ_z the variance of the activation batch, and α_z a parameter the network learns during training [3]. Like the name suggests, batch normalization is performed with one section of the data at a time, and the effects are applied back to the weights via back-propagation [25].

3.1.4 Max-pooling

Max-pooling computes the maximum of a submatrix from the original input matrix. In AlexNet this input matrix is the output of a previous layer. The results are pooled versions of the input which equalize the influence of frequent features [26]. Max-pooling is able to reduce the dimensions of the input while producing another layer with the same depth [20]. Max-pooling allows the NN to learn from local features efficiently while ignoring the effect of noise, unlike average pooling. This strategy also allows the NN to consider local features to be more relevant than others [26].

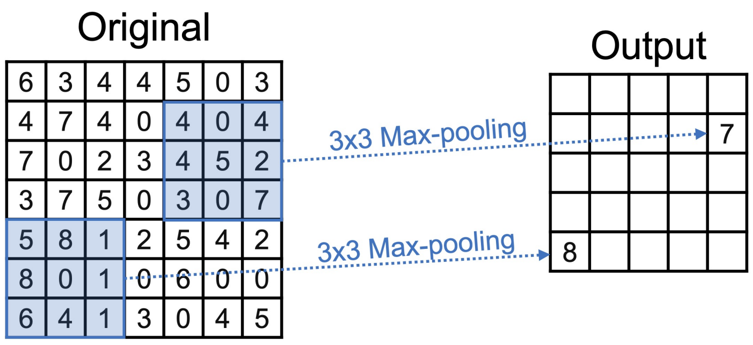


Figure 3.3. *3×3 Max-pooling with stride value 1. Adapted from source [20].*

Figure 3.3 visualizes 2 separate max-pooling operations on the input matrix. The maximum of a 3×3 area is processed and the result is placed in the output matrix. The area is "slid" over the original matrix by moving it one stride at a time. The stride is selected to be one in the example, which means the area moves one step at a time in a given direction. The stride value determines how much the input is down-sampled: the bigger it is, the smaller the output, at the expense of detail.

3.1.5 Dropout

Dropout is a technique where random neurons and their connections are temporarily removed from the NN during training [27]. It prevents over-fitting with randomly imposed sparsity and provides a method for combining many different NN architectures [28]. Over-fitting happens when a model learns the details of the training data too well and is unable to generalize properly.

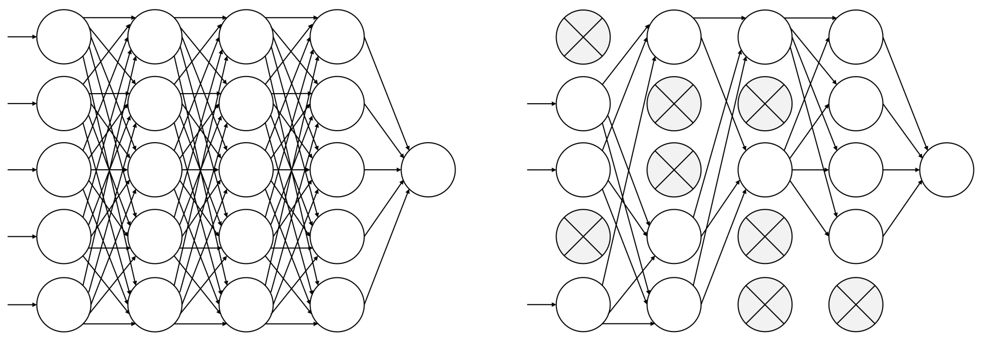

(a) NN before dropout. ***(b)*** NN after dropout.

Figure 3.4. *Dropout performed on a fully-connected neural network.*

Figure 3.4 demonstrates the dropout method on a fully-connected NN. The NN after dropout has randomly selected nodes removed which would not be considered during training. Dropout uses node sampling from input and hidden layers, which makes it impossible to compute the loss or provide a prediction with the information from these units [20]. When training the NN with back-propagation, the sampled nodes are selected in each iteration, which means the same units are not always ignored.

3.2 Convolutional Neural Networks

Convolution is a mathematical operation where a function is processed with another, producing a new result. It is represented with the equation

$$(f * w)(t) = \int f(\tau)w(t-\tau)d\tau, \tag{3.4}$$

where f and w are functions, and $w(n-\tau)$ is a weighing function which shifts the function $f(n)$ in the τ-axis [23]. The discrete convolution derived from equation (3.4) is defined as

$$(f * w)(t) = \sum_{m=0}^{N-1} f(\tau)w(t-\tau), \tag{3.5}$$

where f, w are functions in N-dimensional space [5]. In convolutional neural networks, the first argument f is the input, while the second w is known as the kernel. The output of a discrete convolution operation can be referred to as the feature map. [1]

When classifying objects in images, the spatial and temporal context of features is relevant. When NNs perform classification, they first extract small features from the images and compute an output based on the features. Later layers combine these extracted features into larger concepts, eventually allowing the NN to assign classes to the object.

Convolutional neural networks (CNNs) are neural networks that include convolutional layers. These layers perform convolution operations following equation (2.4) which reduce images while retaining relevant information. For example, one layer might retain edges while other irrelevant data is stripped away. [3] Each convolutional layer activates different features from the image [20]. CNNs are thus able to learn the features of the input image especially well, and when this is utilized in classification, the results are superior to most NNs [21].

In traditional NNs, the features extracted from an image are transformed into an easily processable format, such as a vector. They often use matrix multiplication to connect the layers, and each input maps to an output value, making them fully-connected. They are prone to over-fitting to training data, and generalization is difficult [3, 20]. CNNs replace the matrix multiplication of traditional NNs with a convolution operation in some layers [1]. The neurons performing convolution operations output smaller matrices compared to the inputs. This makes CNNs sparsely connected, which also reduces the number of computations required [1].

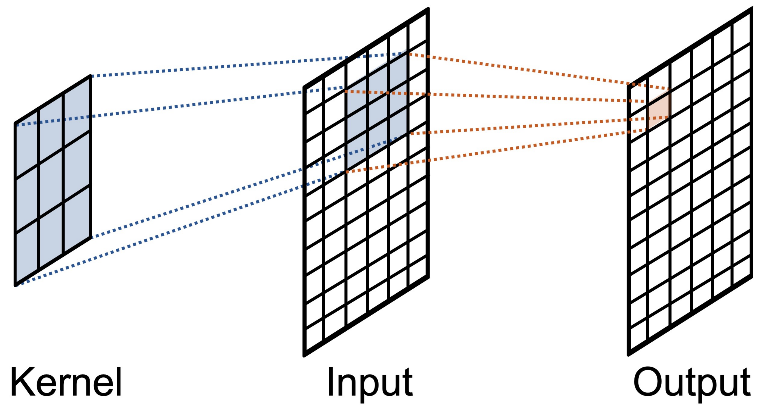


Figure 3.5. *A 1-dimensional convolution operation.*

Figure 3.5 demonstrates the connection between the kernel, input, and output matrices in equation (3.5). The kernel is "slid" across the input matrix, and the result is mapped to the output. The size of the output depends on the selected stride, the amount the kernel is moved between operations. Similarly in CNNs, convolutional layers take an input, for example for the first layer directly a portion of the input image, and perform the convolution operation with the kernel [3]. As the kernel is smaller than the input image, the resulting output reduces the input depending on the stride.

3.3 Baseline Model

AlexNet [12] is a CNN architecture optimized for classifying images from the ImageNet database. Figure 3.6 demonstrates the AlexNet model architecture. The model's input is an RGB image which is resized. The first 5 layers are convolutional layers, each of which has a ReLu activation function, not shown in the figure. These layers are also followed by batch normalization. A portion of the convolutional layers are followed with a max-pooling layer to reduce the kernel size.

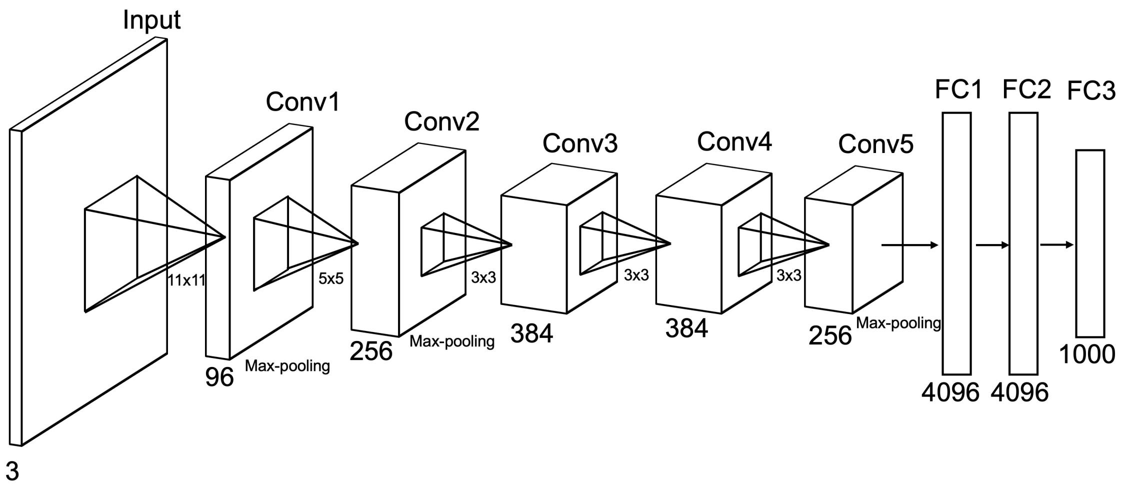


Figure 3.6. *Illustration of the AlexNet [12] architecture. Modified from source [20].*

The last three layers are fully-connected, with the first two being dense layers holding 4096 neurons each. As all the outputs of the first dense layer are given as inputs to the next one, the weights for the neurons in these layers are heavy to compute. The last layer performs the softmax operation in (3.2) to reduce output to the same size as the number of classes in the data set. The output of this layer is the probabilities for the input belonging in each class.

The loss function being minimized in the AlexNet implementation is cross-entropy, or logarithmic, loss. The loss function for a single instance is

$$L = -\log(\vec{y_r}), \tag{3.6}$$

where $\vec{y_1}...\vec{y_k}$ are the probabilities of the k classes from the softmax activation function (3.2), and the rth class is the ground-truth. [20]

AlexNet was selected as the baseline model for this thesis, as it is reasonable to build locally and the architecture is well-optimized for image classification. The implementation and changes to the structure are further discussed in chapter 4.

4. EXPERIMENTS

The purpose of these experiments was to investigate if Fourier magnitude and phase spectra images can be used for image classification with convolutional neural networks. Models were built for both RGB images and Fourier spectra. The models trained and evaluated with RGB images were then compared against the models trained on Fourier spectra.

The experiments were conducted on a MacBook Pro and implemented using the Python programming language. Multiple open-source Python libraries were utilized: one of the most significant being TensorFlow [29]. It is an open-source software library for machine learning tasks developed by the Google Brain Team that offers an API that allows it to be used with a variety of programming languages. Amongst other things, TensorFlow provides methods for building neural networks that follow desired architectures, as well as methods for training and evaluating them. Keras [30] was used as an interface to the TensorFlow library. Finally, NumPy [31] was used to perform mathematical operations on multi-dimensional matrices.

Different models following similar architectures were defined and implemented. The data used for training these models, as well as how it was processed, is described in section 4.1. The training process itself is described in section 4.2. The results are presented and discussed in section 4.3.

4.1 Data Sets and Pre-processing

The CNN used for this thesis was an own implementation of AlexNet [12] described in section 3.3. The training was performed on a CPU of the used device. Due to this, the data sets needed to be chosen carefully to limit the computational complexity of the operations. As a result, the CIFAR-10 and CIFAR-100 data sets [32] were selected. The CIFAR-10 data set contains 60 000 32×32 RGB images belonging to 10 non-overlapping classes. Similarly, the CIFAR-100 data set contains the same number of samples in the same format, but the samples are divided among 100 classes instead. The data sets were loaded from Keras to avoid needing to store data locally. After loading, the data sets were then split into training, validation, and test data.

Figure 4.1. *Example images from the CIFAR-10 dataset [32] classes.*

Figure 4.1 displays example RGB images used for training. Since a CNN model was trained for both RGB images and Fourier magnitude and phase spectra, the data processing steps varied depending on the input. When a model for classifying RGB images was trained, the data was split and pre-processed in accordance with the AlexNet structure described in section 3.3. The pre-processing involved resizing the data, as well as performing batch normalization to have a mean of 0 and a variance of 1.

The model structure for classifying frequency domain images is the same as for RGB images apart from the input size. Instead of a 3 channel image, the model classifying FFT images takes input samples with 6 channels, thus their size is $6 \times 32 \times 32$. As described in section 2.1.3, the DFT of an image is complex-valued, and in order to represent it, the magnitude and phase spectra are calculated. Thus for each channel of an RGB image, the frequency representation will yield 2 channels.

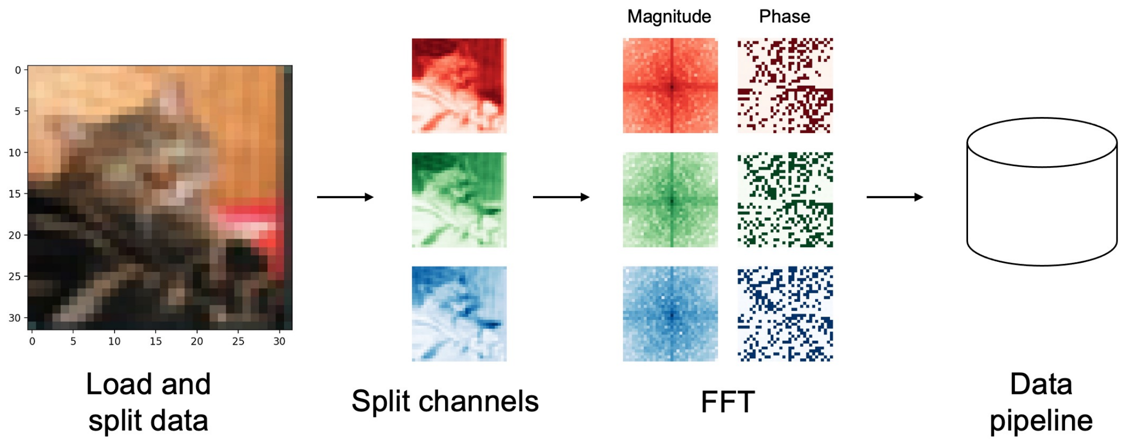


Figure 4.2. *The pre-processing pipeline for the FFT image classifier.*

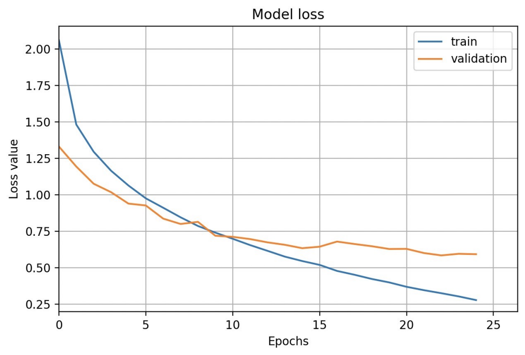
The pre-processing pipeline for the FFT CNN is shown in Figure 4.2. First, data is loaded and split into training, validation, and test data. At this point, the data contains RGB images with dimensions $3 \times 32 \times 32$. After this, the DFT of each sample is calculated, and the data is now represented with dimensions $6 \times 32 \times 32$, where the first 3 channels contain the magnitude spectra of the RGB channels, respectively, and the latter 3 contain the phase spectra. The last step of pre-processing the data involves resizing the image to fit the appropriate model architecture, normalizing it as described above, and distributing it into batches. Dividing the data into batches of 32 images helps the model learn from the data before seeing all of it [12].

4.2 Training the Models

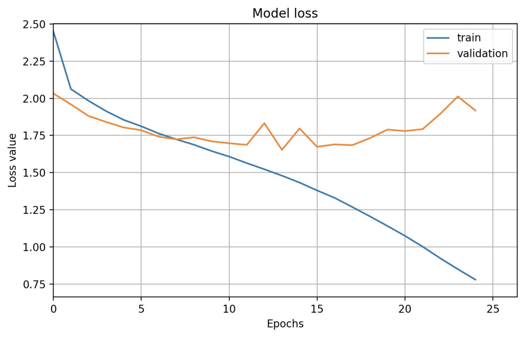
The classification was performed using an own implementation of the AlexNet [12] model. The architecture is described in more detail in section 3.3. First, a baseline model for RGB image classification was implemented and trained on CIFAR-10 data. The model was initially trained for 5 epochs which provided preliminary results for script optimizations. An epoch determines how many times the model is re-trained on the full training data set [3].

Next, a model for Fourier magnitude and phase spectra classification was implemented following the AlexNet model structure in Figure 3.6. However, the input shape was changed to accommodate 6 channels. These were the magnitude and phase spectra for all three color channels demonstrated in Figure 4.2.

Finally, four separate models were trained for 25 epochs. For both CIFAR-10 and CIFAR-100 data sets, two models were trained; one using the RGB images, and one using the magnitude and phase spectra. A high number of epochs increased the time spent on training but did provide more reliable classification results. The models trained on RGB images would provide benchmark results to use for the experiment.



(a) RGB images

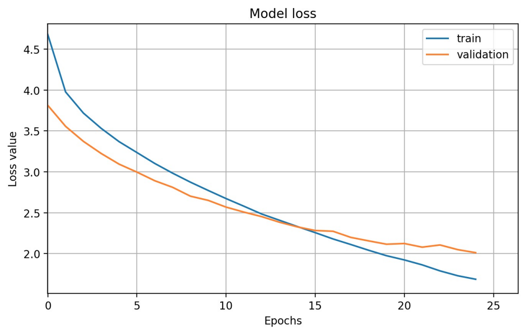


(b) FFT magnitude and phase spectra

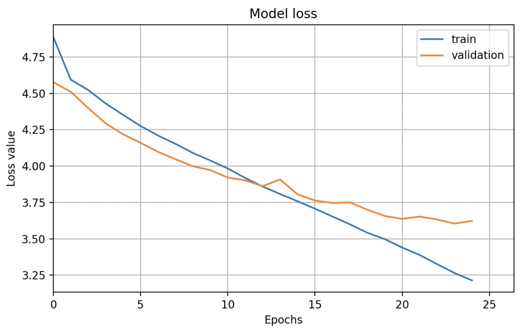
Figure 4.3. *Model loss for CNNs trained using the CIFAR-10 data set.*

Figure 4.3 displays the CNN model loss as the training progressed. These models both used CIFAR-10 data, but the model shown on the right was trained with FFT data. When a model learns and classification improves during training, the loss value decreases as the epochs progress.

Figure 4.3a demonstrates the training of the model trained on CIFAR-10 RGB images. Both the training and validation loss decrease, meaning the model is learning with each epoch. The training on Fourier spectra shown in Figure 4.3b progresses differently. During the first 5 epochs, the model is learning from the training data well, which is demonstrated by the decrease in loss. However, after these initial epochs, the validation data loss becomes stagnant, and even increases as the epochs progress, diverging from the training loss which continues to decrease. After 10 epochs, the model starts over-fitting to the training data. It has learned specific features of the training data too well, resulting in an inability to generalize enough to accurately classify the separate validation and test data.



(a) Using RGB images



(b) Using FFT magnitude and phase spectra

Figure 4.4. *Model loss for CNNs trained using the CIFAR-100 data set.*

The next two models with the same architecture were trained on CIFAR-100 data instead. The training of these two models is depicted in Figure 4.4. The training with RGB images displayed in Figure 4.4a progresses similarly to the CIFAR-10 data training. Both the validation and training loss decrease as the epochs progress, and the model learns well. Figure 4.4b illustrates improvements to the learning process when CIFAR-100 data is used instead. Unlike with CIFAR-10, the model loss continues to decrease with the epochs. This demonstrates learning without perceivable over-fitting to the training data.

4.3 Results and Analysis

Table 4.1 are the test data classification results for the 4 models after training. These results include the accuracy of the classification, defined as the portion of correctly predicted test images, as well as the loss value.

Figures 4.3 and 4.4 displaying loss values during training indicate that the selected model for RGB image classification was able to learn from the data faster and more reliably than its frequency domain counterpart. The accuracies presented in Table 4.1 demonstrate that the worse learning results result in significantly worse test data accuracy, regardless of the data set.

Table 4.1. *Test data classification results of differently trained models*

	Top-1 Accuracy (%)	*Loss*
CIFAR-10		
RGB	79.23	0.65
FFT	38.78	1.94
CIFAR-100		
RGB	47.80	1.98
FFT	15.84	3.61

As described in section 2.2, CNNs are well known to be especially suitable for RGB image classification. The model implementation for RGB image classification was able to perform well. Due to the limitations of the training setup and model implementation, the results were not quite as good as in the original article [12]. This was a combination of the limited size of the data set, the resolution of the samples, as well as the number of epochs. The implemented model does provide a good baseline for its counterpart, as they were both implemented, trained, and tested following the same procedures.

The results from the frequency domain image classification clearly indicate that a CNN model does not yield superior classification results compared to the RGB images. CNNs perform well in image classification as they are able to extract geometric features from the images. However, when classifying magnitude and phase spectra, all such information has already been extracted. The convolutional layers of the model do not offer additional information only increase the number of parameters.

5. CONCLUSION

Image classification is an important application of machine learning. Many methods have been proposed to perform image classification but convolutional neural networks are the state-of-the-art solution. Studying and comparing the performance of these methods is an essential part of research.

The Fourier transform described in section 2.1 has become crucial in many image processing applications. It offers a unique way of accessing the frequency features of images compared to the spatial domain. The goal of this thesis was to study how CNNs would perform when performing classification using frequency domain samples. This was done by implementing and training CNNs with RGB and Fourier spectra images. The implementation source code for conducting this research, as well as documentation and pre-trained models, can be found on GitHub. [1]

CNNs rely on the convolution operation to extract information from images, which is why they are successful in image classification. However, in the frequency domain, this information has already been extracted. As a result, the convolution operation on frequency domain images only increases computation complexity while negatively impacting classification results. This work verified the results of previous works, demonstrating that image classification on Fourier transformed images is not suitable for CNNs.

The benefit of performing image classification in the frequency domain is the ability to design a NN with fewer layers, allowing it to be trained relatively quickly and maintain or even improve classification accuracy. This benefit is neglected when performing classification with convolutional neural networks, as the Fourier transform adds a layer on top of the convolutional layers. A more shallow NN architecture for frequency domain classification should be explored in future research, as previous works have shown promising results for them.

[1] `https://github.com/sophietotterstrom/fourier-alexnet`

REFERENCES

[1] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. [Online]. MIT Press, 2016. URL: http://www.deeplearningbook.org (visited on 03/10/2023).

[2] Broughton, S. A. *Discrete Fourier analysis and wavelets applications to signal and image processing*. 1st edition. Hoboken, N.J: John Wiley & Sons, 2009, pp. 71–104. 356 p. ISBN: 1-282-24264-4.

[3] Venkatesan, R. *Convolutional neural networks in visual computing : a concise guide*. Data-enabled engineering. Boca Raton: CRC Press, 2018, pp. 65–115. ISBN: 978-1-351-65032-8.

[4] Beerends, R. J. *Fourier and Laplace transforms*. Cambridge University Press, 2003, pp. 60–80, 138–161, 340–389. 447 p. ISBN: 1-107-13134-0.

[5] Damelin, S. B. *The mathematics of signal processing*. Cambridge texts in applied mathematics ; 48. Cambridge: Cambridge University Press, 2012, pp. 129–219, 412–419. ISBN: 1-107-22943-X.

[6] Weisstein, E. W. *Fourier Transform. From MathWorld–A Wolfram Web Resource*. URL: https://mathworld.wolfram.com/FourierTransform.html (visited on 03/08/2023).

[7] Cooley, J. W. and Tukey, J. W. An Algorithm for the Machine Calculation of Complex Fourier Series. *Mathematics of Computation* 19.90 (1965), pp. 297–301. ISSN: 00255718, 10886842. URL: http://www.jstor.org/stable/2003354 (visited on 03/08/2023).

[8] Stuchi, J. A., Boccato, L. and Attux, R. Frequency learning for image classification. *arXiv.org* (2020). ISSN: 2331-8422. URL: https://arxiv.org/pdf/2006.15476.pdf.

[9] Dung, C. V. and Anh, L. D. Autonomous concrete crack detection using deep fully convolutional neural network. *Automation in construction* 99 (2019), pp. 52–58. ISSN: 0926-5805.

[10] Kundu, R., Das, R., Geem, Z. W., Han, G.-T. and Sarkar, R. Pneumonia detection in chest X-ray images using an ensemble of deep learning models. *PloS one* 16.9 (2021), e0256630–e0256630. ISSN: 1932-6203.

[11] 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.

[12] Krizhevsky, A., Sutskever, I. and Hinton, G. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems* 25 (Jan. 2012). DOI: `10.1145/3065386`.

[13] Simonyan, K. and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv.org* (2015). ISSN: 2331-8422.

[14] He, K., Zhang, X., Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2016-. IEEE, 2016, pp. 770–778. ISBN: 9781467388511.

[15] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*. Vol. 2017-. 2017, pp. 5999–6009. URL: `https://arxiv.org/pdf/1706.03762.pdf` (visited on 03/24/2023).

[16] Dai, Z., Liu, H., Le, Q. V. and Tan, M. CoAtNet: Marrying Convolution and Attention for All Data Sizes. *arXiv.org* (2021). ISSN: 2331-8422.

[17] Franzen, F. and Yuan, C. Visualizing Image Classification in Fourier Domain. *The European Symposium on Artificial Neural Networks*. 2019. ISBN: 978-287-587-065-0. URL: `https://www.esann.org/sites/default/files/proceedings/legacy/es2019-66.pdf`.

[18] Ceci, M., Hollmén, J., Todorovski, L., Vens, C. and Dzeroski, S. FCNN: Fourier Convolutional Neural Networks. *Machine Learning and Knowledge Discovery in Databases*. Vol. 10534. Lecture Notes in Computer Science. Switzerland: Springer International Publishing AG, 2017, pp. 786–798. ISBN: 3319712489.

[19] Kiruluta, A. Reducing Deep Network Complexity with Fourier Transform Methods. *arXiv.org* (2018). ISSN: 2331-8422.

[20] Aggarwal, C. C. *Neural networks and deep learning : a textbook*. Cham, Switzerland: Springer, 2018. ISBN: 978-3-319-94462-3.

[21] Sharma, N., Jain, V. and Mishra, A. An Analysis Of Convolutional Neural Networks For Image Classification. *Procedia Computer Science* 132 (2018). International Conference on Computational Intelligence and Data Science, pp. 377–384. ISSN: 1877-0509. DOI: `https://doi.org/10.1016/j.procs.2018.05.198`. URL: `https://www.sciencedirect.com/science/article/pii/S1877050918309335`.

[22] Doon, R., Kumar Rawat, T. and Gautam, S. Cifar-10 Classification using Deep Convolutional Neural Network. *2018 IEEE Punecon*. 2018, pp. 1–5. DOI: `10.1109/PUNECON.2018.8745428`.

[23] Lin, H. W., Tegmark, M. and Rolnick, D. Why Does Deep and Cheap Learning Work So Well?: *Journal of Statistical Physics* 168.6 (July 2017), pp. 1223–1247. DOI: `10.1007/s10955-017-1836-5`. URL: `https://doi.org/10.1007%5C%2Fs10955-017-1836-5`.

[24] Nair, V. and Hinton, G. E. Rectified linear units improve Restricted Boltzmann machines. *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*. 2010, pp. 807–814. ISBN: 9781605589077.

[25] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. eng. *32nd International Conference on Machine Learning, ICML 2015*. Vol. 1. 2015, pp. 448–456. ISBN: 1510810587. eprint: `1502.03167`. URL: `https://doi.org/10.48550/arXiv.1502.03167` (visited on 04/02/2023).

[26] Murray, N. and Perronnin, F. Generalized Max Pooling. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 2473–2480. ISBN: 9781479951178.

[27] Baldi, P. and Sadowski, P. Understanding dropout. eng. *Advances in Neural Information Processing Systems*. 2013. URL: `https://proceedings.neurips.cc/paper_files/paper/2013/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf` (visited on 04/02/2023).

[28] 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* 15.56 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.

[29] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Jia, Y., Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[30] Chollet, F. et al. *Keras*. 2015. URL: `https://keras.io` (visited on 04/03/2023).

[31] Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C. and Oliphant, T. E. Array programming with NumPy. *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`. URL: `https://doi.org/10.1038/s41586-020-2649-2`.

[32] Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. (2009), pp. 32–33. URL: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.