Handwritten Sinhala Character Recognition Using Deep Learning

M. L. Karunarathne

198762H

Faculty of Information Technology
University of Moratuwa

July 2022

Handwritten Sinhala Character Recognition Using Deep Learning

M. L. Karunarathne

198762H

Dissertation submitted to the Faculty of Information Technology, University of Moratuwa, Sri Lanka for the partial fulfillment of the requirements of the Master of Science in Information Technology.

July 2022

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Name of Student	Signature of Student
M. L. Karunarathne	*UOM Verified Signature* …….. Date:04.08.2022

Supervised by:

Name of Supervisor	Signature of Supervisor
Dr. Chaman Wijesiriwardane	…….. Date:

Dedication

This effort is dedicated to my family in appreciation of their support, unconditional love, and sacrifice.

Acknowledgements

I would like to acknowledge my supervisor Dr. C. P. Wijesiriwardana for his guidance, encouragement and coordination during the whole research and writing phase. I also like to express my gratitude to Dr. S. C. Premaratne and Dr. W. G. C. W. Kumara for their guidance and suggestions during the research. In addition, I should be grateful to Mr. D. H. M. Jayarathna, a friend of mine, for his support during the research project. A special thanks should go to my family for their unwavering support and understanding when undertaking not only this research but the entire course.

Abstract

Sinhala language is the national language in Sri Lanka. Sinhala alphabet includes 60 characters and is slightly complex compared to other languages like English. Around 25-30 researches have been done since 1990 regarding Sinhala handwritten character recognition. Handwritten Sinhala character recognition remains mostly unsolved in pattern recognition, due to many perplexing characters and excessive curves in Sinhala handwriting. The existing recognizers are also unable to provide acceptable performance for practical applications.

This research aims to enhance the performance of handwritten Sinhala character recognition by using a new approach focused on deep neural networks, which have recently given excellent performance in many applications. This research implements Convolutional Neural Networks (CNNs) and Gabor initialized Convolutional Neural Network (GCNN). In addition to that, it investigates the performance of the proposed network architectures when introducing the dropout. To apply Gabor initialized CNN, the effect of the parameters of the Gabor filter over the Sinhala character image dataset is also examined. Considering the effect of the parameter on the GCNN architecture, parameter values for the proposed GCNN architecture are determined. The training accuracy of the first CNN method is 96.33 % and the testing accuracy is 90.14%. According to the literature, this is the highest accuracy obtained for 60 Sinhala characters compared with primitive methods. This accuracy is obtained with the 0.5 dropout effect. The Gabor initialized CNN architecture provides 95.15% training and 80% testing accuracy. Even though the training accuracy is approximately 1% less than the training accuracy of the first CNN architecture, it converges to the results rapidly. So, it saves time and computational cost.

Considering the results of implemented CNN architectures and Gabor initialized CNN architecture, the best-performing architecture is selected for the Sinhala handwritten character recognition process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Handwritten character recognition (HCR) has recently become a challenging research space in the pattern recognition discipline, which provides both educational and commercial value. The primary uncounted problem in the HCR process is the diversity of handwriting patterns of various writers in different languages. There are various word writing techniques in some of the sophisticated handwriting. Some languages, including Thai and Japanese, use isolated characters when writing words. In some languages like Sinhala, English, and Bengali characters are curved, complex and sometimes the characters overlap with each other. When compared with printed characters, recognition of handwritten characters is more difficult because characters have different features, such the size and shape, they are not all the same. Additionally, the different writing styles of different characters make the task of recognition difficult. Character recognition task also faces difficulties due to the similarities in different character shapes, character overlapping, interconnections of the adjacent characters. Furthermore, different writers' writing styles, Complicated character traits and similarities in different characters cause many issues in the recognition process.

Deep learning is a subsection of the machine learning domain that emulates the process of humans gaining certain types of knowledge. Recently, Deep learning has demonstrated exceptional performance in the field of pattern identification. Deep Neural Networks (DNN) comprise Deep Belief Networks, Stacked Auto-Encoder and Convolutional Neural Networks. Due to the combination of feature extraction and classification layers in DNN it is more efficient for learning. On the other hand, the majority of deep learning approaches use raw photos as inputs and perform image normalization without the requirement for a feature extraction process. DNNs' low and middle layers extract the features from the image input and perform classification on the extracted features.

The Sinhala words are two-dimensional compositions of Sinhala character symbols. The writing method is syllabic, and the script is written from left to right. Characters in a word are written separated from each other in a non-cursive manner. It is difficult to improve performance with a straightforward strategy since the Sinhala alphabet contains

many characters with similar shapes. Sinhala character recognition has a wide range of useful applications, including automatic number plate identification for automobiles, vehicle parking management, postal service automation, internet banking and many more.

1.2 Background

There are 60 letters in the Sinhala script, both vowels and consonants. The Sinhala alphabet uses horizontal lines from left to right to write its characters, which are typically circular, without any vertical or horizontal lines. Although each character in the Sinhala alphabet is distinct in shape, there are some characters that appear to be very similar but actually differ slightly. The Sinhala language uses more than 15 different types of modifiers. The combination of each letter of the Sinhala alphabet with those modifiers will produce more than 1000 letters that can be pronounced in many different ways. In contrast to the English language, there are no distinctions between simple and block capital letters.

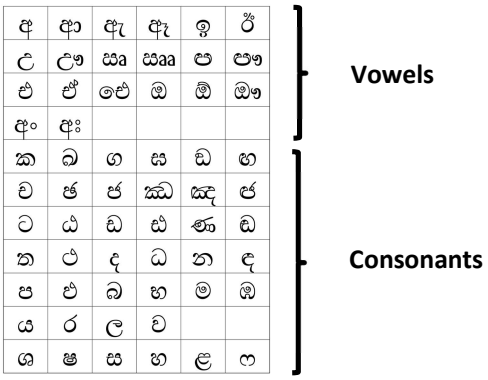


Figure 1.1: Sinhala alphabet

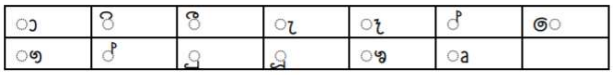


Figure 1.2: Modifiers used in the Sinhala language

There are considerably many similar shaped characters in the Sinhala language. Because of the misinterpretation of remarkably similar shape elements of a similar collection of characters, individual Sinhala character recognition is a difficult and complex procedure. Even though a significant study has been done on this subject since the 1990s, there is still no Sinhala handwritten identification system in Sri Lanka that achieves a high level of accuracy. This research mainly aimed to find an accurate and effective approach for

Sinhala HCR by integrating deep learning techniques with some image processing techniques for the entire Sinhala alphabet.

1.2.1 Evaluation and Categorization of Sinhala language

The Sinhalese writing method used in Sri Lanka is a Brahmi-derived syllabic writing method that includes vowels, punctuation marks, consonants and a unique symbol structure. Consonant-vowel combinations are written as segments in syllabic writing, which is a segmental system of writing where each segment is developed on a consonant letter and vowel notation is only used secondarily. Both Sanskrit and Pali languages appear to have influenced the Sinhala writing system in the Anuradhapura period. New sounds were introduced to the language as words were taken into Sinhalese both as derivatives and in the pure form. On the other hand, the Sinhala language reads left to right, descended from the Brahimi script. Sinhala script is also used for writing Pali and Sanskrit [33]. As discussed, Sinhala is a Brahmic script that holds many similar characteristics with the other languages, including Kannada, Malayalam, Telugu, Tamil, and Devangar. [33]. "The presence of a set of five nasal sounds known as half-nasal or prenasalized stops distinguishes Sinhala from its Indo-Aryan sister languages" [33].

"Sinhala characters are divided into two groups. The uddha sihala alphabet (Pure Sinhala) is formed by the basic set of letters, which is a subset of the mira sihala alphabet (Mixed Sinhala)" [34]. The ancient Sinhala alphabet contains vowels and Consonants [34]. Vowels are classified into two forms: independent and diacritic. When a consonant is not immediately followed by a vowel, such as at the beginning of a word, the independent form is used. The diacritic form is used when a vowel follows a consonant. There are various locations where the diacritic can be applied, depending on the vowel. "The śuddha alphabet includes eight plosives, two fricatives, two affricates, two nasals, two liquids and two glides" [34]. Furthermore, "the two graphemes for the retroflex sounds ඟ and ළ, which are not phonemic in contemporary Sinhala but nonetheless make up the set" [34].

1.3 Motivation

The method of handwritten character recognition enables the transformation of various handwritten document types into editable, accessible, and analyzable documents. The ultimate goal of the HCR process is to illuminate human invention in reading in such a way that the computer may read, update, and engage in documents as a person. The

invention of the HCR mechanism has obtained notable attention from many researchers over the past, and many significant achievements have been made in this domain [1]. “HCR is still a difficult procedure, though, because of the wide range of handwriting styles, the abundance of similar-looking characters, and the sheer amount of character types” [2].

CNN is a well-established deep learning model that takes its cues from how the human brain ordinarily interprets visual data. deep learning architecture that is inspired by how the human brain naturally processes visual information. CNN is capable of carrying out a variety of tasks, including scene tagging, object detection, movement recognition, visual saliency monitoring, position estimation, and voice and human language processing. CNN architectures and initialization techniques come in a wide variety, although their basic structure is relatively similar. The main reason why the CNN approach was applied for this study is its defined architecture and most importantly, there is no need for feature extraction. The main impression of CNN is that it utilizes convolution of images and filters to create invariant features which are transferred to the next layer. In addition to that, deep convolutional networks such as CNN have good flexibility and work well on image data. The key benefit of CNN over other neural networks is that it recognizes significant features automatically without human intervention. In addition to that, CNN required less preprocessing of data hence reducing the computational cost. CNN actually provides superhuman accuracy for most of the applications and can be implemented on any platform.

“Gabor filters are used for edge detection, picture pre-processing, texture analysis, feature extraction, and feature modification in the field of image processing” [21]. Gabor filters behave like human vision systems and are good for texture analysis tasks, separating texture information from images, hence used in a variety of applications including face, speech, and vein recognition. To detect the information present in an input image, a CNN architecture essentially learns the filter structure required. Although CNN filters are capable of extracting sophisticated and higher-dimensional features, they have the drawbacks of overfitting and hindering generalization capability for a relatively low number of training samples. As discussed, Gabor filters work well for extracting texture information, thus a Gabor filter bank will be used instead of a trainable convolutional layer as the CNN's initial layer. Developing a non-trainable Gabor filters bank as the initializer of the first layer in CNN provides some outstanding benefits. “It

exhibits superior performance when trained on small datasets, prevents overfitting, and provides greater generalization on unseen data" [22]. According to the previous research, "Gabor filters merged with CNN layers provide accurate results with significant improvement in convergence in the case of handwritten character recognition applications" [16]. "It has been demonstrated that the Gabor initialized CNN model improved performance in terms of high sensitivity and specificity because of the robustness of the Gabor filter" [21]. Even with noisy input and limited training data available, Gabor initialized the CNN model greatly performed when compared to other initialization techniques.

1.4 Aim and Objectives

Aim: The aim of the research is to develop a model to recognize and interpret Sinhala handwritten characters using deep learning techniques.

Objectives:

1. To develop the following models using deep learning techniques to recognize the handwritten Sinhala characters.
   - Convolutional Neural Networks (CNN)
   - Convolutional Neural Networks with dropout
   - Convolutional Neural Networks with dropout and Gabor filters
2. To carry out a comprehensive comparison of the above three different approaches to find out the best approach to recognize the Sinhala handwritten characters.

1.5 Summary

This chapter mainly focuses on the background, aim and objectives of the research project. Chapter 2 discusses the literature review which describes prevailing solutions obtained by previous researches for the specified problem and draws a comparison between them and the proposed method. Chapter 3 contains the details of the technology being adopted within the proposed method. Chapter 4 contains details regarding the use of that technology within the proposed system. Chapter 5 describes the test approach and discussion. Finally, a conclusion and further work to be done is given in Chapter 6.

Chapter 2

Literature Review

2.1 Introduction

Sinhala handwritten characters are unpredictable compared with typewritten or printed characters. Handwritten letters vary from person to person and even sometimes, it varies with the same writer. As Sinhala letters are comparatively complex, handwritten Sinhala character recognition becomes a more interesting and challenging area of research. This chapter discusses the previous work done by researchers on the Sinhala character recognition areas. Currently, some of them have been implemented.

2.2 Review of other's work

Handwritten character recognition is an artificial intelligence application that belongs to the scientific discipline called pattern recognition. Pattern recognition is a complicated task that applies complicated logic and theories with it, and it requires much more effort to improve its accuracy and performance. It is more difficult to recognize Sinhala handwritten characters compared to English characters. There are many reasons for those inconsistencies and a few listed below [12][13];

- Usual curveness shape of the letter
- Size of the character
- Shape of the character depends on the speed of writing
- Age of the writer
- Literacy of the writer
- Personal writing styles.

Hence Sinhala handwritten characters vary from writer to writer and even sometimes with the same writer. As Sinhala characters are more complicated with some similarities to some letters, the handwritten Sinhala letter recognition process becomes a more complicated and challenging area of research.

There has been some research carried out on this topic since 1990. Many approaches have been used for Sinhala handwritten characters recognition, such as Convolution Neural Networks (W.V.S.K.Wasalthilake and T.Kartheeswaran,2020)[3], Neural Networks (Rohana K. Rajapakse and Seneviratne, 1995)[2], Hidden Markov Model

(Hewavitharana et al., 2015)[1], Projection file methods(Nilaweera, Premeratne, and Sonnadara, 2014)[4], Contour tracing method(Silva, Jayasundere and Kariyawasam, 2016)[5] and some other were similar as the above methods.

Most of the above researchers used their own collected datasets with different suitable processing techniques. All those existing researches had many drawbacks such as:

- Models were giving results for a limited amount of Sinhala characters
- Used complex algorithms
- Less accuracy
- Models were not suitable to recognize touching characters

The proposed research mainly contains objectives to address those drawbacks and planning to obtain higher accuracy levels.

2.2.1 Research on Foreign languages

In this section HCR researches carried on foreign languages such as Tamil, Tai, Japanese are discussed with their corresponding algorithms, accuracy levels, strength and limitations.

Paper	Language	Algorithms	Accuracy Level	Strength and Limitation
[7]	Tamil	• CNN	Training accuracy 95.16% Validation accuracy 92.74%	• Dropout is used • 156 classes • Model unable to recognized similar characters
[8]	Tai Lü language (No strokes as horizontal or vertical strokes)	• Gabor filters • PCA and SVM	Recognition rate is 87.08%.	• Preprocessing approaches are used. • constructed a bank of Gabor filters with various sizes and angles • PCA is applied to reduce dimensions • SVM is the classifier that perform recognition task
[9]	Hiragana and katakana (Japanese)	• CNN and SVM methods • Dropout	CNN- 87.82%. CNN and SVM - 88.21%.	• Preprocessing used • CNN as a tool for feature extraction and SVM as a tool for character identification

[10]	Tai Le Language	• DNN and logistic regression algorithms	98.85% accuracy	• CNN and ensemble learning are used in conjunction. • Dropout is used • Addressed the problem of recognizing numerous similar characters.
[11]	Bangla numerals	• Deep Belief Network • CNN • CNN with dropout • CNN with dropout and Gaussian filters • CNN with dropout and Gabor filters.	• DBN-:97.20% • CNN & Gaussian -:97.70% • CNN & Gabor-:98.30% • CNN & Gaussian & Dropout -:98.64% • CNN + Gabor+ Dropout -:98.78%	• Six model are implemented to select the best

Table 2.1: Foreign language handwritten character recognition research

2.3 Summary

In this chapter, the different approaches that have already been considered by different researchers in this area were discussed.

Chapter 3

Technology adapted

3.1 Introduction

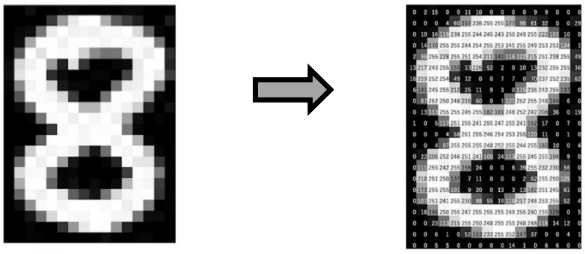
This chapter will discuss different tools, methodologies, and techniques that have been applied to implement the models to recognize the Sinhala characters. Two convolutional models and a Gabor filter initialized CNN model were proposed. The primary components of the model were implemented using Python, and Jupyter Notebook is the Integrated Development Environment (IDE) that is used for coding. In order to obtain a brief understanding of the technologies and techniques utilized to develop the models, it is essential to understand fundamental concepts. Therefore, this chapter will provide some insight into such technologies and their usage in this research project.

3.2 Image Representation

When considering image classification and recognition, it is important to study how the computers see images & identify the issues faced while performing a computer vision task. An image can be described as a numerical matrix in the computing field. These specific numbers are represented by varying light brightness that altogether constructs the picture. A pixel is a set of numbers within a matrix that is arranged in rows and columns. The data is represented by the pixel values, which can range from a lowest to a highest value. The values of the pixel light intensity are often expressed as an 8-bit value, or a number between 0 and 255.

The intensity increases with a larger pixel value, making the pixel look brighter. A colored image is made up of matrices with three distinct red, green, and blue channels that overlay one another. When it comes to image processing tasks, colour images require more computation than grayscale images due to the fact that there are three streams to process on instead of one stream. It is possible to manipulate and examine a picture or an array of pictures using computation methods since an image is represented as a structure made up of matrices filled with intensity values. When considering the image of handwritten number 8, since the image is made up of pixels and is grayscale, each pixel can be characterized by a single value, allowing the image to be represented as a two-dimensional matrix of integers, one for each pixel in the image. The RGB color

image can be shown as three two-dimensional images concatenated on top of one another rather than as a grayscale image.



Input Image Pixel Representation

Figure 3.1: Pixel representation of number 8

3.3 Deep learning approach

Artificial neural network (ANN)-based techniques are the focus of the machine learning subfield known as deep learning. The ANN can learn from vast amounts of data and mimic the activity of the human brain. A DNN uses a combination of data inputs, weights, and biases to simulate the human brain. These variables are applied to accurately identify, categorize, and characterize items in the data. Deep neural networks use several, interconnected layers, each building on the one before it, to more accurately and efficiently forecast or categorize data. The input and output layers are observable layers where the input layer absorbs the data for processing, and the final prediction or classification is made by the output layer. DNN comprises Deep Belief Networks, Stacked Auto-Encoder and CNNs. Because of the combination of feature extraction and classification layers in DNNs, it is more efficient for learning. In contrast, the majority of deep learning techniques utilize raw photos as inputs and perform image normalization without the requirement for a feature extraction procedure. Deep learning has recently demonstrated its exceptional capabilities in the areas of pattern recognition and machine learning.

3.4 Convolutional neural networks (CNNs)

The main advantages of CNNs are their usage in image classification and computer vision. They are able to identify details and patterns in an image. Fukushima made the initial suggestion for the CNN structure in 1980[14]. A gradient-based learning method was successfully applied to CNN by LeCun et al. in the 1990s [14]. In addition, fundamental benefits of DNNs, CNN also has additional advantages, such as being

designed to simulate how the human eye works. Additionally, it has extremely efficient designs for learning to extract and abstract two-dimensional characteristics. The CNN model's max-pooling layer is excellent at detecting shape variations. Furthermore, CNN uses less parameters than a fully connected network of equivalent size. Additionally, CNN can also be trained using the gradient-based learning method. As a result, it has less difficulty with the vanishing gradient problem N can create highly tuned weights when using the gradient-based method to train the complete network to directly avoid errors. CNNs were recently used to recognize handwritten characters and produced the best recognition accuracy.

The spatial dependencies in a picture can be acquired by a CNN and linked to the image's content for recognition purposes. “To understand the structure of the image during the training stage, the weights, parameters, and biases are present in the transformations of the initial image to feature vector” [15]. In order to classify a picture, the CNN structure offers a remarkable match of the original image to differentiating features. Convolution, pooling, and fully connected are the key layers that compose CNN, and the process of CNN is illustrated in Figure 3.2.

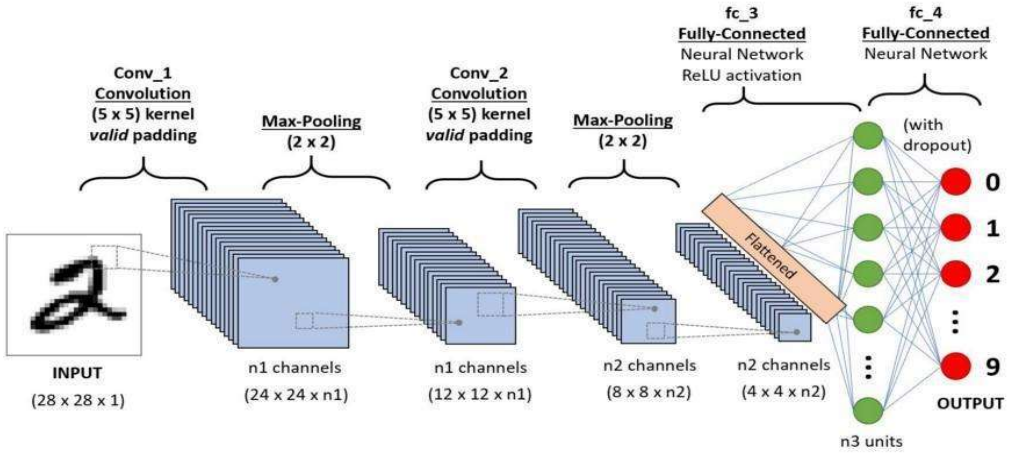


Figure 3.2: Convolutional neural network

3.4.1 Convolution layer

The process of convolution is used to sharpen and enhance features in images. “In the process of convolution, an image is passed over a single matrix of digits known as a kernel or filter to shape the image with the filter” [16]. To create one value for the filtered image, each chosen pixel value from the image sub-matrix must be multiplied by the

appropriate pixel value taken from the kernel matrix and then added together [15]. This operation should be carried out continuously for the whole image. Convolution can repeatedly grab all of the low-level features, including edges, color, gradients, and position. The objective of a convolution layer is to eliminate the strong features as edges of the source image. When compared to the input, the technique has the effect of reshaping the characteristics to reduce dimensionality [17].

3.4.2 Pooling layer

Pooling is an important layer of CNN as well. The goal of this layer is to gradually increase the spatial proportions of the image representation in order to reduce the network's computational difficulty. Pooling can be implemented using a variety of non-linear functions, including maximum, adaptive, minimum and average pool. In the CNN architecture, the maximum pool (Max-Pooling) technique is the most used one. It conserves computational power by diminishing dimensionality while retaining the important properties, which are translational and rotational variants that preserve the model's effective training phase.

Max-Pooling provides the highest value derived from its image area enclosed by the kernel. However, average pooling downregulating noise is better. It eliminates the noisy inputs and performs de-noise coupled with dimensionality reduction [16]. By minimizing the number of variables, the memory consumption, and data processing in the network, the pooling layer allows control overfitting. Introduce a pooling layer among consecutive convolutional layers activated by a function is executed in the CNN architecture. With the exception of a kind of global pooling that promotes local translation invariance, pooling layers do not help a CNN achieve global translation invariance [15]. As illustrated in Figure 3.3, a widely attractive max-pooling layer of size 2×2 is used with a stride of 2, subsampling each input depth slice by two on the both width and height and rejecting seventy-five percent of the activations.

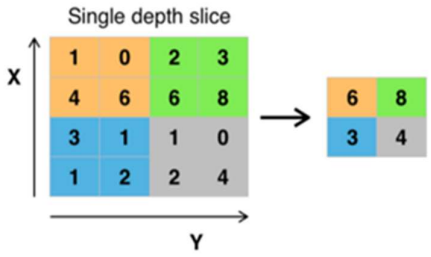


Figure 3.3: Max Pooling

3.4.3 Fully connected layer

Fully connected layer is applied at the final stage of CNN. During the training part, weights related to the CNN are determined. Fully connected layers accept the convolutional and pooling layers' end results to determine the best-matched identifier that reflects the original image. “This layer established the relationship between the image feature vector and the image class. The weights related to the network connection path are multiplied by the outputs obtained from the convolution or pooling operation” [14]. After that, the outcome is processed through an activation function. The process of a fully connected layer is shown in Figure 3.4.

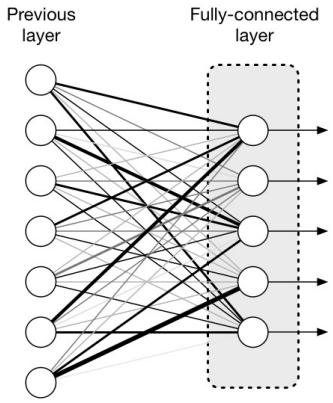


Figure 3.4: Fully connected layer

3.5 Regularization Techniques

The primary purpose of a good deep learning model is the capability to extend the training data to any problem-related data and facilitates making accurate predictions on the data that the model has never reached. Generalization refers to how well the model has learned the perception to use for any data rather than just with the specific trained data during the training process. If the deep learning model is not generalized, a problem of overfitting emerges [18]. The deep learning model ultimately performs poorly on

unobserved data because it begins to learn the features and noise from the training data. In other words, as the model becomes more complicated, the training error decreases but the validation error does not, as shown in Figure 3.5 and more tends to overfitting. Regularization is a technique that facilitates avoiding overfitting in convolutional neural networks and hence improving the accuracy of deep learning models when it provides completely new data. There are many commonly used regularization techniques such as:

- L2 & L1 regularization
- Early stopping method
- Dropout method
- Data augmentation

For this research, a dropout regularization method is applied to avoid model overfitting.

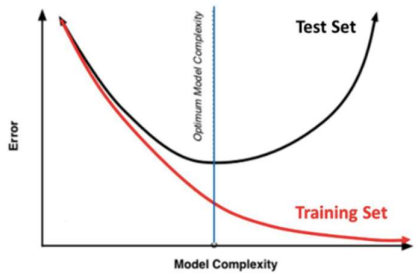
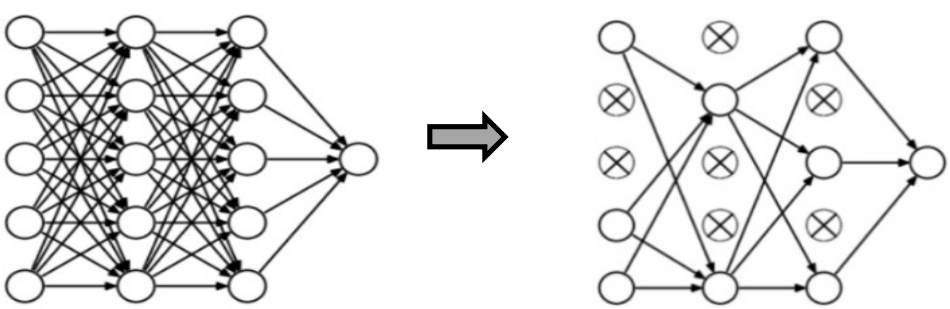


Figure 3.5: Test set error vs training set error

3.5.1 Dropout for regularizing

Dropout is a straightforward and effective regularization technique that can be applied to neural networks and deep learning models. Deep learning neural networks are expected to overfit quickly as fully connected layers engage in most of the parameters. Aggregation of neural networks with various model architectures is facilitated to reduce overfitting, although this requires the extra computational effort of training and managing several models [18]. Dropout, which involves randomly removing nodes from a network during training, offers a very efficient regularization strategy to minimize generalization error and lessen overfitting in deep neural networks. By preventing the training of all nodes on the training data, dropout reduces the overfitting rate. This method remarkably upgrades training rate also. “The dropout technique helps to reduce the inter-node interactions by facilitating the learning of more powerful features that better generalize to unknown data” [19].

Dropout refers to the concept of removing neurons from the hidden and observable layers, as well as the input layer. This process is carried out by temporarily eliminating a neuron together with its inputs and outputs, as shown in Figure 3.6. By adjusting the rate at which these neurons are eliminated, dropout can likewise be employed as a hyperparameter. As discussed above, the main objective of applying dropout regularization is to minimize model overfitting by separating neuronal interactions among layers, which reduces generalization problems. When considering the previous research, “dropout has been discovered to be a standard method that is not domain specific, and it helps to improve training accuracy by breaking up the interaction within the network's hidden layers” [18]. Applying the dropout effect to the model makes each neuron in the layer independent of other neurons in the layer.



Standard Network After applying Dropout

Figure 3.6: Appling Dropout

In previous research, it was described that applying dropout can obtain state-of-the-art results. It is also shown that results can be optimized by using a dropout rate between the range [0.4, 0.8] [17]. But, when applying the dropout effect on smaller datasets, the benefits were not much as expected. The main drawback of the dropout effect is that it lengthens training times and increases computation requirements for networks, forcing a trade-off between speed and accuracy. Additionally, dropout can be used to enhance test accuracy. Although increased training time has a disadvantage, it significantly increases accuracy, particularly for large data sources, which is more important.

3.6 Gabor filters

Dennis Gabor invented the Gabor filter in 1946 [20]. “Gabor filters are bandpass filters that are used in image processing to extract features, analyze texture, and estimate stereo disparity” [20]. To extract various textures, edges, and features the Gabor filter is used.

The Gabor filter examines an image in order to acquire any particular frequency content in appropriate direction in a confined area close to the analysis point. Gabor filters are created by combining a Gaussian kernel function with a sinusoidal wave. For instance, a 2-D Gabor filter is a sinusoidal plane wave modulating a Gaussian kernel function. Equation (1) provides the complex equation for the Gabor filter, while Equations (2) and Equations (3) illustrate how the waveform's real and imaginary components combine. The Gabor filter's core frequency, which exhibits the maximum response, is controlled by Equations (4) and (5).

Complex

$$g(x, y;\ \lambda, \theta, \psi, \sigma, \gamma) = \left(\frac{-x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) exp\ (i\left(2\pi \frac{x'}{\lambda} + \psi\right)) \text{ ---------- (1)}$$

Real

$$g(x, y;\ \lambda, \theta, \psi, \sigma, \gamma) = \left(\frac{-x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) cos\ cos\ \left(i\left(2\pi \frac{x'}{\lambda} + \psi\right)\right) \text{ ---------- (2)}$$

Imaginary

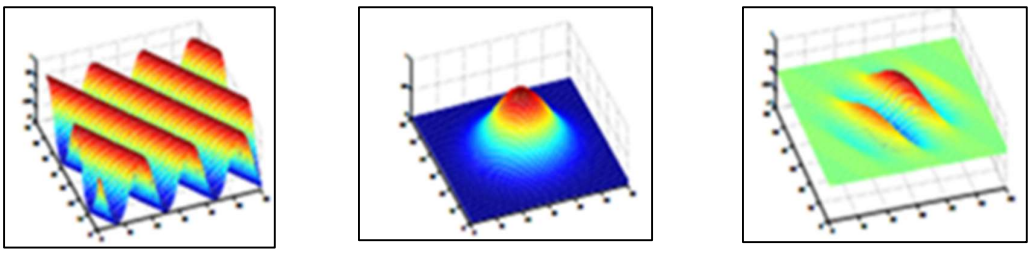
$$g(x, y;\ \lambda, \theta, \psi, \sigma, \gamma) = \left(\frac{-x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) sin\ (i\left(2\pi \frac{x'}{\lambda} + \psi\right)) \text{ ---------- (3)}$$

where

$$x' = xcos\ \theta + y\ sin\ sin\ \theta \text{ ---------- (4)}$$

$$y' = -xsin\ \theta + y\ cos\ cos\ \theta \text{ ---------- (5)}$$

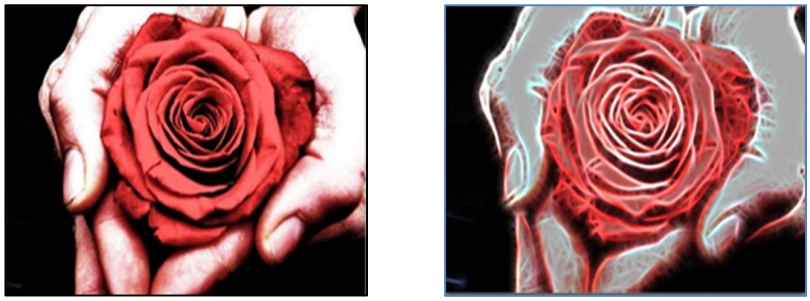
“The power spectrum of the Gabor filter is formulated by two impulses of a sine wave and a Gaussian. Multiplying these in the spatial domain is known to be a convolution in the frequency domain” [6]. The filter employs numerous parameters that must be adjusted and altered in order to retrieve feature information. These parameters give the filter unique characteristics and produce different results on the image. In these equations, λ denotes the wavelength of the sinusoidal factor, θ represents the direction of the normal to the parallel stripes of a Gabor function, ψ is the phase offset, σ represents the standard deviation of the Gaussian envelope and γ denotes the spatial aspect ratio and specifies the ellipticity of the Gabor function. With many parameters, it would be difficult and time-consuming to determine an optimal solution for small or large datasets.



Sinusoid oriented 30^0 with x axis | 2 D Gaussian | The related 2 D Gabor filter

Figure 3.7: 2 D Gabor filter

According to Figure 3.7, it can be concluded that the sinusoid has been spatially localized. A set of Gabor filters with different orientations should be used while studying texture or extracting information from an image [32]. When Gabor filters are applied to an image, it acts in a similar way as conventional filters. The highest extraction occurs along edges and locations where texture changes when a Gabor filter is applied to an image. Figure 3.8 demonstrates the actual image and the image after applying a particular bank of Gabor filters. As discussed, Gabor filters react to edges and texture changes of the image which means the filter has a unique value at the feature's spatial position. The results of passing the image of a white circle in the black background through each filter in the filter bank are shown in Figure 3.9. It explicitly states that the circle's deleted edge is the edge that is inclined at the same angle as the Gabor filter.



Sample input to the Gabor filter | Output after applying Gabor filters

Figure 3.8: Image after applying particular bank of Gabor filters

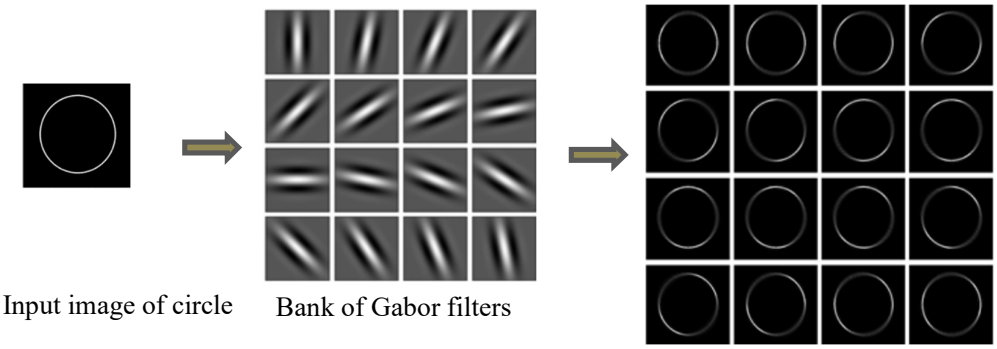


Figure 3.9: Output when passing through each Gabor filter

3.6.1 Gabor Kernel Size, *ksize*

If *ksize* = (*m, n*), then Gabor kernel of size *m* x *n* pixels. When changing *ksize*, the size of the convolution kernel is also changing. It can be seen that the convolution kernel's size has no impact on the output image when the parameter *ksize* is changed while retaining the kernel square and of an odd size. Considering that scaling the kernel's size is equivalent to scaling the size of the image, this further shows that the convolution kernel is scale-invariant.

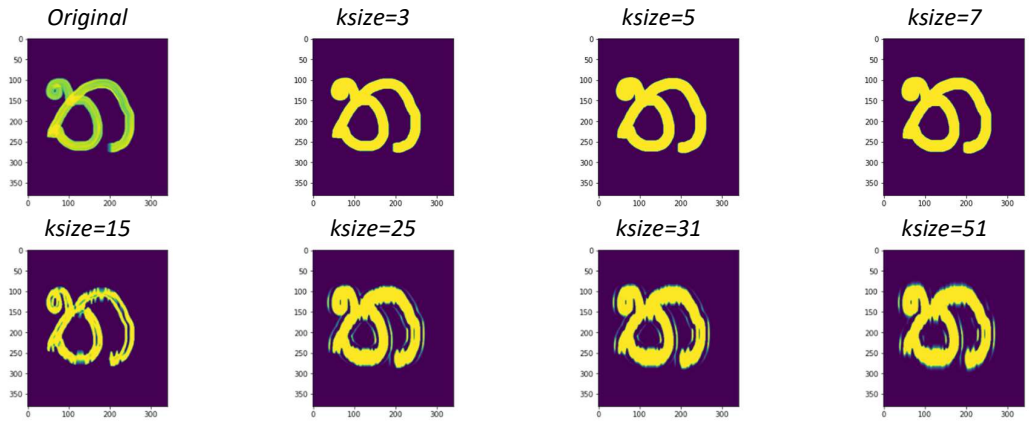


Figure 3.10: Gabor filtered images for different ksize

Figure 3.10 illustrates a few results with different *ksize*. For all the following images, $\sigma = 4.0$, $\theta = 0$, $\lambda = 10.0$, $\gamma = 0.5$, $\psi = 0$. ($[\sigma, \theta, \lambda, \gamma, \psi] = [4.0, 0.0, 10.0, 0.5, 0.0]$). *ksize* = [3, 5,7,15, 25,31,51] for each occurrence. These findings demonstrate that the image seems to discover specific textures with smaller kernels whereas large items tend to be found with larger kernels. Using a larger scale image may show different output, and at some point, it may not see the difference in the image filtered with a larger kernel size.

Actually, this is the case when only the Gaussian blurring effect is visible as the filter size increases because the filter closely matches the input image [23].

3.6.2 Standard Deviation, σ

This parameter controls the width of the Gaussian envelope used by the Gabor kernel. It controls the spread within the Gaussian function. To illustrate the behavior of the σ, value of σ is changed to [10.0, 30.0, 45.0] while maintaining other parameters unchanged as [$\lambda = 30$, $\theta = 0.0$, $\gamma = 0.25$, $\psi = 0$] and results are shown in Figure 3.11. It is concluded that when changing the sigma value from small to larger, the Gabor function has more stripes overall. A few of the results produced by changing this parameter are shown in Figure 3.12. From left to right, in this test σ = [1.0, 2.0, 3.0, 4.0, 5.0]. The values of the fixed parameters *ksize*, θ, λ, γ, and ψ are [15×15, 0.0, 10.0, 0.5, 0.0] correspondingly. It appears that the image gets blurrier.

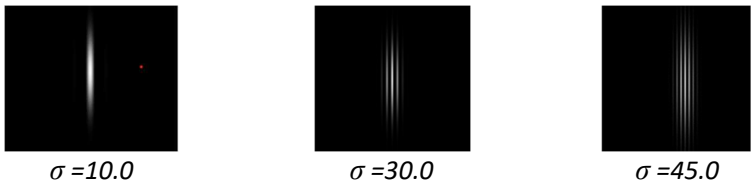


Figure 3.11: Gabor filters for different σ values

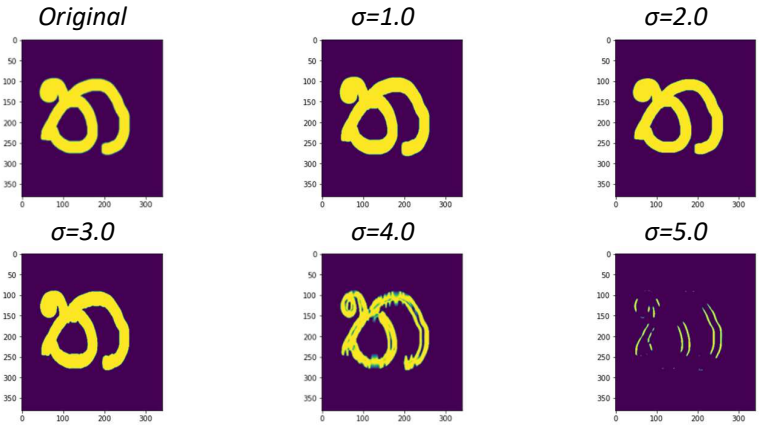


Figure 3.12: Gabor filtered images for different σ values

3.6.3 Orientation, θ

This is considered to be one of the most crucial Gabor filter parameters. This parameter determined the types of features to which the filter responds. θ is a state within the sin wave that describe distinct directions of the filter. Passing this variable through the waveform will show various features and extractions from the image. At some angles,

certain aspects of the image might not be extracted or might not be clearly apparent, but at other angles, those features might be disclosed. The features of the image can be exposed more effectively by merging the various image extractions. Figure 3.14 illustrates how the output of the image after applying the filter. In the Figure 3.13, the kernel outputs are obtained by changing the θ from 0.0 to 45.0 and 90.0 the Gabor function rotates on maintaining other parameters unchanged [$\lambda = 30.0$, $\gamma = 0.25$, $\sigma = 10.0$, $\psi = 0.0$].

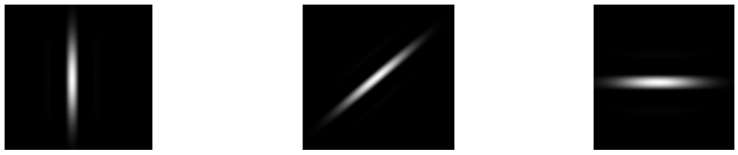
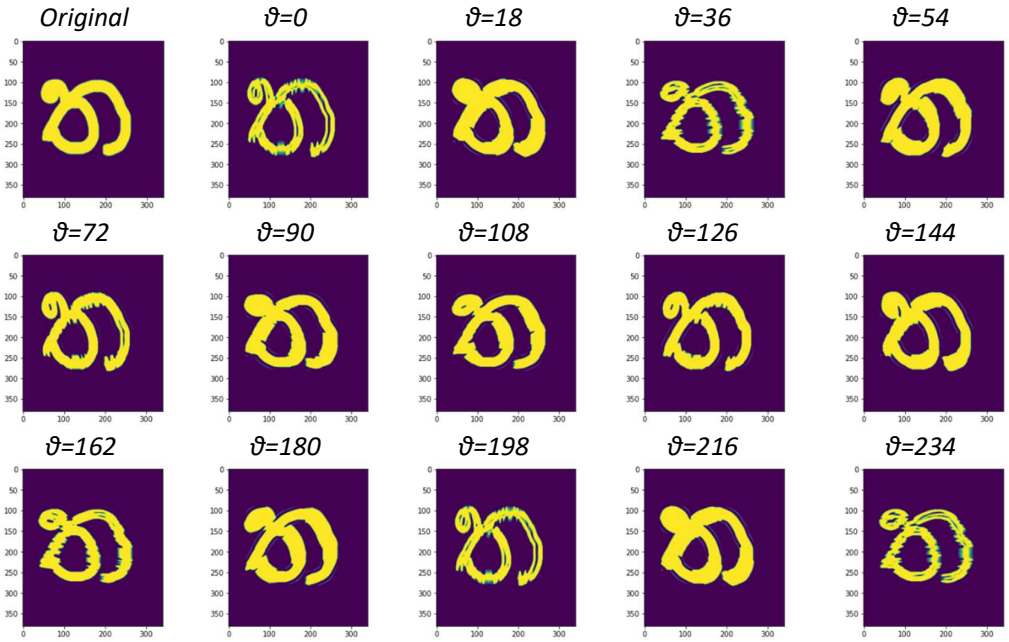


Figure 3.13: Gabor filters for different θ values

Giving θ is zero in Figure 3.13 demonstrates that the filter only responds to horizontal characteristics. Therefore, to gather all features at various angles in an image, it is required to generate filters with different θ values. In Figure 3.14, the θ range between 0 and 360 is divided into 20 equal parts, and creates a Gabor kernel for every θ value, thus obtaining the outputs. Given an input image of size 28 × 28, θ value changed as θ = [0, 18, 36, 54, 72, 90, 108, 126, 144, 162, 180, 198, 216, 234, 252, 270, 288, 306, 324, 342, 360]. The variables *ksize*, σ, λ, γ, and ψ are constant at [15 × 15, 4.0, 10.0, 0.5, 0.0] respectively. Noted that the image rotates around its central axis.



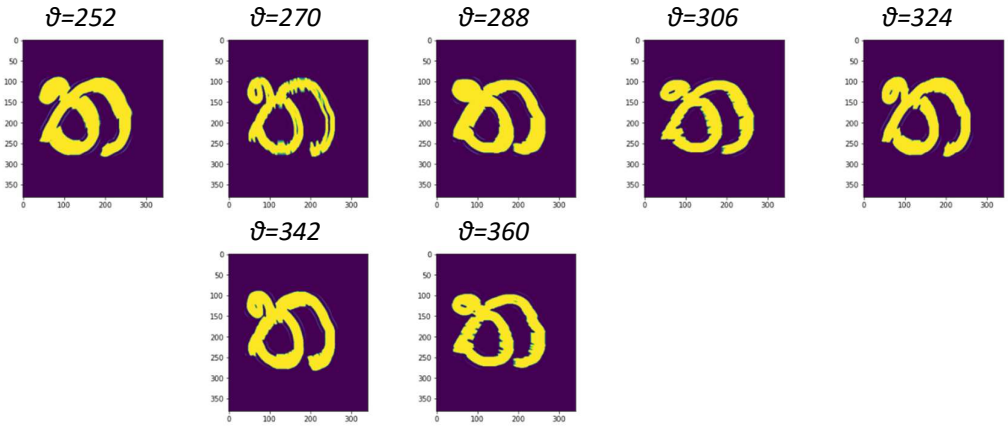


Figure 3.14: Gabor filtered images for different θ values

3.6.4 Wavelength of Sinusoidal factor, λ

Parameter λ governs the wavelength size inside the sinusoid. The parameter λ is responsible for the width of the Gabor filter's bars. Increasing the wavelength and having a greater λ will create a broader bar. Decreasing the wavelength, a smaller value of λ will produce a thinner bar. According to the previous research, the wavelength value needs to be less than one-fifth the size of the input image. [21]. As illustrated in Figure 3.15, when λ increases, the kernel becomes larger in width. In here parameter λ changed to λ = [5.0, 10.0, 15.0] while holding other variables unchanged [θ = 0.0, γ = 0.25, σ = 10.0, ψ = 0.0].

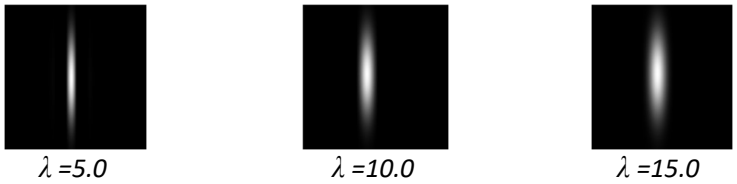


Figure 3.15: Gabor filters for different λ values

For a given input image of size 28 × 28, λ is changed as λ = [0.0, 5.0, 10.0, 15.0, 20.0] and other variables *ksize*, σ, θ, γ, and ψ are kept as constant at [15 × 15, 4.0, 0.0, 0.5, 0.0] respectively. The outputs shown in Figure 3.16 conclude that the image gets clearer as λ increases in value.

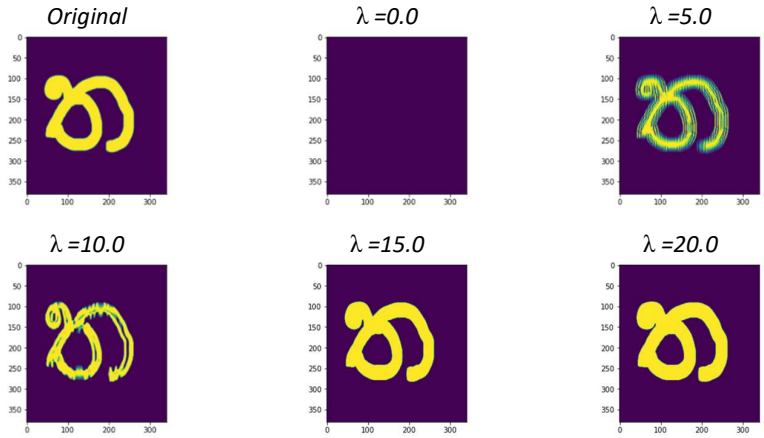


Figure 3.16: Gabor filtered images for different λ values

3.6.5 Spatial Aspect Ratio, γ

The parameter Gamma is responsible for the height of the filter. When the γ parameter value decreases to a lower value, the filter reaches the pixel size, and increasing the γ value will stretches the filter to the image size [22]. As a result, the height increases greater for extremely tiny gamma values and becomes very small for very high aspect ratios. The output of the Gabor filters obtained from keeping the parameter values unchanged as [$\lambda = 30.0$, $\theta = 0.0$, $\sigma = 10.0$, $\psi = 0$], and modifying the γ value from 0.25 to 0.5 and 0.75 are shown in Figure 3.17.

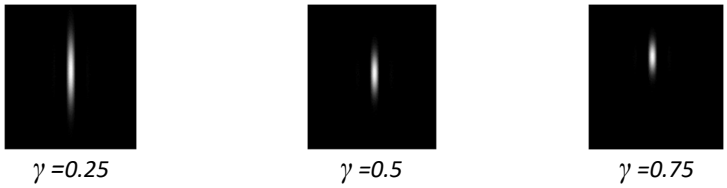


Figure 3.17: Gabor filters for different γ values

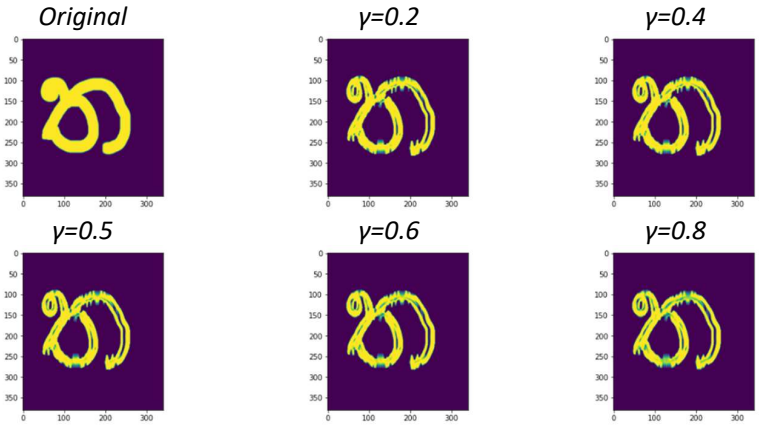


Figure 3.18: Gabor filtered image for different γ values

For a given input image of 28 × 28, the output of the Gabor kernels that are obtained by applying values for variables ksize, σ, θ, λ, and ψ as [15 × 15, 4.0, 0.0, 10.0, 0.0] sequentially and changing γ values to [0.2, 0.4, 0.5, 0.6, 0.8] is illustrated in Figure 3.18.

3.6.6. Phase Offset, ψ

Although ψ is not typically used, it is nonetheless required in specific applications. By setting ψ, kernel bands can be shifted either left or right. "Because Gabor filters are used in a bank, the number of filters in the bank generally cancels out the off-setting frequency of the sinusoidal" [20]. The Figure 3.19 illustrates the output of an image of 28×28 applying ψ changing as ψ = [0.0, 2.0, 4.0, 6.0, 8.0, 10.0, 12.0] while keeping other variables *ksize*, σ, θ, λ, and γ are constant at [15 × 15, 4.0, 0.0, 10.0, 0.5] correspondingly.

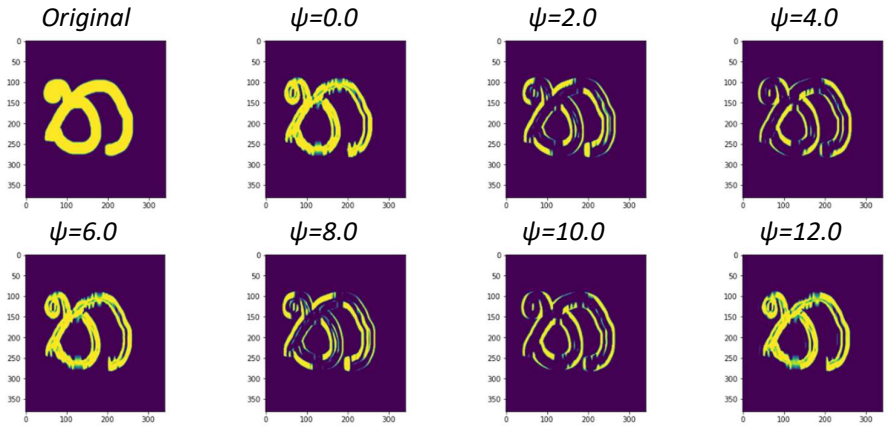


Figure 3.19: Gabor filtered images for different ψ values

3.7 Initialization approaches for CNNs

Researchers have attempted different approaches to enhance the learning stage of training process by changing or developing various methods to initialize the CNN. Some commonly used methods are discussed here.

Random search for hyper-parameter optimization

Hyperparameters are model configuration arguments defined by the developer to direct the learning process for a given dataset. Even through hyperparameters have a known consequence on a model, it has no proper way to set hyperparameters for a given dataset. So, it required to determine a set of hyperparameters that provide accurate performance for the model which called as hyperparameter optimization. The most popular optimization techniques for hyper-parameter optimization are grid search and random

search. “Grid search method defines a search space as a grid of hyperparameter values and estimates every position in the grid, whereas the random search algorithm defines a search space as a restricted domain of hyperparameter values and samples points in that domain at random” [23]. Grid search is often less accurate and less efficient than random hyper-parameter optimization [24]. Grid search algorithm is usually applied as an adequate method for determining optimal parameters but the computational cost to apply this technique is relatively high. Even though grid search is simple and comparatively reliable for low dimensional parameter space architecture, but when apply to a larger dimensional space for parameters, it experienced the curse of dimensionality problem. In addition to being effective and simple to use, random search algorithm outperformed grid search results in both minimum and maximum dimension spaces [23]. “It is found that while both grid search and random search required more attempts to achieve a satisfactory accuracy rate, random search outperformed grid search and required less iterations” [27].

Region proposed convolutional neural networks (R-CNN)

The concept of R-CNN is region proposals. “Images having bounding boxes that have already been localized and segmented were used in the R-CNN architecture” [27]. So, it used preprocessed input data to the convolutional neural networks. “The R-CNN architecture has been found to significantly improve accuracy” [27]. This might be viewed as an additional pre-processing or initialization technique for convolutional neural networks.

Median filtering forensics based on CNNs

Researchers looked into employing CNN networks to automatically extract characteristics rather than manually obtaining them as a result of CNNs being improved. “Chen invented a CNN network that contained the median filter as the first layer in the network” [26]. The main advantage of using a median filter is that it can detect non-linearities while conserving edge information. By applying the median filter by the side of the CNN, image borders and textures can be extracted earlier, allowing the CNN to learn features much easier. This method of developing a filtering layer for collection of data is an innovative way to separate essential features inside the images. Researchers

were able to achieve higher accuracy when implementing conventional CNNs with the median filter as a first layer when compared to earlier methods.

3.7.1 Gabor filter and CNN initialization

As discussed, the Gabor filter has the capability to extract edges and characteristics at various frequencies. Given that CNNs exhibit the same behavior in terms of extracting these characteristics, it is therefore conceivable to use the Gabor filter as a filtering method within them. The fundamental advantage of utilizing Gabor filters in CNN design is that the CNN provides the filter as a starting point rather than requiring users to learn it [31]. This suggests that CNNs have a solid starting point and thus save training time. But this advantage of reducing training time may not be valued for all situations, as the search method needed for obtaining the appropriate Gabor filter parameters may take more time than saving training time. Therefore, scientists have been looking for various methods to increase the accuracy and speed of these networks. “The techniques and tactics that researches are primarily interested in are pre-filtering images before feeding them into the CNN, modifying or changing layer structures by using dropout layers or pooling layers, try to minimize or increase the number of parameters or finding parameters that enable a CNN architecture to converge efficient manner, and using and implementing various types of filters” [23].

Many researchers are concerned about the capacity to apply and implement Gabor filters within the first layer and extend it to other layers within a CNN. “Modifying the initial layer with a Gabor layer was proposed in first research paper regarding the Gabor filters with neural networks published by Caldern et. al.” [23]. “Mr. Sarwar proposed in 2017 that using Gabor Filters implemented within the CNN architecture is beneficial by achieving equivalent accuracy results while being computationally efficient” [28]. “Ozbulak achieved higher accuracy using Gabor filters with the widely tested MNIST, CIFAR-10, and CIFAR-100 datasets” [29].

3.7.2 CNNs using Gabor filters

According to the R-CNN network and median filtered CNN network architectures, it is possible to realize that images can be filtered and then supplied into the CNN to achieve significantly better results. “Caldern introduced a CNN architecture by adding a Gabor feature extracting layer as the initial layer of the CNN” [23]. Caldern introduced the

GCNN in order to obtain the benefits of using Gabor filters to extract features and allowing these feature detectors to feed through a CNN and improve the accuracy and training time. “By building a CNN employing the Gabor filter in the first and second convolutional layers, Sarwar established another Gabor CNNs structure” [28]. The primary aspiration of this GCNN structure was to achieve reasonable energy savings regarding computation time by simplifying the CNN network by utilizing error persistence. “It's thought that particularly expensive operations like backpropagation, gradient computation, and weight changes within the layers” [25]. In the GCNN architecture, trained layers of CNN are replaced by fixed Gabor kernels and hence CNN can illuminate these expensive computation methods within the layers.

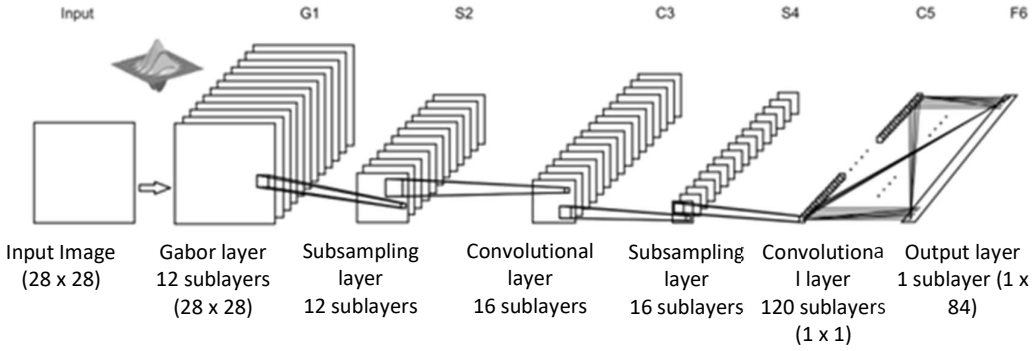


Figure 3.20: Implementation of Gabor filter

3.8 Tools used

3.8.1 Anaconda

For use in data science, machine learning systems, large-scale data computing, and predictive analytics, there is a distribution of Python and R called Anaconda. The distribution comes with data-science packages for Windows, Linux, and macOS. It contains a GUI called Anaconda Navigator. Package versions in Anaconda are managed by the package management system named conda. Conda determines how to install a compatible set of dependencies by analyzing the current environment, including any installed packagers and any version restrictions.

3.8.2 IDE- Jupyter Notebook

A computational notebook, or Jupyter Notebook, is a free, open-source, interactive web tool. It enables researchers to create documents that incorporate software code, computational results, explanatory language, and multimedia resources. The community

of users and developers has backed Jupyter in its efforts to modify the architecture so that the notebook may reflect several programming languages, including Julia (Ju), Python (Py), and R.

3.8.3 Tensorflow

A free and open-source software library called TensorFlow is used for artificial intelligence and machine learning tasks. Although it has a wide range of uses, deep neural network training and applications are of particular interest to it. It makes it easier for developers to build multi-layered, large-scale neural networks. Classification, detection, comprehension, discovery, prediction, and creativity are the key uses of TensorFlow. Many programming languages, including Python, Javascript, C++, and Java, support TensorFlow. TensorFlow can be used for time series analysis, picture recognition, text-based deep learning applications, voice or sound identification, and image recognition.

3.9 Summary

This section has been used to explain the methods, techniques, and approaches that were applied to implement the CNN architectures and Gabor initialized CNN architecture in detail. The following section will include the design methodology for the Sinhala character recognition process.

Chapter 4

Methodology

4.1 Introduction

This chapter presents the experimental methodology followed in our study. The chapter is organized as follows: Section 4.2 explains the construction of the data set. Section 4.3 presents the implementation of two different CNN models. Then, section 4.4 describes how the proposed Gabor initialized CNN architecture is implemented. Finally, chapter 4 will be concluded with a summary.

4.2 Data Set

Since the unavailability of a proper Sinhala handwritten character image dataset, it is required to construct a data set using appropriate mechanisms. The constructed image dataset contains all 60 Sinhala characters in the alphabet. According to the literature, the previous research regarding Sinhala character recognition using deep learning approaches was not carried out for all 60 Sinhala characters [3][4]. The constructed dataset contained a total of 6000 black and white images in .png format. These handwritten character images were gathered using the MS Paint application. Figure 4.1 illustrates a set of character images obtained from the constructed dataset.

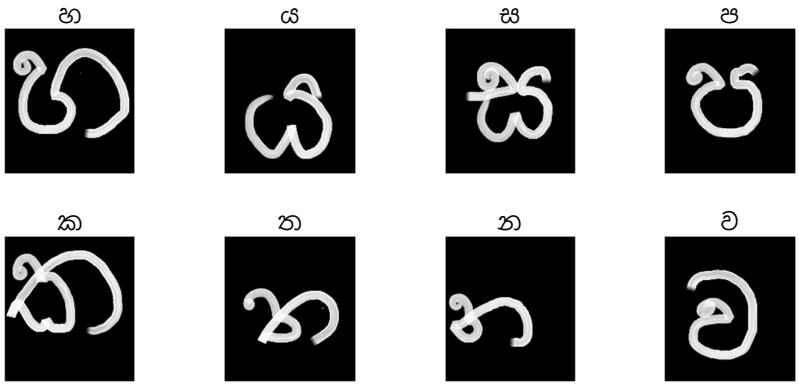


Figure 4.1: Sample images of characters

4.2.1 Data Collection

To collect images of Sinhala characters, twenty-five numbers of different writers were selected with age groups between 6 to 60 years, including both gender types, as illustrated in Table 4.1. One writer provided four samples of one Sinhala letter for all 60

characters creating a total of 6000 samples of images. Images are manipulated by using the MS paint tool. These handwritten Sinhala letters will be utilized as the training and testing dataset.

	Male	Female	Total
06-15 years	2	3	5
16-25 years	2	3	5
26-35 years	3	2	5
36-45 years	3	2	5
46-60 years	2	3	5
Total	12	13	25

Table 4.1: Distribution of selected writers

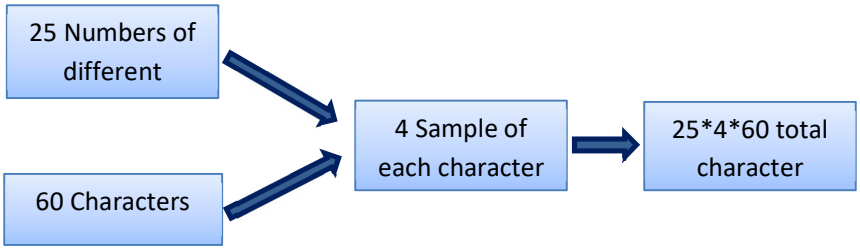


Figure 4.2: Character image dataset

4.3 CNN architectures

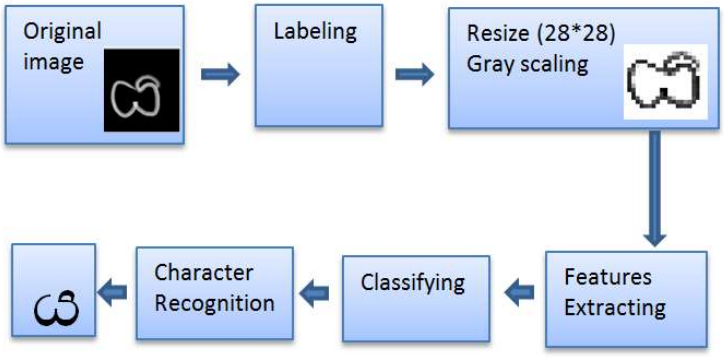
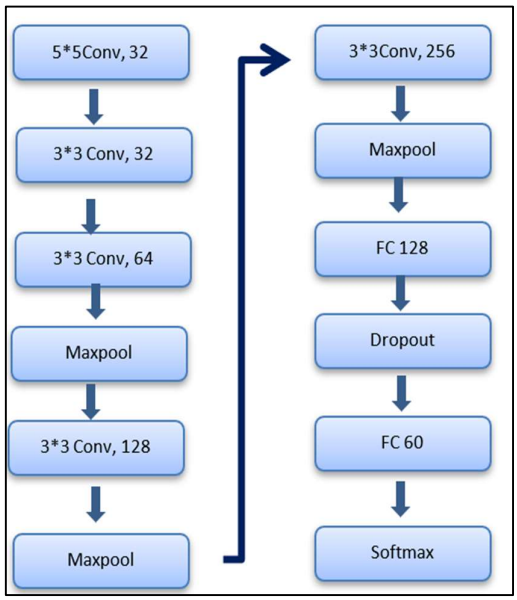


Figure 4.3: CNN implementation process

CNN is a deep learning approach that requires less effort in pre-processing than the primitive methods. The proposed CNN implementation process included basic steps such as labeling images, resizing to 28 × 28, converting to grayscale, feature extraction, classification and recognition, as shown in Figure 4.3. However, the performance of the CNN model is dependent on the parameter values of the architecture. Therefore, creating the model with optimum parameter values is required by fine-tuning and testing the

model various times. The ideal parameter set for the CNN models described in this study is determined using the experimentation approach. The proposed methodology consists of two different CNN models that use different parameter values and architecture.

4.3.1 First CNN model



Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 24, 24, 32)	832
conv2d_1 (Conv2D)	(None, 22, 22, 32)	9248
conv2d_2 (Conv2D)	(None, 20, 20, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 10, 10, 64)	0
conv2d_3 (Conv2D)	(None, 8, 8, 128)	73856
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 128)	0
conv2d_4 (Conv2D)	(None, 2, 2, 256)	295168
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 256)	0
flatten (Flatten)	(None, 256)	0
dense (Dense)	(None, 128)	32896
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 60)	7740

Total params: 438,236
Trainable params: 438,236
Non-trainable params: 0

None

Figure 4.4: First CNN architecture

The architecture of the first model is as follows: the first layer is a convolutional layer that contains 32 filters of size 5 × 5. A Rectified Linear Unit (ReLU) activates each of these filters. It is intended to replace the initialization of the first layer from the Xavier uniform distribution to Gabor filter initialization in the proposed Gabor initialized model, making it the most important layer in the proposed CNN architecture. The subsequent layer is a convolutional layer that performs ReLU activation and has 32 filters of size 3 × 3. The next layer is also a convolutional layer that has 64 filters of size 3 × 3 with ReLU activation. Following that, the data pass through a max-pooling layer with a size of 2 × 2 and a stride of 2. The data is then pooled and passed through a third convolutional layer of 128 filters of size 3 × 3. These data are once again activated by ReLU and treated into the max-pooling layer of size 2 × 2 with a stride of 2. Then it is gone through a fourth layer which has 256 filters of size 3 × 3 and are activated by ReLU. The data was then transferred into a next max pooling layer with a size of 2 × 2 and a stride of 2. The output is first pooled, and then it is fed into two fully connected layers, the last of which employs a softmax activation to categorize data. The dropout effect was applied during training before the softmax activation and after the first fully

connected layer as shown in Figure 3.5. Four dropout values D= (0, 0.25,0.5,0.8) have been applied to achieve the optimal dropout value for the models. The Xavier (Glorot) uniform distribution for the parameters is used as the initialization distribution for all convolutional layers in this model. All biases are initialized with a value of 0. The stochastic gradient descent (SGD) optimization approach was used throughout this CNN model. The selected hyperparameters for learning rate, learning rate decay and momentum are 0.01, 0.0005, and 0.9 respectively. The loss function is formed on the Categorical cross entropy. The visual representation of the first CNN model is illustrated in Figure 4.4.

4.3.2 Second CNN model

As discussed earlier, the effectiveness of CNN architecture focused on the values of parameters. Therefore, it is required to design the model with optimum parameter values by tuning and testing the model numerous times. The pooling layers of the architecture are often employed after the convolution layers. Nevertheless, the research conducted by W.V.S.K.Wasalthilake [3] has carried out a trial and error approach and invented that when inserting a pooling layer after the input layer, accuracy increased by 4%. Therefore, the second model is implemented using a max-pooling layer after the input layer.

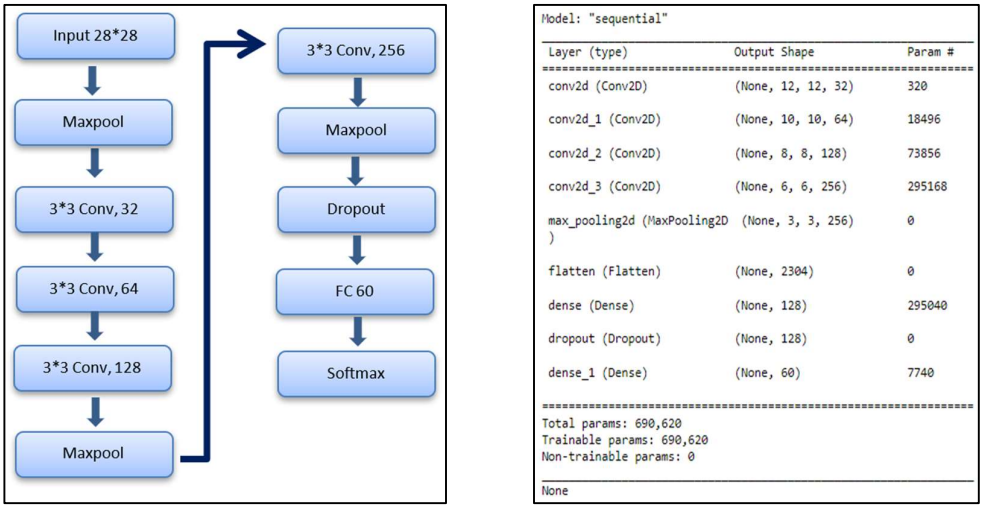


Figure 4.5: Second CNN architecture

The second CNN architecture is as follows: the first layer is a max pooling layer of size 2 × 2 with a stride of 2 and it is applied directly after the input layer. The following layer is a convolutional layer with 32, 3 × 3 filters. ReLU activates each of these filters. The

third layer is a convolutional layer that has 64 filters of size 3 × 3 and uses ReLU activation as well. A convolutional layer with 128 filters of size 3 × 3 is then applied to the data. Once more, ReLU activates these channels, which are then fed into the size 2 × 2 max-pooling layer with a 2 stride. Then it is gone through a convolution layer which has 256 filters of size 3 × 3 and is activated by ReLU. The output then passed into max pooling layer of size 2 × 2 with a stride of 2. Once completing the pooling, the output from here is then fed into two fully connected layers, where the final fully connected layer uses a softmax activation for classification. The dropout effect was applied during training before the softmax activation and after the first fully connected layer, presented in Figure 4.5. Four dropout values D= (0, 0.25,0.5,0.8) have been used to obtain the optimal dropout value for the models. In this model, the Xavier (Glorot) uniform distribution is used for the initialization of each convolutional layer. All biases are initialized with a value of 0. The stochastic gradient descent (SGD) optimization approach was used throughout this CNN model. For learning rate, learning rate decay, and momentum, the chosen hyperparameters are 0.01, 0.0005, and 0.9, respectively. The categorical cross entropy is used to define the loss function.

4.4 Gabor initialized CNN architecture

This section discusses the mechanism to develop the Gabor filter bank and the method to use it within the defined CNN structure. It is essential to discuss how the parameters of the Gabor filter are determined and how they are initialized within the CNN structure. The convolutional layers' default initializer when applying the Keras framework for implementation is a Xavier (Glorot) uniform distribution [29]. In this research, when using the Gabor initialization, it has to use the Gabor filter bank in the position of the Xavier distribution in the first convolution layer. This approach is also used in the previous research also [29]. There are two different methods to create a Gabor filter bank called grid-search and randomization which will be discussed in the next section. For all the experiments conducted in this paper, the grid-search method is utilized.

4.4.1 Grid search initialization method

The research conducted by Ozbulak described that the grid search initialization method was an excellent first move to use Gabor filters within CNN [27]. The central concept of the grid search is to obtain parameters by exhaustively selecting each option in a test [27]. Applying the grid search method requires utilizing a set interval within the

parameter's range. The Gabor filter contains five parameters that can be tuned. Tuning these parameters to obtain the most suitable initialization technique for the CNN structure may be laborious and computationally expensive. The proposed Gabor initialized CNN structure contains the variables that can be adjusted, such as kernel size (*ksize*), sigma (σ), lambda (λ), theta (θ), gamma(γ), and psi (ψ). Considering previous research conducted on this topic, finite ranges for each of these parameters are randomly selected for this study as shown in Table 4.2 [27]. As listed in Table 4.2, these domains were applied as the upper bound and lower bound of data points using a period. Data points are based on how many filters were intended to use in the first layer of the GCNN. In this research, it is designed to apply 32 filters, so for each parameter, it is required to divide each parameter range into 32 equal intervals and one Gabor filter is produced for each interval for each parameter, for a total of 32 Gabor filters. In this study, the Gabor filter bank was only applied to the initial convolutional layer because previous research has shown that there was a small payback of improvement by utilizing the Gabor filter in the following layers and sometimes, it decreases the performance of the CNN also [27].

Parameter	Range
ksize	5 × 5
sigma (σ)	[2,21]
lambda (λ)	[8,100]
theta (θ)	[0,360]
gamma (γ)	[0,300]
psi (ψ)	[0,360]

Table 4.2: Range for GCNN parameters

Table 4.2 illustrates the ranges and possible values applied for the Gabor filter utitized in GCNN. Here, *ksize* refers to the Gabor filter's size, and sigma stands for the Gaussian envelope's standard deviation. The sinusoidal factor's wavelength is determined by the lambda. The direction of the Gabor function's normal to its parallel strips is represented by the theta. The ellipticity of the Gabor function and the phase offset between the bands are defined, respectively, by the gamma and psi, which stand for the spatial aspect ratio [30].

4.4.2 Random initialization method

The random initialization approach distributes the model parameter values randomly across the parameter ranges without a predefined distribution, in contrast to the Gabor

grid-search method previously stated. Using the same parameter ranges listed in Table 4.2, it can compare directly between grid search and randomization to see which method is best for initializing the CNN. In this research, the randomization method is not considered for the initialization of CNN. Grid- search method versus the randomization method is visually represented in Figure 4.6.

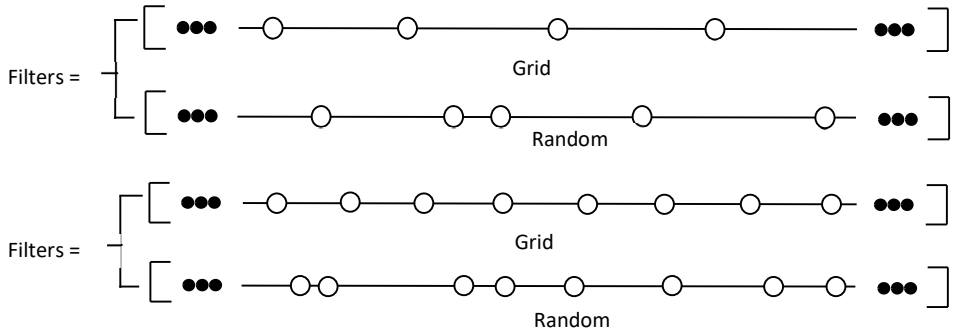


Figure 4.6: Grid- search method versus random initialization method

4.4.3 Gabor CNN (GCNN) architecture

This section will describe Gabor initialized CNN architecture used for the Sinhala character recognition. The Gabor initialized CNN architecture is similar to the CNN architecture discussed in section 4.3.1, but instead of a Xavier normal distribution used in the initial layer is replaced by a Gabor filter bank. In the proposed GCNN architecture, the first layer is a convolutional layer that contains 32 filters of size 5 × 5 initialized with the Gabor filter. A ReLU function then activates each of these filters. The second layer is a convolutional layer that uses ReLU activation and 64 filters of size 3 × 3. The following layer is a max-pooling layer with a dimension of 3 × 3 and a stride of 2. After the data had been pooled, a third convolutional layer with 128 filters of size 3 × 3 was applied. Once again, ReLU activates these output channels, which are then passed into the fourth convolutional layer, which has 256 filters of size 3 × 3 and is also activated by ReLU. Following that, output data went through a max pooling layer with a size of 3 × 3 and a stride of 2. After pooling is finished, the output is fed into two fully connected layers. For categorization, the final fully connected layer employs softmax activation. Dropout function is utilized during training before the softmax activation with four different values D= (0.0,0.25,0.50,0.80). The most important aspect is that, aside from the first layer, all convolutional layers are initialized with the Xavier uniform

distribution. Biases are all set to zero by default. The RMSprop optimizer was used as the optimization algorithm. For learning rate, momentum, and learning rate decay, the preferred hyperparameters are 0.01, 0.9, and 0.0005, respectively. The Categorical cross entropy is used to calculate the loss function. Figure 4.7 demonstrates a visual presentation of the proposed GCNN model.

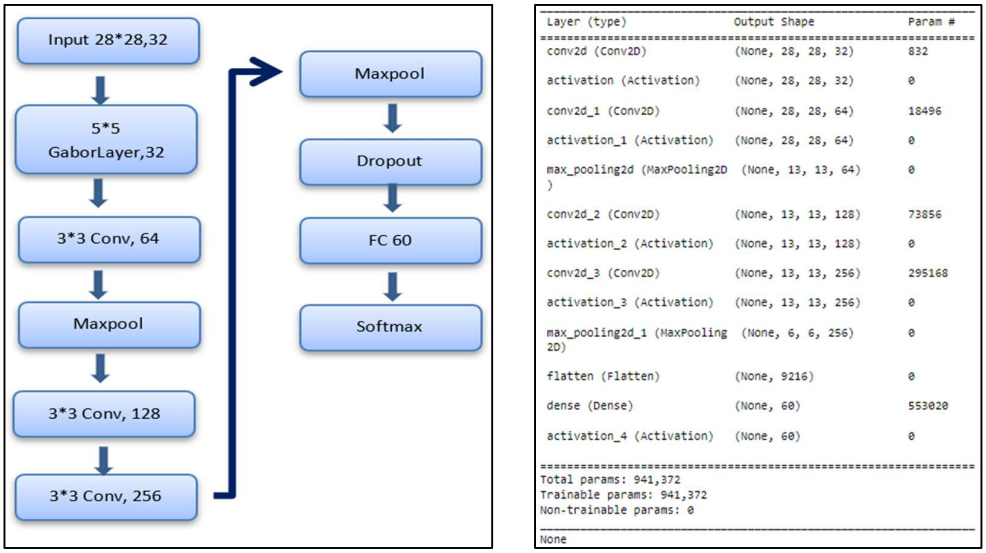


Figure 4.7: GCNN architecture

4.5 Summary

This section has discussed the two CNN models and the GCNN model architectures and the key concepts behind them. In addition, individual processes and the nature of operation of the separate components of each architecture were discussed in detail. In the next section, testing and results will provide testing strategies and experiments carried out for each architecture and corresponding results obtained.

Chapter 5

Testing and Results

5.1 Introduction

This chapter will discuss the testing process and the results obtained from all three models proposed in the study. The chapter contains a detailed description of how each experiment will be conducted and what results they have provided. For each test, the number of trials and epochs applied is discussed under each experiment. In addition to that, each experiment includes testing and validation accuracy plots. All results are rounded to four decimal places to provide a high precision level because, in some situations, some of the results obtained seem to be similar closely. Section 5.2 contains the results of two CNN models and comparisons of their accuracy levels concerning training and testing. Section 5.3 includes experiment procedures regarding the proposed GCNN model. The experiment conducted under section 5.3.1 will investigate the behavior of each parameter used in the Gabor filter. This experiment is carried out to determine whether Gabor filter parameters have a significant impact on the proposed GCNN based on the Sinhala character dataset. In addition to that, the effect of dropout is also investigated in both cases CNN and GCNN. Chapter 5 concludes with a summary.

5.2 Testing on CNN models

After preprocessing the Sinhala character image dataset, the final dataset consisted of 6000-character images. For training and testing the proposed CNN models, the dataset was divided into two subsets. The training dataset contained 3600-character images. The testing dataset contained 2400 images. Both CNN architectures were tested for 30 epochs and are trained using a learning rate of 0.001 and used with Stochastic Gradient Descent optimizer. Furthermore, each deep learning model is trained using a default learning rate scheduler. All CNN architectures are experimented with four different dropout values to determine how the dropout effect improves the performance of the CNN models.

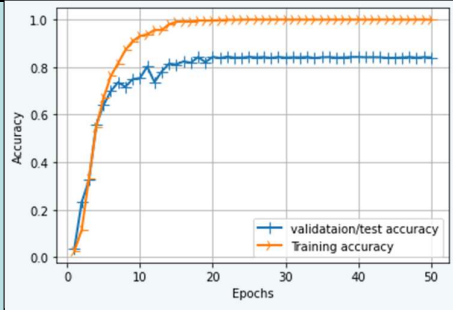
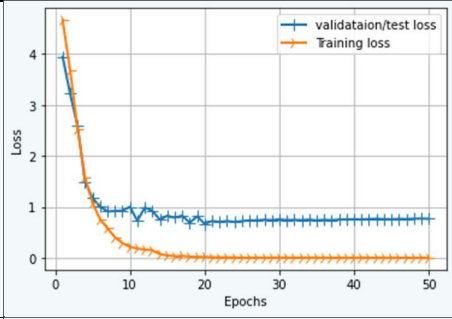
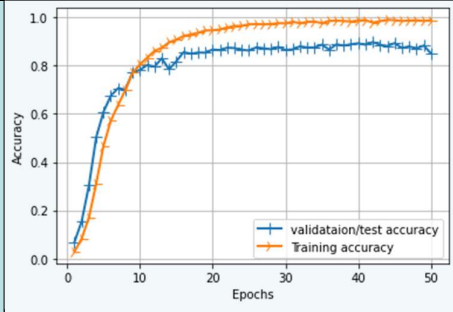
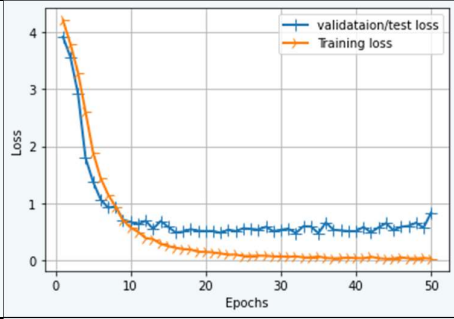
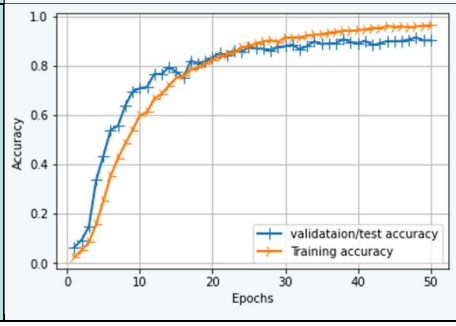
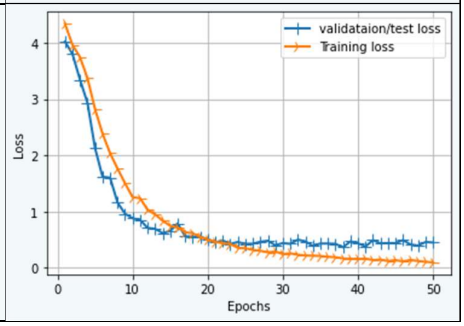
5.2.1 Results of the first CNN model

The results of the first CNN model are recorded in Table 5.1. In here 50 epochs has been used for four different dropout values D= [0.0 ,0.25, 0.5,0.8].

Dropout	Training Accuracy	Training Lost	Testing Accuracy	Testing Loss
0.0	0.9994	0.0025	0.8407	0.7662
0.25	0.9888	0.0327	0.8502	0.8360
0.5	0.9633	0.0976	0.9014	0.4514
0.8	0.6203	1.1180	0.8485	0.4735

Table 5.1: Results of the first CNN model

For this experiment, the accuracy values are changed from time to time when tuning parameters. The highest training accuracy was achieved without the dropout effect and the highest validation accuracy was achieved with 0.5 dropout effect. The highest training accuracy achieved is 0.9994 and the testing accuracy is 0.9014. The testing and validation curves were illustrated in Figure 5.1 for each dropout value.

Dropout value	Training accuracy vs Validation accuracy	Training loss vs Validation loss
0.0		
0.25		
0.5		

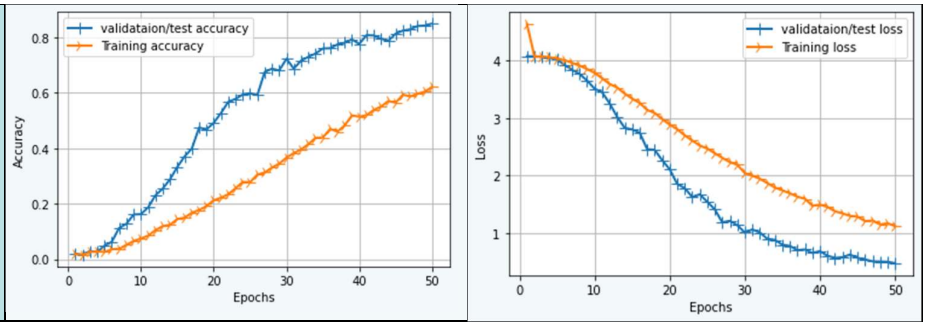
0.8	

Figure 5.1: Testing and validation curves of the first CNN model

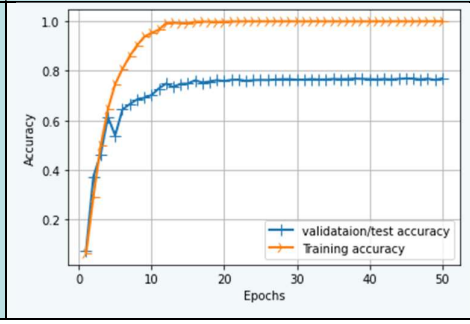
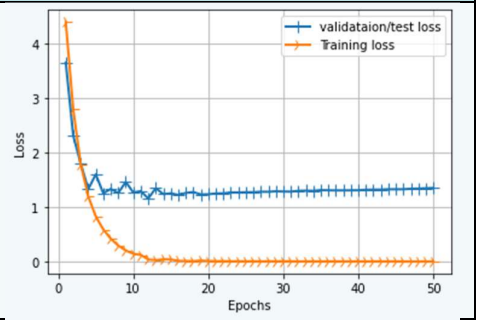
5.2.2 Results of the second CNN model

The outcomes of the second CNN model are reported in Table 5.2. In here 50 epochs has been used for four different dropout values D= [0.0 ,0.25, 0.5,0.8].

Dropout	Training Accuracy	Training Lost	Testing Accuracy	Testing Loss
0.0	0.9997	0.0013	0.7693	1.3529
0.25	0.9891	0.0334	0.7706	1.2502
0.5	0.9417	0.1543	0.7942	1.0058
0.8	0.3430	2.1192	0.5807	1.4657

Table 5.2: Results of the second CNN model

For this experiment also, the accuracy levels are changed from time to time when tuning the parameters. The highest training accuracy was achieved without the dropout effect and the highest validation accuracy was achieved with 0.5 dropout effect. The highest training accuracy achieved is 0.9997 and the testing accuracy is 0.7942. The testing and validation curves were illustrated in Figure 5.2 for each dropout value.

Dropout value	Training accuracy vs Validation accuracy	Training loss vs Validation loss
0.0		

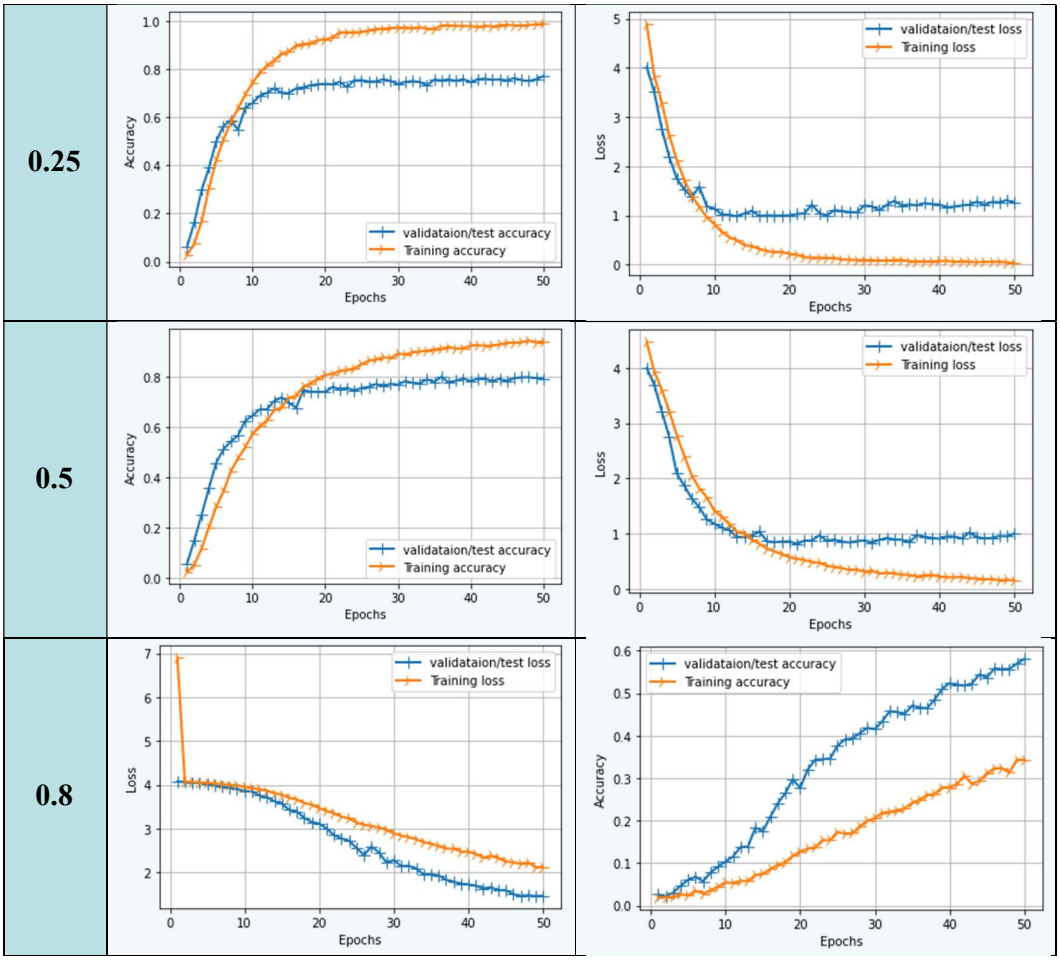


Figure 5.2: Testing and validation curves of the second CNN model

5.2.3 Discussion

Considering Figure 5.3, training accuracy drops when both CNN architectures introduce dropout and rapidly decreases when dropout is greater than 0.5. But considering the validation accuracy, it is slightly increased up to 0.5 dropout and then reduced. Validation and training error of both CNN architectures increases with dropout and it rapidly increases for higher dropout values. In addition, the first model has better training and validation accuracy than the second model. So, introducing a max polling layer after the data input layer and before the convolution layer does not improve the training and validation accuracy for the constructed Sinhala character dataset.

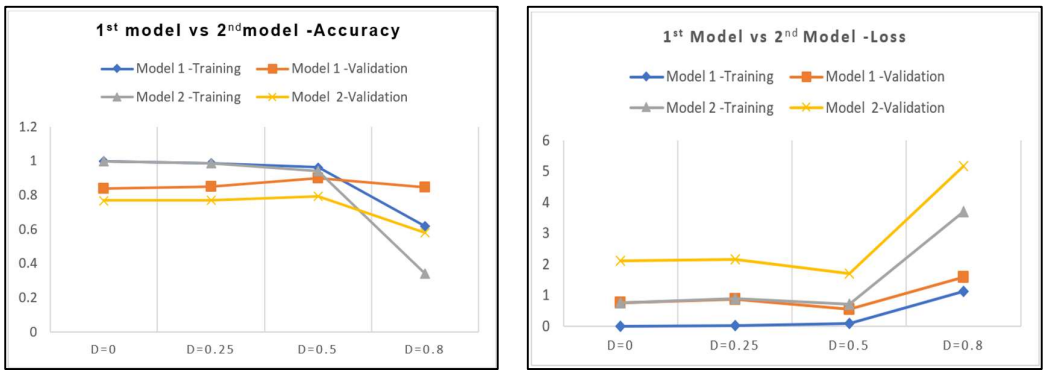


Figure 5.3: Comparison of CNN architectures

5.3 Testing on GCNN model

The experiment described under Section 5.3.1 was carried out to investigate the behavior of each parameter used in the Gabor filter. This experiment helps to determine significant level of the parameters over the GCNN architecture concerning the Sinhala character dataset. In addition, the effect of dropout is also investigated under four test cases. In the test conducted using GCNN architecture, the Gabor filter collection used in the initial layer is initialized using the grid search algorithm.

5.3.1 Importance of Gabor Parameters Training on Dataset

In this experiment, the Gabor filter collection was applied to initialize the initiatory layer of the proposed architecture to determine the influence of each parameter on the overall GCNN model. The generated Gabor filter collection will be fixed across all tests in which each variable is evaluated and adjusted once per test. For example, when considering one test, the list of parameters will appear like [k= 5×5, σ= σ', λ=50, θ=0, γ=150, ψ=0], where σ' is a static constant figure given from the closed interval of I = [2, 22], which labels the lower bound value at 2 and upper bound value at 22 for σ parameter. For all test cases, the GCNN architecture presented in Section 4.3.3 will be applied. Each test trains for 30 epochs and the filter size will remain in 5×5 for all test cases in this experiment at 32 filters in the initial layer.

As described above, for each test and the training phase, the Gabor filter bank will be maintained in a static manner, and created Gabor filters enable learning through training. The parameter values selected for the test cases are based on previous research [29]. Table 5.3 provides a list of the Gabor filter's tunable parameters. The mentioned ranges for parameters are inclusive. Furthermore, the experiment is also designed to determine

the impact of the dropout effect on the quality of results. In here two different dropout values of 0.0, 0.5 (D = [0, 0.50]) are applied. Each test will contain 30 epochs. The total number of Gabor filters generated will be constant at 32 filters. Each test is run once for each parameter change across all parameters. Consider one test run, $\sigma = 14$, while other variables [*ksize,* σ*,* λ*,* θ*,* γ*,* ψ] in the Gabor filter are kept constant and in the next test case, σ will be a gradual increment to $\sigma = 18$ and the other variables [*ksize,* σ*,* λ*,* θ*,* γ*,* ψ] are fixed. Each Gabor parameter has six test cases, each with a distinct value within the range, as indicated in Table 5.4, and the intervals between each test are fixed. Therefore, for one given Gabor parameter, a total of twelve have to be conducted considering the two dropout values. Theta and Psi values of 0◦ and 360◦ provide the same result, so one value will be considered.

Parameter	Range	Selected Values
Sigma, (σ)	[2, 22]	[2, 6, 10, 14, 18, 22]
Lambda, (λ)	[0, 100]	[0, 20, 40, 60, 80, 100]
Theta, (θ)	[0, 360]	[0, 60, 120, 180, 240, 300]
Gamma, (γ)	[0, 300]	[0, 60, 120, 180, 240, 300]
Psi, (ψ)	[0, 360]	[0, 60, 120, 180, 240, 300]

Table 5.3: Gabor parameter ranges and selected values

Experiment	Dropout	Ksize *(ksize)*	Sigma *(σ)*	Lambda *(λ)*	Theta *(θ)*	Gamma *(γ)*	Psi *(ψ)*
I: Sigma	[0.0,0.50]	5×5	[2, 22]	10	0	150	0
II: Lambda	[0.0,0.50]	5×5	4	[0,100]	0	150	0
III: Theta	[0.0,0.50]	5×5	4	10	[0,300]	150	0
IV: Gamma	[0.0,0.50]	5×5	4	10	0	[0,300]	0
V: Psi	[0.0,0.50]	5×5	4	10	0	150	[0,300]

Table 5.4: Gabor parameter test cases

5.3.2 Results

As discussed in Section 5.3.1, a total of twelve tests per parameter are carried out in the pre-defined test environment with 30 epochs. All the results are listed in Table 5.5 for parameter Sigma, Table 5.6 for parameter Lambda, Table 5.7 for parameter Theta, Table 5.8 for parameter Gamma and Table 5.9 for parameter Psi. The height training and validation accuracy achieved for each dropout value is bolded.

I. Sigma, σ

Parameters	Dropout	σ	Training		Validation	
			Accuracy	Loss	Accuracy	Loss
ksize=5 × 5 ***σ -vary*** ***λ =50*** ***θ =0*** ***γ =150*** ***ψ =0***	0	2	0.9707	0.1372	0.7585	2.0700
	0	6	0.9630	0.2009	0.7348	2.7967
	0	10	0.9716	0.1657	0.7955	2.2569
	0	**14**	**0.9790**	**0.1340**	**0.8024**	**2.1065**
	0	18	0.9532	0.2686	0.7206	3.4617
	0	22	0.9633	0.2002	0.7477	2.7944
	0.5	2	0.9139	0.2724	0.7688	1.0060
	0.5	**6**	**0.9208**	**0.2567**	**0.8768**	**0.7960**
	0.5	10	0.9082	0.2730	0.7921	0.8882
	0.5	14	0.8579	0.4204	0.7008	1.2771
	0.5	18	0.9030	0.3116	0.7895	0.8313
	0.5	22	0.8852	0.3591	0.7619	0.9887

Table 5.5: Results of the GCNN with static first Gabor layer, changing σ

In this test, a static first layer was implemented using the Gabor filter bank by only changing the σ in the range with the values of [2, 6, 10, 14, 18, 22] inclusive. The layers can learn during the training. Gabor variable σ changes six times for each dropout value [0.0, 0.5]. Training and validation accuracy achieved by changing σ with a static Gabor filter in GCNN for dropout values 0.0 and 0.5 are demonstrated in Figure 5.4 and Figure 5.5 respectively. Each colored bar in the legend represents the training dropout rate.

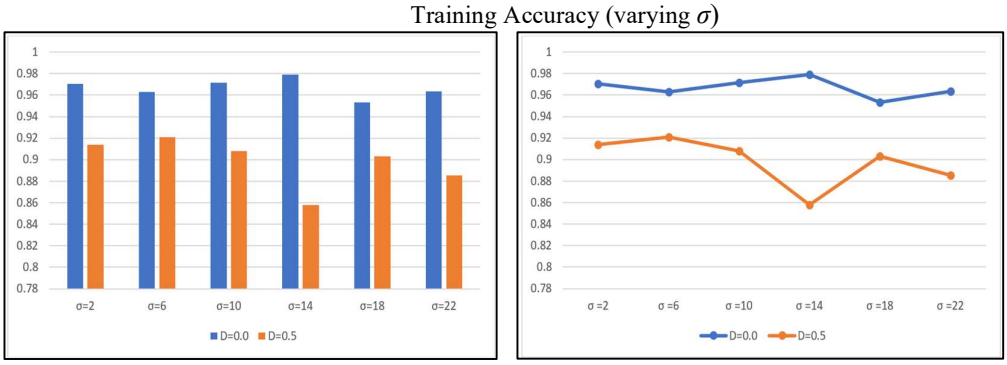


Figure 5.4: Training accuracy of the GCNN with static first Gabor layer, changing σ

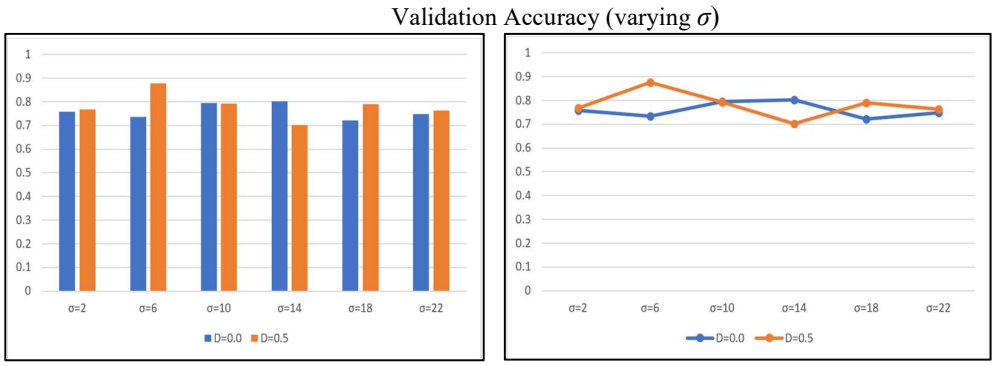


Figure 5.5: Validation accuracy of the GCNN with static first Gabor layer, changing σ

II. Lambda, λ

Parameters	Dropout	λ	Training		Validation	
			Accuracy	Loss	Accuracy	Loss
ksize=5 × 5 ***σ =10*** ***λ -vary*** ***θ =0*** ***γ =150*** ***ψ =0***	0	0	0.0181	-	0.0159	-
	0	**20**	**0.9885**	**0.0657**	**0.8015**	**2.3630**
	0	40	0.9690	0.2022	0.789	2.3974
	0	60	0.9693	0.1494	0.7740	2.9163
	0	80	0.9630	0.2114	0.7456	2.8259
	0	10	0.9429	0.3097	0.7288	2.9747
	0.5	0	0.0164	-	0.0185	-
	0.5	20	0.9208	0.2477	0.7856	1.0048
	0.5	**40**	**0.9437**	**0.1805**	**0.7908**	**0.9356**
	0.5	60	0.9021	0.3137	0.7572	0.9714
	0.5	80	0.8878	0.3666	0.7632	0.9834
	0.5	10	0.9130	0.2707	0.7740	0.9853

Table 5.6: Results of the GCNN with static first Gabor layer, changing λ

In this test, using the Gabor filter bank and simply modifying the λ in the inclusive range [0, 20, 40, 60, 80, 100], a static initial layer was created in this test. Gabor variable λ changes six times for each dropout values [0.0, 0.5]. Training and validation accuracy achieved by changing λ with a static Gabor filter in GCNN for dropout values 0.0 and 0.5 are pictured in Figure 5.6 and Figure 5.7 respectively. The legend's colored bars display the dropout rate that was applied during training.

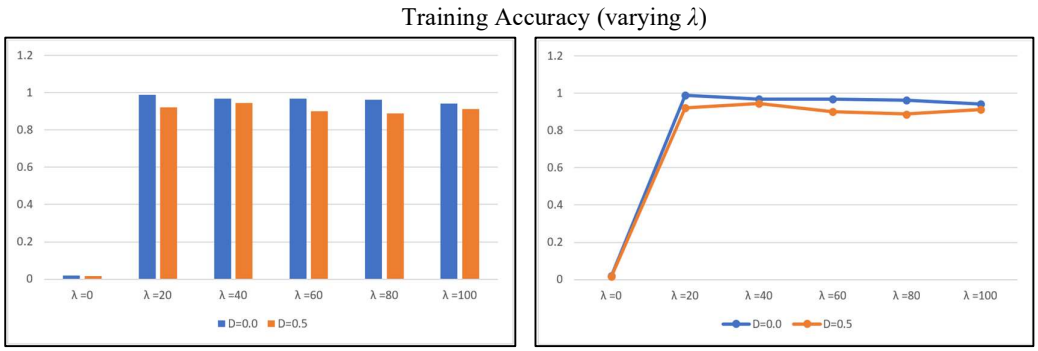


Figure 5.6: Training accuracy of the GCNN with static first Gabor layer, changing λ

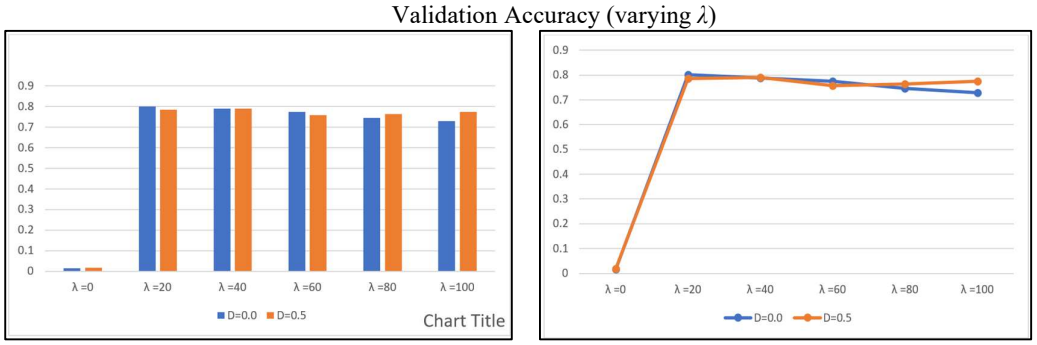


Figure 5.7: Validation accuracy of the GCNN with static first Gabor layer, changing λ

III. Theta, θ

Parameters	Dropout	θ	Training		Validation	
			Accuracy	Loss	Accuracy	Loss
ksize=5 × 5 ***σ =10*** ***λ =50*** ***θ -vary*** ***γ =150*** ***ψ =0***	**0**	**0**	**0.9793**	**0.0753**	**0.8037**	**1.6760**
	0	60	0.9294	0.5964	0.7185	5.2836
	0	120	0.9337	0.5007	0.7473	3.2503
	0	180	0.8671	1.2762	0.7047	4.3652
	0	240	0.9317	0.4780	0.763	3.1574
	0	300	0.9216	0.6478	0.6478	4.0422
	0.5	0	0.9389	0.1903	0.8295	0.7818
	0.5	**60**	**0.9452**	**0.1763**	**0.8364**	**0.7158**
	0.5	120	0.8786	0.3924	0.7469	1.1535
	0.5	180	0.8737	0.4069	0.7720	0.9261
	0.5	240	0.8462	0.4935	0.7667	0.9784
	0.5	300	0.8203	0.5743	0.7508	0.9517

Table 5.7: Results of the GCNN with static first Gabor layer, changing θ

A static foremost layer was implemented using the Gabor filter collection by only changing the θ in the inclusive range with the values of [0, 60, 120, 180, 240, 300]. For each dropout value [0.0, 0.5], the Gabor variable changed θ six times. Theta calculating at 0 and 360 provides the same result, so 360 test case is neglected. Training and validation accuracy achieved by changing θ with a static Gabor filter in GCNN for dropout values 0.0 and 0.5 are pictured in Figure 5.8 and Figure 5.9 consecutively.

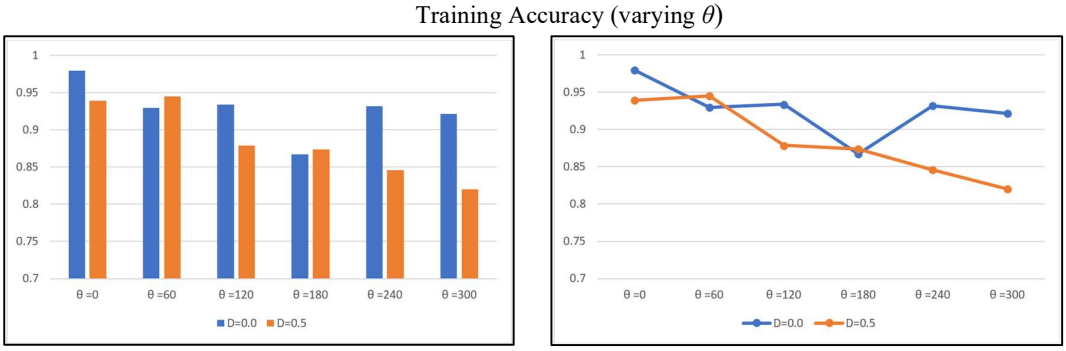


Figure 5.8: Training accuracy of the GCNN with static first Gabor layer, changing θ

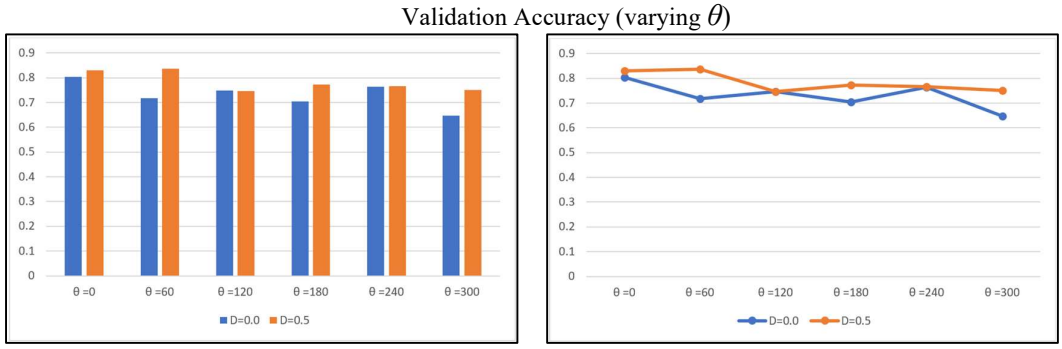


Figure 5.9: Validation accuracy of the GCNN with static first Gabor layer, changing θ

IV. Gamma, γ

Parameters	Dropout	γ	Training		Validation	
			Accuracy	Loss	Accuracy	Loss
	0	**0**	**0.9389**	**0.6047**	**0.7787**	**4.9222**
	0	60	0.9334	0.7226	0.7753	6.5303
	0	120	0.8984	1.6060	0.7404	6.7586
	0	180	0.9483	0.6143	0.7835	4.8240
ksize=5 × 5	0	240	0.9110	0.7176	0.7624	3.7723
σ =10	0	300	0.9162	0.8638	0.7589	4.2815
λ =50	0.5	0	0.8321	0.5610	0.7482	1.0913
θ =0	0.5	60	0.8335	0.5436	0.7439	1.1527
γ -vary						

$\psi = 0$	0.5	120	0.8241	0.5706	0.7559	1.0017
	0.5	**180**	**0.8591**	**0.4497**	**0.7925**	**0.8572**
	0.5	240	0.8525	0.4859	0.7690	0.9573
	0.5	300	0.8111	0.6179	0.7129	1.0890

Table 5.8: Results of the GCNN with static first Gabor layer, changing γ

Here, the Gabor filter bank was used to build a static first layer, and only the γ in the inclusive range [0, 60, 120, 180, 240, 300] was changed. Gabor variable γ changes six times for each dropout values [0.0, 0.5]. Training and validation accuracy achieved by changing γ with a static Gabor filter in GCNN for dropout values 0.0 and 0.5 are pictured in Figure 5.10 and Figure 5.11 consecutively.

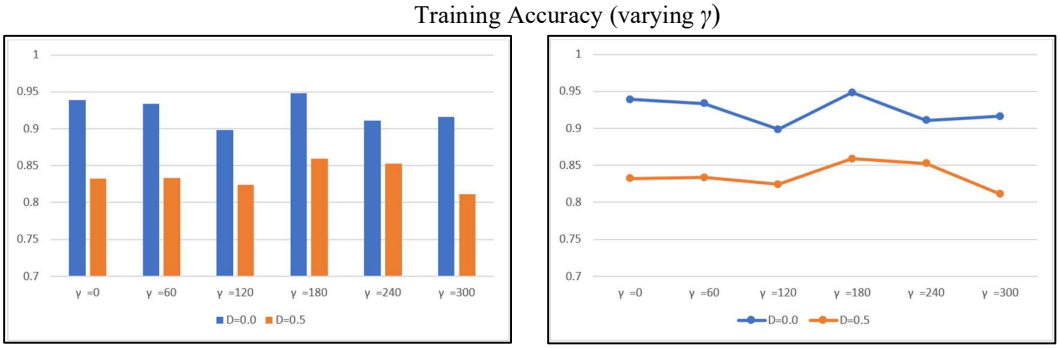


Figure 5.10: Training accuracy of the GCNN with static first Gabor layer, changing γ

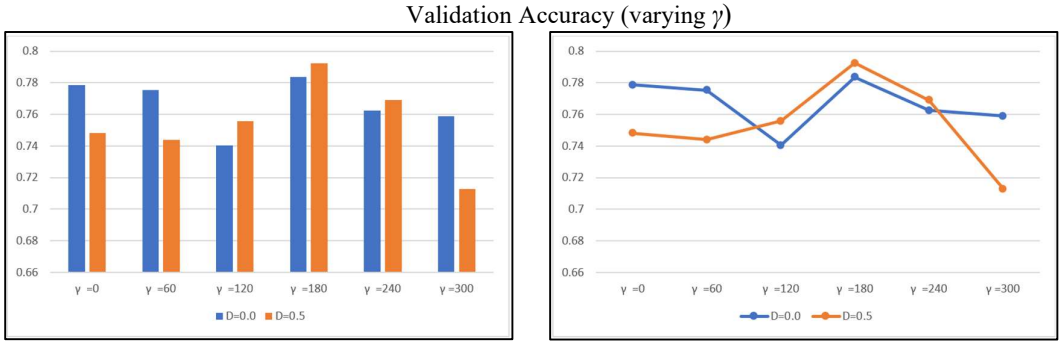


Figure 5.11: Validation accuracy of the GCNN with static first Gabor layer, changing γ

V. Psi, ψ

Parameters	Dropout	ψ	Training		Validation	
			Accuracy	Loss	Accuracy	Loss
	0	0	0.9199	0.8462	0.7693	4.1217
	0	60	0.9199	0.6446	0.7434	3.2302
	0	120	0.0215	4.0747	0.0133	4.0766
	0	180	0.0204	4.0717	0.0125	4.0754

ksize=5 × 5 ***σ =10*** ***λ =50*** ***θ =0*** ***γ =150*** ***ψ -vary***	0	240	0.0210	4.0720	0.0116	4.0750
	0	**300**	**0.9466**	**0.3743**	**0.7886**	**2.9369**
	0.5	0	0.8548	0.4585	0.7744	0.8855
	0.5	60	0.8855	0.3580	0.7985	0.8693
	0.5	120	0.0207	4.0715	0.0121	4.0758
	0.5	180	0.0172	4.0718	0.0138	4.0752
	0.5	240	0.0195	4.0720	0.016	4.0749
	0.5	**300**	**0.8858**	**0.3621**	**0.6919**	**0.8192**

Table 5.9: Results of the GCNN with static first Gabor layer, changing ψ

In this test, a static foremost layer was implemented using the collection of Gabor filter by only changing the ψ in the range with the values of [0, 60, 120, 180, 240, 300] inclusive. Gabor variable ψ changes six times for each dropout value [0.0, 0.5]. Phi calculating at 0 and 360 provide the same result, so 360 test case is neglected. Training and validation accuracy changing ψ with a static Gabor filter in GCNN for dropout values 0.0 and 0.5 are pictured in Figure 5.12 and Figure 5.13 consecutively.

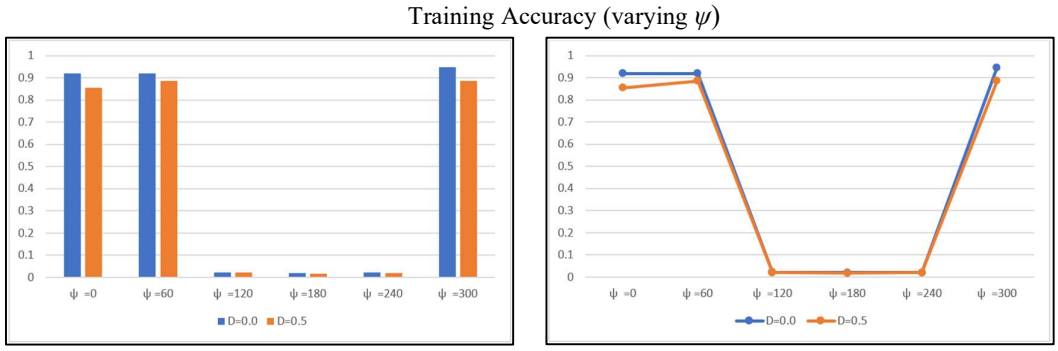


Figure 5.12: Training accuracy GCNN with static first Gabor layer, changing ψ

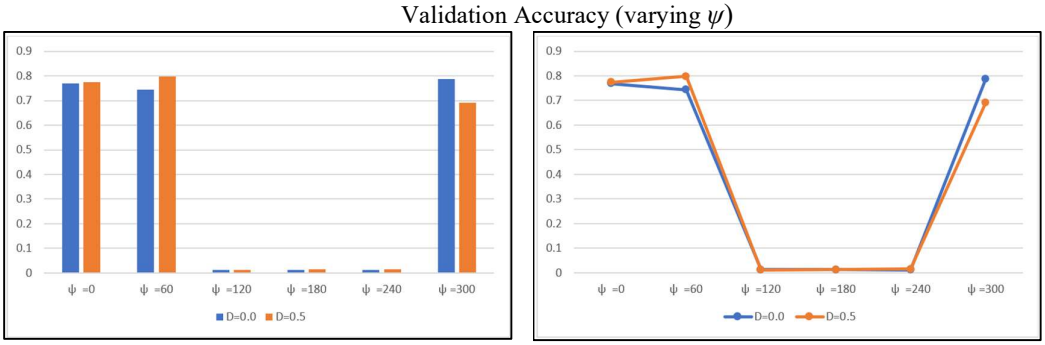


Figure 5.13: Validation accuracy GCNN with static first Gabor layer, changing ψ

5.3.3 Discussion

I. Sigma, σ

Training accuracy decreased when dropout was introduced to the Gabor initialized CNN. It can be observed that the error increases with dropout. There appears no significant pattern in terms of σ. The highest validation and training accuracy is achieved when σ = 14 and σ = 6 for dropout 0.0 and 0.5, respectively.

Table 5.10 displays the comparative training and validation accuracy achieved for the first CNN model architecture proposed in Section 4.3.1 and the GCNN architecture with a static filter bank with σ=14 for dropout 0.0 and σ=6 for dropout 0.5. Considering Figure 5.14 and Figure 5.15, it can conclude that for both dropout values 0.0 and 0.5, the validation and training accuracy of the GCNN are higher than CNN accuracy for each epoch. So, the convergence of the GCNN is faster than the corresponding CNN model. In addition, according to Figure 5.14 and Figure 5.15, σ has no significant impact on the final results.

	Training Accuracy		Validation Accuracy		Training Accuracy		Validation Accuracy	
	CNN	GCNN (σ=14)	CNN	GCNN (σ=14)	CNN	GCNN (σ=6)	CNN	GCNN (σ=6)
Epoch	**D=0**	**D=0**	**D=0**	**D=0**	**D=0.5**	**D=0.5**	**D=0.5**	**D=0.5**
1	0.0273	0.1518	0.0383	0.2652	0.0235	0.0741	0.0628	0.2260
2	0.1160	0.4380	0.2320	0.4796	0.0505	0.1751	0.0917	0.3771
3	0.3332	0.6352	0.3246	0.5441	0.0835	0.3255	0.1464	0.5006
4	0.5471	0.7735	0.5579	0.6500	0.1553	0.4753	0.3349	0.5794
5	0.6696	0.8576	0.6401	0.6449	0.2500	0.5844	0.4339	0.6005
6	0.7623	0.8967	0.6991	0.7081	0.3533	0.6490	0.5381	0.6651
7	0.8163	0.9159	0.7361	0.7452	0.4237	0.6983	0.5549	0.6802
8	0.8717	0.9363	0.7172	0.7559	0.4819	0.7491	0.6367	0.7103
9	0.9093	0.9480	0.7495	0.7490	0.5373	0.7856	0.6931	0.7318
10	0.9305	0.9552	0.7538	0.7718	0.5962	0.8042	0.7077	0.7262

Table 5.10: Comparison of GCNN with constant σ and first CNN model results

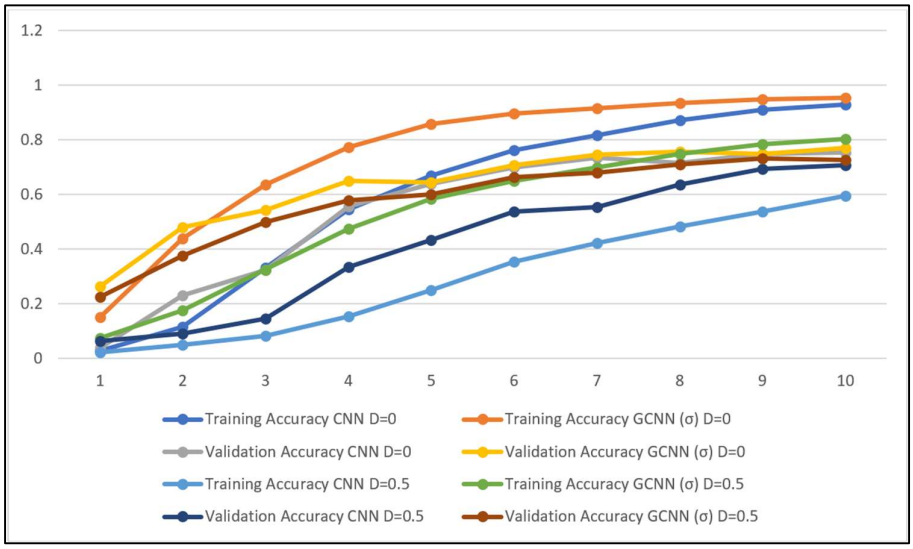


Figure 5.14: Accuracy of the GCNN with constant σ and first CNN model

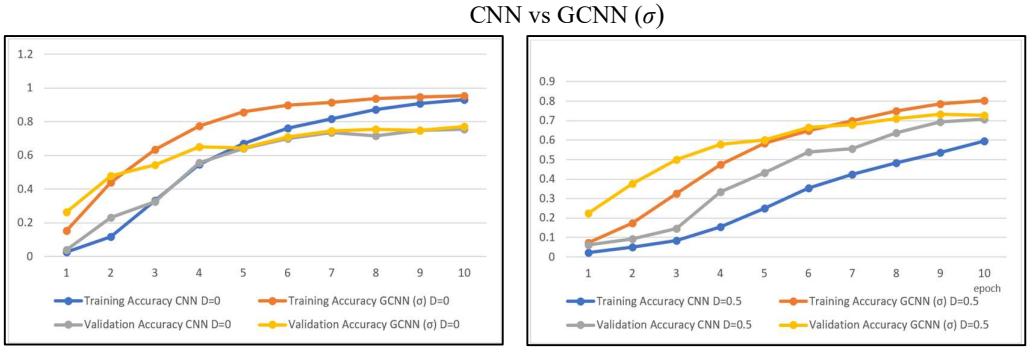


Figure 5.15: Accuracy of the GCNN with constant σ and first CNN model for dropout 0.0 and 0.5

II. **Lambda, λ**

The highest scoring accuracy values for dropouts 0.0 and 0.5 are bolded in Table 5.6. According to Figure 5.6 and Figure 5.7, It can be observed that the accuracy levels are very close to each other and irregular like the σ. For both dropout values of 0.0 and 0.5, λ appears to be the good in the range [20, 100]. When dropout is introduced to the GCNN architecture, there is no significant change in performance. The important conclusion of this test is that $\lambda = 0$ should never be used for the Gabor filter parameter setting because the network has trouble learning when there is no value given to lambda. So λ must be greater than 0 to achieve results from the GCNN architecture.

The highest training and validation accuracy levels are obtained when λ=20 and λ=40 for dropout value 0.0 and 0.5 respectively. Considering Figure 5.17, it can conclude that for dropout value 0.0, the training and validation accuracy of the GCNN with λ=20 become almost similar to CNN accuracy for higher epochs. Considering Figure 5.16, for

dropout value 0.5, the validation accuracy of both CNN and GCNN with λ=40 becomes very close when the number of epochs increases. In contrast, the training accuracy of GCNN is higher than the CNN for a higher number of epochs. Finally, it is shown that λ has no significant impact on the results.

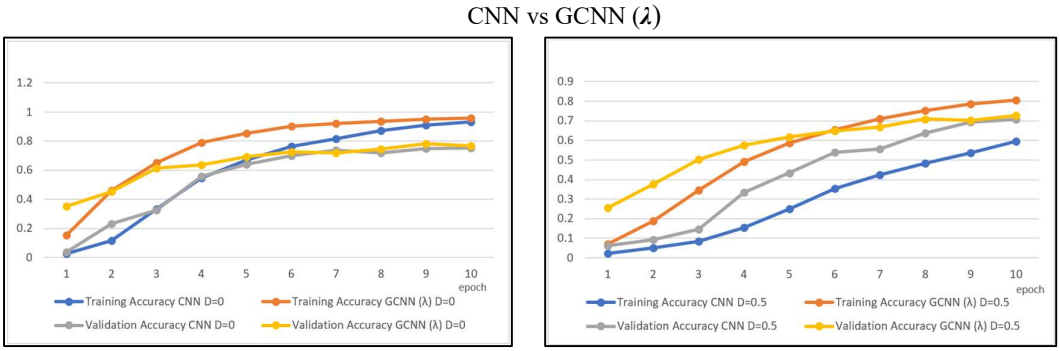


Figure 5.16: Accuracy of the GCNN with constant λ and first CNN model for dropout 0.0 and 0.5

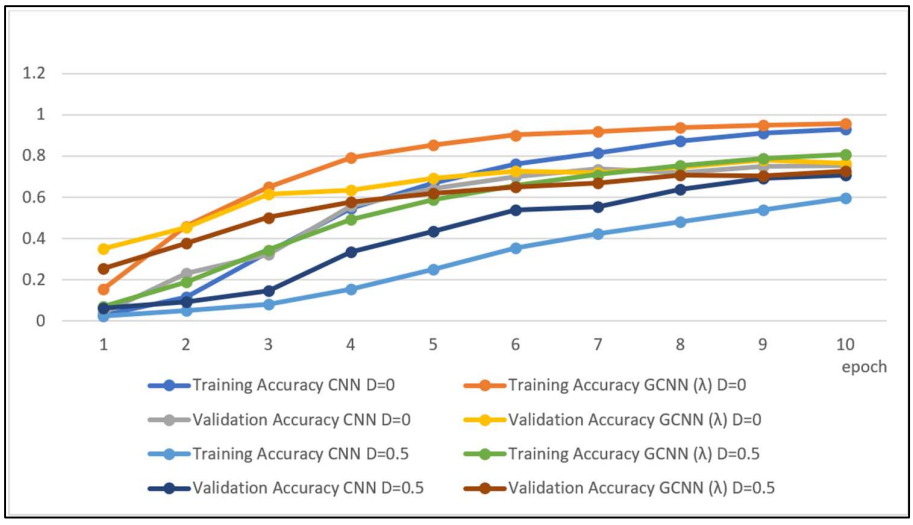


Figure 5.17: Accuracy of the GCNN with constant λ and first CNN model

	Training Accuracy		Validation Accuracy		Training Accuracy		Validation Accuracy	
	CNN	GCNN (λ=20)	CNN	GCNN (λ=20)	CNN	GCNN (λ=40)	CNN	GCNN (λ=40)
Epoch	D=0	D=0	D=0	D=0	D=0.5	D=0.5	D=0.5	D=0.5
1	0.0273	0.1538	0.0383	0.3517	0.0235	0.0692	0.0628	0.2548
2	0.1160	0.4598	0.2320	0.4537	0.0505	0.1886	0.0917	0.3775
3	0.3332	0.6507	0.3246	0.6156	0.0835	0.3447	0.1464	0.5019
4	0.5471	0.7905	0.5579	0.6350	0.1553	0.4914	0.3349	0.5751
5	0.6696	0.8530	0.6401	0.6926	0.2500	0.5875	0.4339	0.6177
6	0.7623	0.9010	0.6991	0.7271	0.3533	0.6553	0.5381	0.6483

7	0.8163	0.9199	0.7361	0.7198	0.4237	0.7092	0.5549	0.6677
8	0.8717	0.9363	0.7172	0.7452	0.4819	0.7537	0.6367	0.709
9	0.9093	0.9492	0.7495	0.7817	0.5373	0.7873	0.6931	0.7025
10	0.9305	0.9581	0.7538	0.7663	0.5962	0.8060	0.7077	0.7279

Table 5.11: Comparison of GCNN with constant λ and first CNN model results

III. **Theta, θ**

The highest accuracy levels for dropouts 0.0 and 0.5 are bolded in Table 5.7. When θ =0 and θ =60 (π/3) highest accuracy level can be obtained for the dropout 0.0 and 0.5, respectively. It is shown that there is no significant impact on the training and validation accuracy when introducing the dropout to the GCNN. According to Figure 5.8, training accuracy may decrease with the higher value of θ for both dropout values. In addition, there is no significant pattern to recognize from the accuracy level and they are very close to one another.

Considering Figure 5.18 and Figure 5.19, it can conclude that, for dropout 0.0 the training and validation accuracy of the CNN and GCNN with θ =0 becomes almost similar for higher epochs. Considering Figure 5.18, for dropout value 0.5, the validation and training accuracy of GCNN with θ =60 is higher than the corresponding CNN accuracy. As discussed, different features at each frequency can be derived by rotating the Gabor filters; hence Gabor filter is considered an alternative to transfer learning [27].

	Training Accuracy		**Validation Accuracy**		**Training Accuracy**		**Validation Accuracy**	
	CNN	**GCNN (θ=0)**	**CNN**	**GCNN (θ=0)**	**CNN**	**GCNN (θ=60)**	**CNN**	**GCNN (θ=60)**
Epoch	**D=0**	**D=0**	**D=0**	**D=0**	**D=0.5**	**D=0.5**	**D=0.5**	**D=0.5**
1	0.0273	0.1501	0.0383	0.2329	0.0235	0.0792	0.0628	0.2363
2	0.1160	0.4216	0.2320	0.4443	0.0505	0.2359	0.0917	0.4559
3	0.3332	0.6220	0.3246	0.5437	0.0835	0.4440	0.1464	0.5725
4	0.5471	0.7572	0.5579	0.6569	0.1553	0.5867	0.3349	0.6492
5	0.6696	0.8505	0.6401	0.6982	0.2500	0.6771	0.4339	0.7043
6	0.7623	0.9036	0.6991	0.7004	0.3533	0.7509	0.5381	0.7068
7	0.8163	0.9248	0.7361	0.7516	0.4237	0.7913	0.5549	0.7400
8	0.8717	0.9429	0.7172	0.7297	0.4819	0.8149	0.6367	0.7632
9	0.9093	0.9598	0.7495	0.7873	0.5373	0.8444	0.6931	0.7826
10	0.9305	0.9630	0.7538	0.7611	0.5962	0.8708	0.7077	0.7856

Table 5.12: Comparison of GCNN with constant θ and first CNN model results

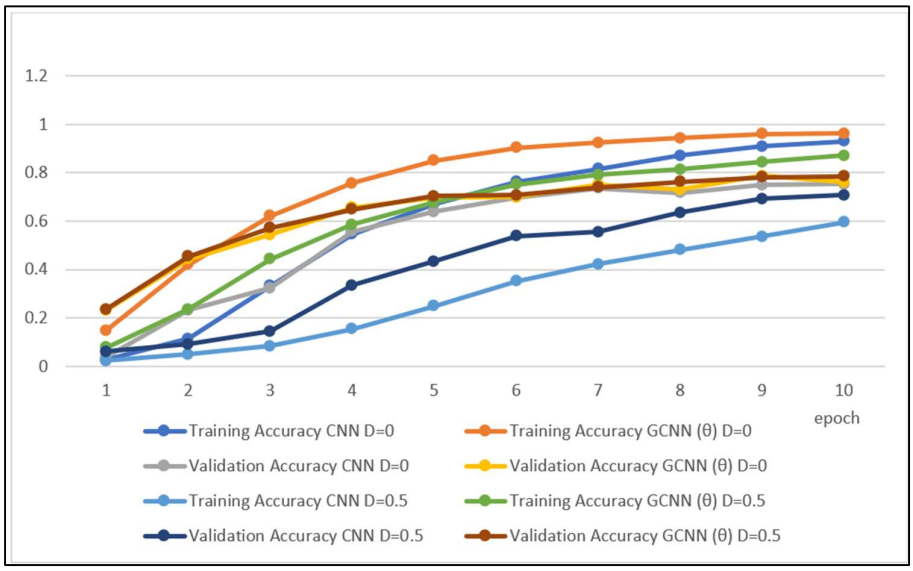


Figure 5.18: Accuracy of the GCNN with constant θ and first CNN model

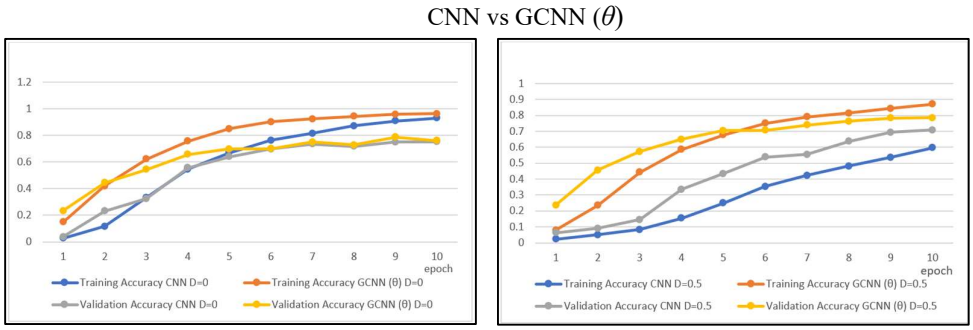


Figure 5.19: Accuracy of the GCNN with constant θ and first CNN model for dropout 0.0 and 0.5

IV. Gamma, γ

The highest accuracy levels for dropouts 0.0 and 0.5 are bolded in Table 5.8. When γ =0 and γ=180, the highest accuracy level can be obtained for the dropout 0.0 and 0.5 respectively. Training accuracy is decreased when introducing the dropout effect. It can be observed that there is no significant pattern visible when increasing the γ value. Considering Figure 5.20, it can conclude that, for dropout 0.0, the training and validation accuracy of the GCNN with γ =0 becomes almost similar to CNN accuracy for the higher epoch. Considering Figure 5.21, for dropout value 0.5, validation and training accuracy of GCNN with γ =180 is higher than the corresponding CNN accuracy.

	Training Accuracy		Validation Accuracy		Training Accuracy		Validation Accuracy	
	CNN	GCNN (γ=0)	CNN	GCNN (γ=0)	CNN	GCNN (γ=180)	CNN	GCNN (γ=180)
Epoch	D=0	D=0	D=0	D=0	D=0.5	D=0.5	D=0.5	D=0.5
1	0.0273	0.1435	0.0383	0.3069	0.0235	0.0807	0.0628	0.3035
2	0.1160	0.4219	0.2320	0.5045	0.0505	0.1897	0.0917	0.3285
3	0.3332	0.5830	0.3246	0.5536	0.0835	0.2388	0.1464	0.3771
4	0.5471	0.6900	0.5579	0.5394	0.1553	0.3249	0.3349	0.4335
5	0.6696	0.7563	0.6401	0.6759	0.2500	0.4182	0.4339	0.5230
6	0.7623	0.8192	0.6991	0.7150	0.3533	0.4891	0.5381	0.5248
7	0.8163	0.8496	0.7361	0.7245	0.4237	0.5433	0.5549	0.5764
8	0.8717	0.8714	0.7172	0.6771	0.4819	0.5844	0.6367	0.6160
9	0.9093	0.8878	0.7495	0.7271	0.5373	0.6105	0.6931	0.6139
10	0.9305	0.9001	0.7538	0.7021	0.5962	0.6452	0.7077	0.6694

Table 5.13: Comparison of GCNN with constant γ and first CNN model results

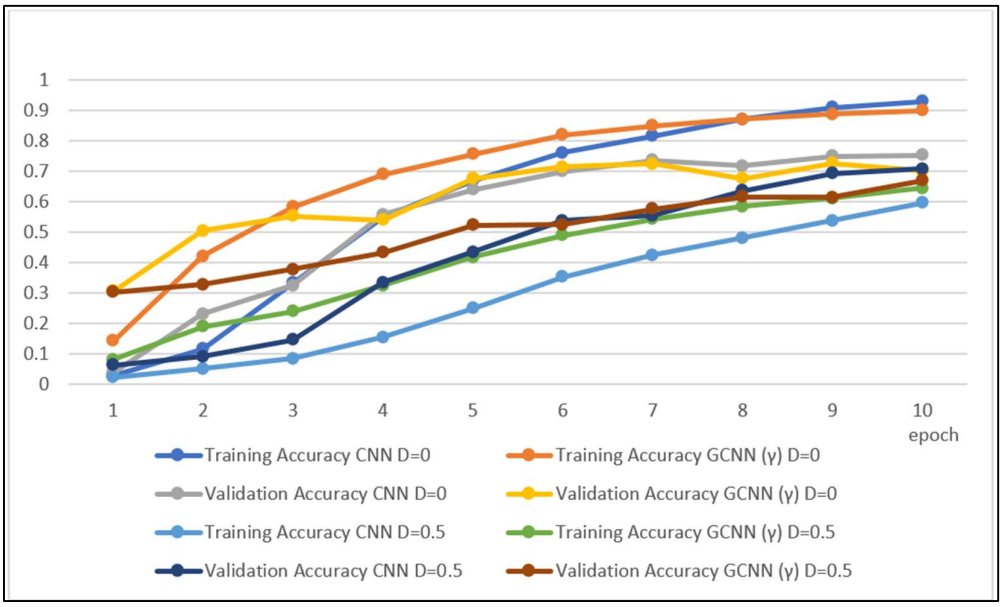


Figure 5.20: Accuracy of the GCNN with constant γ and first CNN model

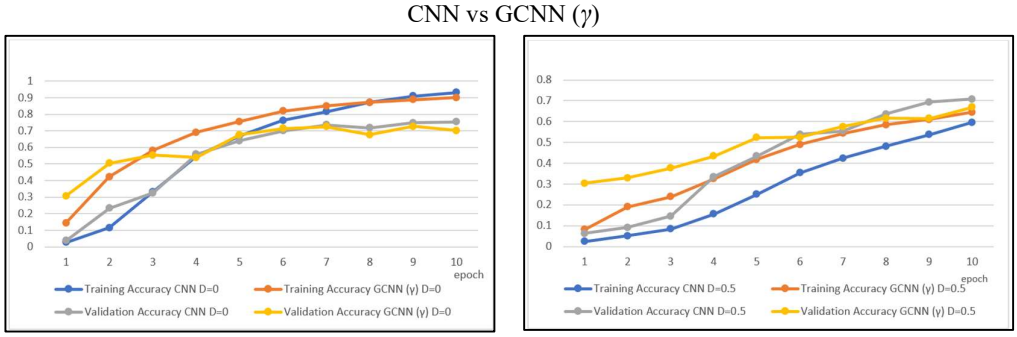


Figure 5.21: Accuracy of the GCNN with constant γ and first CNN model for dropout 0.0 and 0.5

V. **Psi, ψ**

According to Table 5.9 and Figure 5.12, and Figure 5.13, it is clear that there is a domain for the Gabor variable ψ. ψ values available within the range [120, 240] should never be used. The Gabor initialized CNN network has difficulty in training when ψ variable values are within this interval. When adding dropout to the network, the accuracy levels do not significantly change. When ψ =300 highest accuracy level can be obtained for both dropout 0.0 and 0.5.

	Training Accuracy		Validation Accuracy		Training Accuracy		Validation Accuracy	
	CNN	GCNN (ψ=300)	CNN	GCNN (ψ=300)	CNN	GCNN (ψ=300)	CNN	GCNN (ψ=300)
Epoch	D=0	D=0	D=0	D=0	D=0.5	D=0.5	D=0.5	D=0.5
1	0.0273	0.1567	0.0383	0.3448	0.0235	0.0789	0.0628	0.2824
2	0.1160	0.4403	0.232	0.4735	0.0505	0.1863	0.0917	0.4628
3	0.3332	0.5976	0.3246	0.5807	0.0835	0.2968	0.1464	0.5601
4	0.5471	0.7153	0.5579	0.6220	0.1553	0.4096	0.3349	0.5906
5	0.6696	0.7744	0.6401	0.6939	0.2500	0.4971	0.4339	0.6173
6	0.7623	0.8206	0.6991	0.7301	0.3533	0.5700	0.5381	0.6548
7	0.8163	0.8579	0.7361	0.7335	0.4237	0.6036	0.5549	0.6793
8	0.8717	0.8783	0.7172	0.7697	0.4819	0.6536	0.6367	0.7017
9	0.9093	0.9087	0.7495	0.7779	0.5373	0.7001	0.6931	0.7254
10	0.9305	0.9133	0.7538	0.7628	0.5962	0.7173	0.7077	0.7370

Table 5.14: Comparison of GCNN with constant ψ and first CNN model results

Gabor filters are based on the mathematical equation discussed in Section 3.6 with the parameters σ, λ, θ, ψ and γ. When the value of λ is 0, the GCNN network is unable to converge and performs poorly during training. The results show that whenever the value of parameter ψ falls within [120, 240], the network is unable to produce any meaningful findings. The rotating nature of the Gabor filter makes it possible to extract features, hence ideally θ should be given the most emphasis. However, in the above experiment using single parameter value for θ does not show a significant impact. Subject to the constructed Sinhala character dataset, it can be concluded that λ should not be set to 0. The parameter ψ should not fall within the [120 240] range to obtain the optimal accuracy level. Table 5.15 summarizes the results obtained from the experiment conducted under Section 5.3. All results are based on the constructed Sinhala character data set.

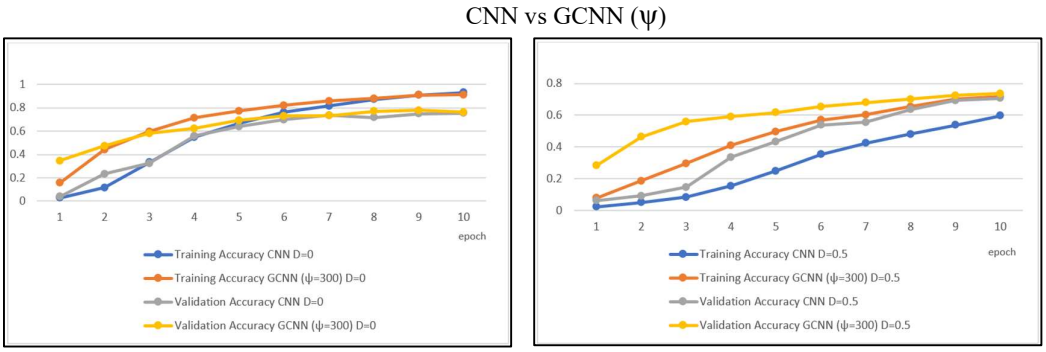


Figure 5.22: Accuracy of the GCNN with constant ψ and first CNN model for dropout 0.0 and 0.5

Parameter	Sigma	Lambda	Theta	Gamma	Psi
Impact	Low	Low	High	Low	High
Best Value Range	[2-14]	[20-100]	0	180	[0-120]
Worst Value Range	-	0	-	-	[120-240]

Table 5.15: Summary

5.3.4 Testing on Gabor initialized CNN model

In this experiment, the collection of Gabor filters applied the proposed GCNN architecture discussed in Section 4.3.3. The parameter values of the Gabor filter bank are obtained by considering the results obtained in Section 5.3.3 and the trial-and-error method to achieve higher accuracy. For the test, 30 epochs are considered. Filter size will remain at 5×5 at 32 filters in the initial layer. The grid search initialization technique was applied to determine the Gabor filter bank. The test was carried out for four different dropout values D= (0.0,0.25,0.5,0.8). Results are discussed in section 5.3.5.

5.3.5 Results

The training and validation accuracy of the Gabor initialized CNN model are listed in Table 5.16. In here 30 epochs has been used for four different dropout values D= [0.0 ,0.25, 0.5,0.8]. Training and validation curves are also indicated in Figure 5.23 for four different dropout values. Finally, the comparison of two different CNN architectures and Gabor initialized CNN architecture is listed in Table 5.17 for each epoch.

Dropout	Training Accuracy	Training Lost	Testing Accuracy	Testing Loss
0.0	0.9575	0.2896	0.7503	3.4728
0.25	0.9515	0.2652	0.8067	2.1316
0.5	0.9030	0.3127	0.7908	1.0172
0.8	0.6455	1.1720	0.6659	1.1311

Table 5.16: Results of the Gabor initialized CNN model

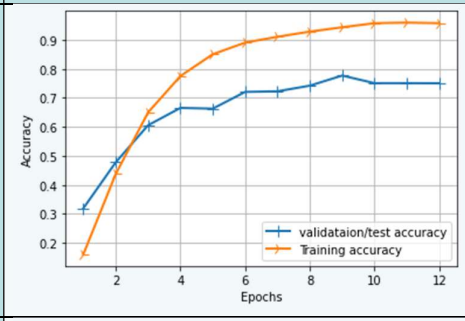
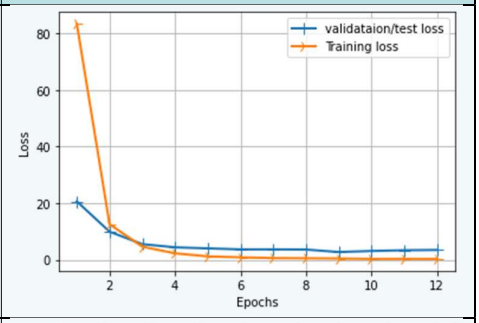
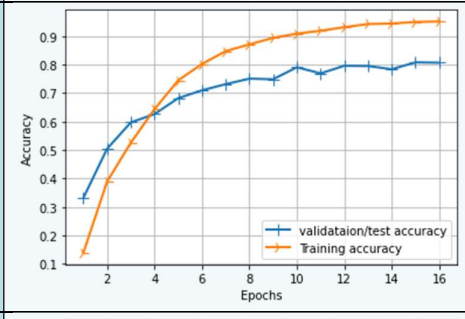
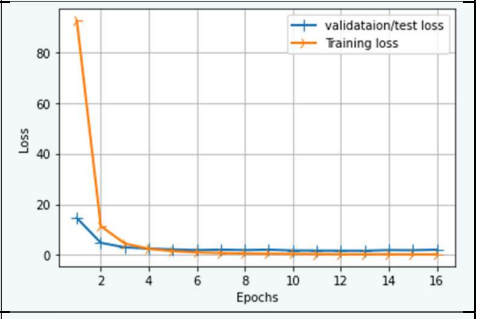
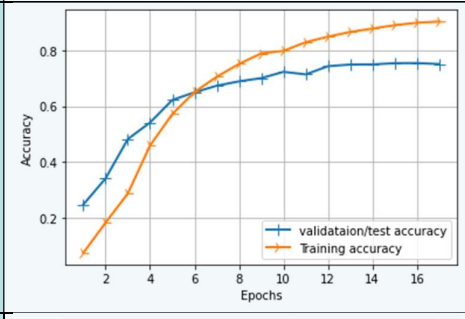
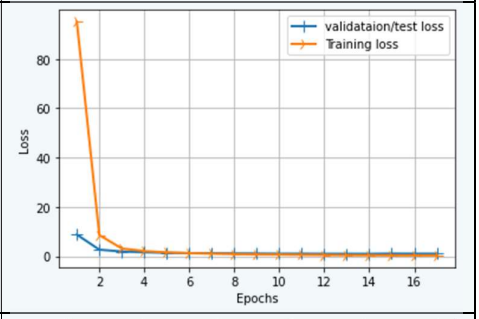
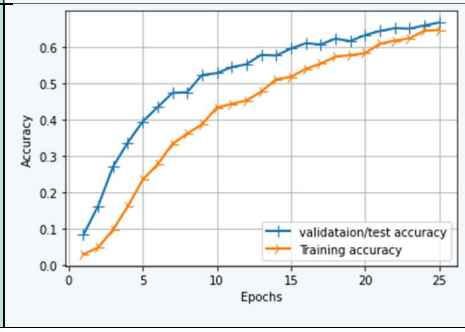
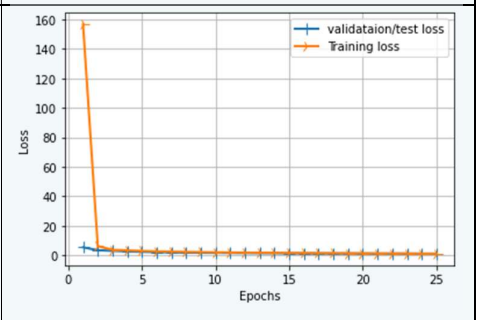
Drop out	Training vs Validation accuracy	Training vs Validation loss
0.0		
0.2		
0.5		
0.8		

Figure 5.23: Training and validation curves of the GCNN model

Epoch	Training Accuracy			Validation Accuracy		
	Glorot 1st Model (D=0.5)	Glorot 2nd Model (D=0.5)	Gabor (D=0.25)	Glorot 1st Model (D=0.5)	Glorot 2nd Model (D=0.5)	Gabor (D=0.25)
1	0.0235	0.0232	0.1392	0.0628	0.0568	0.3319
2	0.0505	0.0525	0.3892	0.0917	0.1507	0.5045
3	0.0835	0.1177	0.5258	0.1464	0.2514	0.5971
4	0.1553	0.2009	0.6438	0.3349	0.3577	0.6263
5	0.2500	0.2821	0.7448	0.4339	0.4580	0.6823
6	0.3533	0.3427	0.8022	0.5381	0.5118	0.7099
7	0.4237	0.4237	0.8473	0.5549	0.5428	0.7314
8	0.4819	0.4753	0.8711	0.6367	0.5669	0.7508
9	0.5373	0.5198	0.8941	0.6931	0.6259	0.7486
10	0.5962	0.5729	0.9087	0.7077	0.6457	0.7908

Table 5.17: Comparison of two different CNN networks and Gabor initialized CNN

5.3.6 Discussion

According to the result listed in Table 5.16, the highest training and validation accuracy was achieved at 0.25 dropout value. Compared with the first CNN model results, the training accuracy of the Gabor initialized CNN is reduced by approximately 3%, and the validation accuracy is reduced by approximately 5%. But considering Table 5.17, it can conclude that the convergence speed of the GCNN is higher than the proposed CNN model.

5.4 Summary

This chapter presents the findings from the two different CNN architectures and Gabor initialized CNN are discussed in detail. In addition, the importance of the parameters of the Gabor filter was investigated. A discussion section corresponding to each test is also included. The next chapter comprises the conclusion of this research project.

Chapter 6

Conclusion

6.1 Introduction

The research aims to use deep learning to recognize Sinhala handwritten characters. HCR is extensively employing for many foreign languages such as English and Japanese. It is quite difficult to adopt an HCR model in South Asian languages because of the shape and compound letters. Sinhala characters are very special because of their unique shape, with curves and horizontal lines. Many academics' attentions have recently been drawn to deep learning for character recognition. Deep neural networks achieve outstanding performance in classification and recognition problem solving. CNN is a deep learning paradigm that encourages more effective character recognition and image categorization. This research focuses on the recognition of Sinhala handwritten characters using CNN. The conclusion of this study is discussed in Section 6.2, and the future work related to this study is discussed in Section 6.3.

6.2 Conclusion

Since the unavailability of an appropriate Sinhala character image dataset for the research, a sample of 6000 character image dataset was constructed for all Sinhala characters in the alphabet. This research mainly proposed two CNN architectures and one Gabor initialized CNN architecture for the Sinhala character recognition. The first CNN model consisted of five convolutional layers with maxpooling layer between some convolutional layers. In the second CNN architecture, the input layer is directly fed into the maxpooling layer and then fed into four convolutional layers. Both CNN architectures introduced a dropout layer with 0.0,0.25,0.5,0.8 values before the fully connected layer to obtain how the dropout effect affects CNN's performance. Even though research conducted by W.V.S.K.Wasalthilake [3] claimed that the accuracy could be improved by feeding input images directly to the maxpooling layer before the convolution layer, this research could not achieve such improvement in the performance of CNN for the constructed character dataset. The first CNN architecture achieved 0.9633 training accuracy and 0.9014 testing accuracy when applying the 0.5 dropout value for 60 classes of characters.

A Gabor filter, invented by Dennis Gabor, is a linear texture analysis filter defined by 2D mathematical equations with variables Sigma(σ), Lambda (λ), Theta (θ), Gamma (γ), Psi (ψ). The value of these parameters defines the characteristics of the Gabor filter. By changing these variables, a Gabor filter bank can be created. Gabor initialized CNN structure is introduced by implementing a CNN with Gabor filters in the foremost and subsequent convolutional layers. The main objective of the GCNN structure was to achieve reasonable energy savings regarding computation time by decreasing the complexity of the CNN network by utilizing error adaptability. In the GCNN architecture, trained layers of CNN are replaced by fixed Gabor kernels and hence CNN can illuminate these expensive computation methods within the layers with the same or higher accuracy levels.

This research conducted an experiment to obtain a better understanding and importance of the Gabor parameters. The design of the experiment and the results which described the impact of each parameter on overall GCNN architecture are discussed in Chapter four in detail. According to the constructed Sinhala character image dataset, Gabor parameter λ should not be equal to 0, and the value of parameter ψ should not be in a range between $[120 - 240]$ to obtain the optimal accuracy level. In the GCNN architecture, trained layers of CNN are replaced by fixed Gabor kernels, which is supposed to reduce expensive computation within the layers. The main objective of using the GCNN structure to recognize handwritten Sinhala characters was to achieve reasonable energy efficiency in terms of computation time while maintaining the same or higher accuracy level as CNN.

When comparing the first CNN model with a dropout 0.5 and GCNN with a dropout 0.25 their eventual accuracy levels are close to each other. But GCNN structures converge to the accuracy level faster than the corresponding CNN model. So, Gabor filters can be applied in the initial layer of the CNN model instead of convolutional learning in the first layer, even though Gabor initialized CNN reduces the accuracy level by approximately 5% under the same hyperparameters. In addition, by trial-and-error method, it is found that constructed Sinhala character image dataset provides a better accuracy level with a low number of filters in the collection Gabor filters. As a part of this research, the impact of the Gabor variables is investigated. The five parameters, Sigma, Lambda, Theta, Gamma, and Psi in the Gabor filter have a remarkable impact

on training performance as some values cause to decrease, increase performance or unable to converge. It was discovered that variables θ and ψ have a greater impact on training. It was found that the λ should not be equal to zero. When λ is greater than zero, it behaves similarly to the variables σ and γ in the training phase.

In this research, Gabor filter initialization was applied to the first layer of the CNN architecture. It is observed that the Gabor initialized CNN trains faster than the corresponding CNN architecture in the initial epochs. It is concluded that the Gabor initialized CNN is capable of training faster in the early phases of the training process. This can prove that Gabor initialized CNN can enhance the convergence of the network and reduce the training time with the same or slightly less accuracy. All of these conclusions were based on the constructed Sinhala character image dataset. Considering the strength and limitations of proposed CNN architectures and Gabor initialized architecture, the more significant architecture was selected to recognize the Sinhala characters.

6.3 Future work

This research has applied the Gabor initialization method only for the initial layer of the CNN. Actually, it can apply to all the convolutional layers. It is important to experiment on how the performance of the GCNN varies when introducing the Gabor initialization method for other convolutional layers. All experiments conducted above use the grid search method for assigning values for the parameters of the Gabor filter. It is important to examine the behavior of the GCNN network with other Gabor initialization methods, such as random initialization. Experiments can be conducted to find the performance of the GCNN architecture when varying the filter size.

References

[1] Hewavitharana, S. et al. (2015) 'Off-line Sinhala Handwriting Recognition using Hidden Markov Models Sri Lanka Institute of Information Technology, Colombo, Sri Lanka', (November).

[2] Rohana K. Rajapakse, a. R. W., and Seneviratne, E. K. (1995) 'a Neural Network Based Character Recognition System for Sinhala Script', Vasa, (January 1995). Available at: http://medcontent.metapress.com/index/A65RM03P4874243N.pdf

[3] W.V.S.K.Wasalthilake ,T.Kartheeswaran (2020), Sinhala Handwritten Character Recognition Using Convolution Neural Networks.

[4] Nilaweera, N. P. T. I., Premeratne, H. L. and Sonnadara, D. U. J. (2014) 'Comparison of Projection and Wavelet Based Techniques In Recognition of Sinhala Handwritten Scripts',(September 2007). Available at:https://s3.amazonaws.com/academia.edu.documents30429910/2007_CSSL_wavelet.pdf

[5] Silva, C. M., Jayasundere, N. D. and Kariyawasam, C. (2016) 'Contour tracing for isolated Sinhala handwritten character recognition', 15th International Conference on Advances in ICT for Emerging Regions, ICTer 2015 - Conference Proceedings. IEEE, pp. 25–31. doi: 10.1109/ICTER.2015.7377662

[6] Alekseev, A., & Bobe, A. (2019, November). GaborNet: Gabor filters with learnable parameters in deep convolutional neural network. In 2019 International Conference on Engineering and Telecommunication (EnT) (pp. 1-4). IEEE.

[7] Kavitha, B. R., & Srimathi, C. (2019). Benchmarking on offline Handwritten Tamil Character Recognition using convolutional neural networks. Journal of King Saud University-Computer and Information Sciences.

[8] ZHANG, L. X., YU, P. F., LI, H. Y., & LI, H. S. (2018). Research on Character Recognition Technology of New Tai Lue Based on Gabor Feature and SVM. DEStech Transactions on Computer Science and Engineering, (ccme).

[9] Nugroho, N. E. W. (2021). Transliteration of Hiragana and Katakana Handwritten Characters Using CNN-SVM. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 15(3).

[10] Guo, H., Liu, Y., Yang, D., & Zhao, J. (2021). Offline handwritten Tai Le character recognition using ensemble deep learning. The Visual Computer, 1-14.

[11] Tsai, C. (2016). Recognizing handwritten Japanese characters using deep convolutional neural networks. University of Stanford, California.

[12] Alom, M. Z., Sidike, P., Taha, T. M., & Asari, V. K. (2017). Handwritten bangla digit recognition using deep learning. arXiv preprint arXiv:1705.02680.

[13] Y. Wang and P. Yu, "Offline Handwritten New Tai Lue Characters Recognition Using CNN-SVM," 2019 IEEE 2nd International Conference on Electronic Information and Communication Technology (ICEICT), 2019, pp. 636-639, doi: 10.1109/ICEICT.2019.8846292.

[14] O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

[15] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)* (pp. 1-6). Ieee.

[16] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, *77*, 354-377.

[17] Wu, J. (2017). Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, *5*(23), 495.

[18]https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization- techniques/

[19]https://towardsdatascience.com/a-comprehensive-guide-of-regularization-techniques-in-deep-learning-c671bb1b2c67

[20] D Gabor. Theory of communication. Part 1: The analysis of information. Journal of the Institution of Electrical Engineers - Part III: Radio and Communication Engineering, 93(26):429–441(12), nov 1946

[21] N Petkov and M.B. Wieling. Gabor filter for image processing and computer vision, 2008.

[22] Yuan, Y., Wang, L. N., Zhong, G., Gao, W., Jiao, W., Dong, J., ... & Xiang, W. (2022). Adaptive Gabor convolutional networks. *Pattern Recognition*, *124*, 108495.

[23] Rai, M., & Rivas, P. (2020, December). A review of convolutional neural networks and gabor filters in object recognition. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 1560-1567). IEEE.

[24] M. Rai and P. Rivas, "A Review of Convolutional Neural Networks and Gabor Filters in Object Recognition," *2020 International Conference on Computational*

Science and Computational Intelligence (CSCI), 2020, pp. 1560-1567, doi: 10.1109/CSCI51800.2020.00289.

[25] Z. Lu, X. Liang, G. Yang and D. Liu, "Small-scale Convolutional Neural Networks with Learnable Gabor Filter for Image Classifications," *2021 4th International Conference on Information Communication and Signal Processing (ICICSP)*, 2021, pp. 425-431, doi: 10.1109/ICICSP54369.2021.9611874.

[26] Jiansheng Chen, Xiangui Kang, Ye Liu, and Z. Jane Wang. Median Filtering Forensics Based on Convolutional Neural Networks. Ieee Signal Processing Letters, 22(11):1849– 1853, 2015.

[27] Pham, L.V. (2019). Gabor Filter Initialization And Parameterization Strategies In Convolutional Neural Networks.

[28] Syed Shakib Sarwar, Priyadarshini Panda, and Kaushik Roy. Gabor Filter Assisted Energy Efficient Fast Learning Convolutional Neural Networks. IEEE ACM International Symposium on Low Power Electronics and Design, pages 1–6, 2017

[29] Gokhan ¨ Ozbulak, Hazm Kemal Ekenel, and Assoc Prof. Gabor Initialized Convolutional ¨ Neural Networks for Transfer Learning. Signal Processing and Communications Application Conference, 26, 2018.

[30] Syed Shakib Sarwar, Priyadarshini Panda, and Kaushik Roy. Gabor Filter Assisted Energy Efficient Fast Learning Convolutional Neural Networks. IEEE ACM International Symposium on Low Power Electronics and Design, pages 1–6, 2017.

[31] de Geus, A. R., Barcelos, C. A., Batista, M. A., & da Silva, S. F. (2019, September). Large-scale pollen recognition with deep learning. In *2019 27th European Signal Processing Conference (EUSIPCO)* (pp. 1-5). IEEE.

[32] Wu, G. (2021, May). Study on Face Recognition Based on Improved CNN and Ensemble Learning. In *2021 33rd Chinese Control and Decision Conference (CCDC)* (pp. 3414-3417). IEEE.

[33] https://en.wikipedia.org/wiki/Sinhala_language

[34] https://www.colombotelegraph.com/index.php/the-evolution-of-the-sinhala-language-an-important-reference/