ANALYSIS OF IMAGE CLASSIFICATION DEEP LEARNING ALGORITHM

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Shivam Kanungo

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

April 2023

Fargo, North Dakota

North Dakota State University
Graduate School

Title

ANALYSIS OF IMAGE CLASSIFICATION DEEP LEARNING ALGORITHM

By

Shivam Kanungo

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Simone Ludwig
Chair

Dr. Kenneth Magel

Dr. Ying Huang

Approved:

04/14/2023
Date

Dr. Simone Ludwig
Department Chair

ABSTRACT

This study explores the use of TensorFlow 2 and Python for image classification problems. Image categorization is an important area in computer vision, with several real-world applications such as object identification/recognition, medical imaging, and autonomous driving. This work studies TensorFlow 2 and its image categorization capabilities. We also demonstrate how to construct an image classification model using Python and TensorFlow 2.

This analysis of image classification neural network problems with the use of Convolutional Neural Network (CNN) on the German and the Chinese traffic sign datasets is an engineering task.

Ultimately, this work provides step-by-step guidance for creating an image classification model using TensorFlow 2 and Python, while also showcasing its potential to tackle image classification issues across various domains.

ACKNOWLEDGMENTS

I'd like to express my sincere gratitude to all those who have contributed to the completion of this paper. First and foremost, I'd like to thank my advisor, Dr. Simone Ludwig, for the valuable guidance, encouragement, and support throughout the research process. Her expertise and insights have been instrumental in shaping this paper.

I would also like to thank my committee members Professor Kenneth Magel, Professor Ying Huang for their interest in my work.

I am grateful to the organizations and individuals who provided us with the data and resources needed to conduct this research. Without their cooperation and support, this study would not have been possible.

Finally, I'd like to thank my family and friends for their unwavering support and encouragement throughout this process. Their love and understanding have been crucial in helping us stay motivated and focused on completing this paper.

Once again, I express my heartfelt appreciation to everyone who has contributed to this paper. Any errors or omissions remain my responsibility.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

Computer vision is an interdisciplinary field encompassing computer science, physics, mathematics, and biology. It strives to automate the extraction of information from digital images. Despite advances in this area, teaching computers how to interpret visual stimuli, identify objects and recognize faces similarly as humans does remains a daunting challenge. Ultimately, computer vision seeks to give computers the capacity to make sense of images just like humans do - or even surpass human performance on some tasks.

Deep learning technology has significantly enhanced computer vision capabilities, enabling it to achieve superhuman performance in tasks such as face verification and handwritten text recognition. Thanks to funding from major IT companies, research in computer vision is flourishing with increased access to visual sensors and data sets. As a result, more complex tasks such as vision-based navigation for autonomous driving, content-based image retrieval, automated annotation/enhancement are being addressed with increasing success.

Computer vision is an exciting and promising field that attracts both experts and newcomers alike. With its interdisciplinary nature, rapid advancements, and ambitious goals, this is truly a thrilling time to be involved in this field. However, despite significant progress made, there remains much work to do before computer vision researchers achieve human-like visual perception; therefore, challenges remain formidable for this research discipline. Nonetheless, computer vision's potential applications are seemingly limitless, guaranteeing its continued impact on various fields for years to come.

Computer vision seeks to extract meaningful, semantic information from images, such as the objects present, their location and number. This problem can be broken down into various subdomains like object classification. Image classification (also referred to as object

classification) involves assigning appropriate labels or classes to images from a predetermined set.

1.1. Image Classification

Image classification is an essential task in computer vision that involves assigning a label or class to an input image based on its visual content. Deep neural network algorithms have revolutionized this field recently, achieving remarkable accuracy rates on large datasets. Typically, deep neural networks go through two main steps to complete image classification: training and testing.

At the training stage, a deep neural network algorithm is presented with an extensive dataset of labeled images and its weights are iteratively adjusted to minimize a loss function that measures discrepancy between predicted labels and true labels. This process typically uses stochastic gradient descent, which adjusts weights according to steepest descent of loss function. As the network continues training, it learns to extract high-level features from images relevant for classification such as edges, corners, and textures. Once complete training has taken place, optimized weights are saved and used during testing phase.

At the testing stage, a deep neural network algorithm is presented with an updated set of unlabeled images and its optimized weights are used to predict their labels. To do this, images must first be preprocessed by normalizing pixel values or cropping and resizing them to a fixed size before passing through the neural network which computes a probability distribution over possible classes. Usually, only one class will be predicted with high probability; however, in certain cases multiple classes may also have high probabilities. Accuracy of classification algorithms is evaluated by comparing predicted labels against true labels on separate validation sets.

Image classification is an essential step in deep learning algorithms. Here are some of the most used techniques.

1.1.1. Convolutional Neural Network

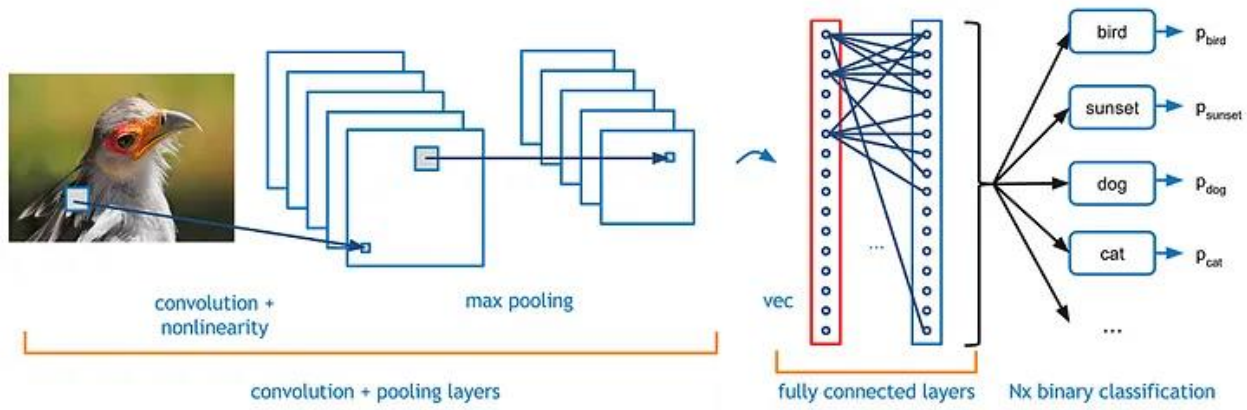

Figure 1: Representation of CNN[6]

Convolutional Neural Network (CNN, or ConvNet) are multi-layer neural networks designed to recognize visual patterns directly from pixel images with minimal preprocessing. It represents an advanced architecture of artificial neural networks. Convolutional neural networks utilize some of the features of visual cortex and have achieved breakthrough results in computer vision tasks. Convolutional neural networks consist of two basic elements, convolutional layers and pooling layers, for efficient computation. Though seemingly straightforward, there are virtually infinite ways to arrange these layers for any given computer vision problem. The core components of a convolutional neural network, such as convolutional and pooling layers, are relatively straightforward to comprehend. Convolutional neural networks offer a powerful solution for modeling complex problems by simply using their basic elements. Their popularity stems from their architecture, with no need for feature extraction required. The system learns to perform feature extraction with the fundamental principle that it uses convolution of image and filters to generate invariant features which are passed along to the next layer. These elements are

then convolved with different filters in order to generate even more abstract, invariant patterns until it generates a final feature/output which is invariant to occlusions.

1.1.2. Support Vector Machine

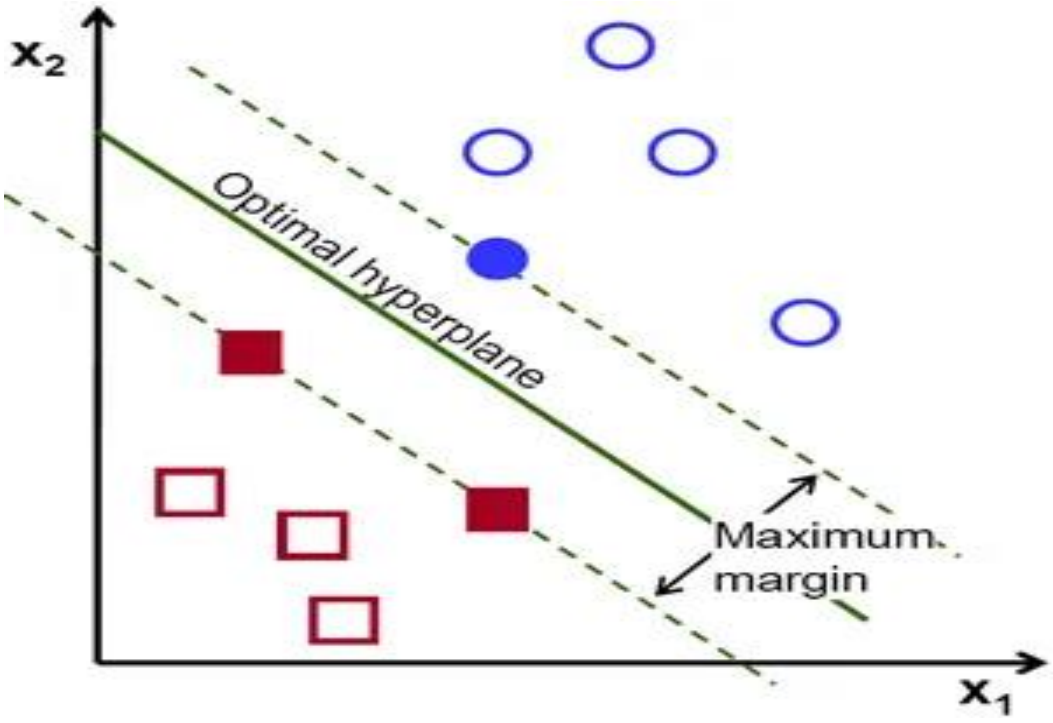

Figure 2: Representation SVM[6]

Support vector machines (SVM) are powerful, yet flexible supervised machine learning algorithms used for both classification and regression tasks. Support vector machines possess a distinct implementation style when compared to other machine learning algorithms. Support Vector Machine models are widely acclaimed for their capacity to handle both continuous and categorical variables. A Support Vector Machine model is simply a representation of different classes on a hyperplane in multidimensional space. Support vector machine will iteratively generate the hyperplane, to minimize error. The objective is to divide datasets into classes and find a maximum marginal hyperplane for each. This algorithm creates a hyperplane or set of hyper-planes in high dimensional space and assigns each class the hyper-plane with the greatest

distance to its nearest training data point. The effectiveness of this algorithm depends on which kernel function is utilized; common kernels include linear kernel, gaussian kernel and polynomial kernel.

1.1.3. K-Nearest Neighbor

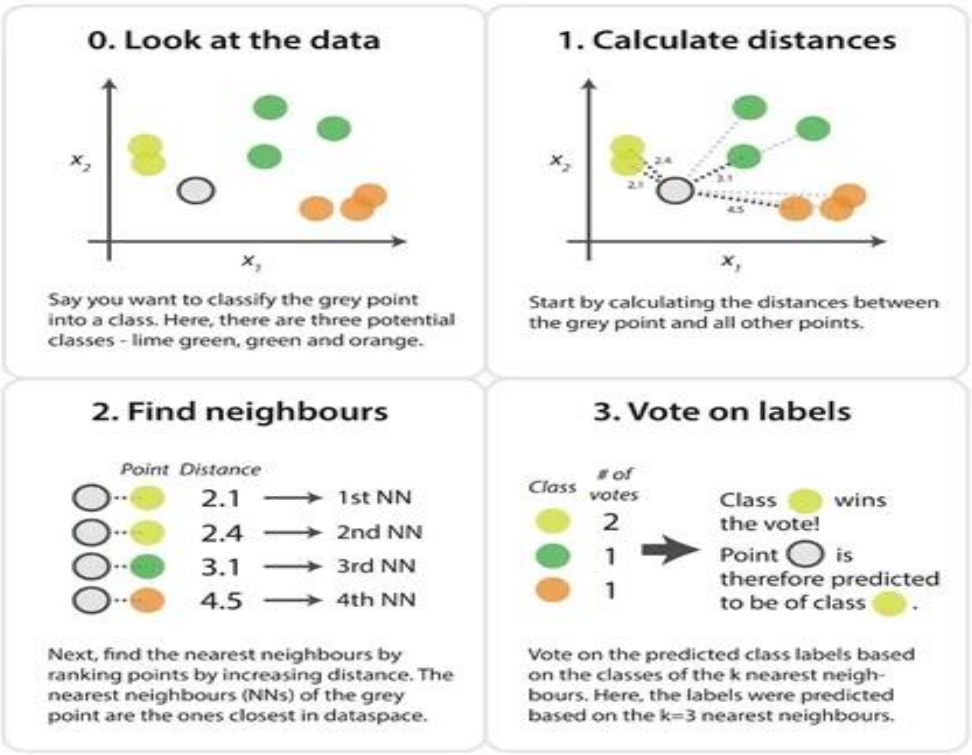

Figure 3: Representation of K-Nearest Neighbor Algorithm[6]

K-Nearest Neighbor is a nonparametric method used for classification and regression, where the input consists of the k closest training examples in the feature space. It's by far the simplest algorithm available. Nonparametric lazy learning algorithm that approximates a function locally but defers all computation until after evaluation of the function. This algorithm utilizes the distance between feature vectors to classify unknown data points by finding the most common class among k-closest examples. To apply the k-nearest Neighbor classification, we need to define a distance metric or similarity function; common choices include Euclidean distance and Manhattan distance. The output will be class membership. An object's classification

is determined by a plurality vote among its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is usually small). If k = 1, then it simply belongs in that neighbor's class. Condensed Nearest Neighbor (CNN, also known as Hart algorithm) reduces this data set for K-Nearest Neighbor classification significantly.

1.1.4. Naïve Bayes Algorithm

Naive Bayes classifiers are a collection of algorithms based on Bayes' Theorem. It isn't one single algorithm but rather an entire family where each pair of features being classified is independent from one another. Naive Bayes is an intuitive method for building classifiers: models that assign class labels to problem instances represented as vectors of feature values, where these labels are drawn from a pre-specified set. Naive Bayes classifiers assume that each feature is independent of all others in a class variable, leading to fast and scalable computation for binary or multi-class classification tasks. Naive Bayes are popular algorithms used for text classification and spam email classification, since they require doing a large number of counts. While they can be trained on small datasets with ease, the algorithm still has limitations as it considers all features to be unrelated and thus cannot learn the relationship between them. While individual features may have importance, naive Bayes cannot identify relationships among them. Different types of naive bayes algorithms exist such as Gaussian naive bayes, multinomial naive bayes and Bernoulli naive bayes.

1.1.5. Random Forest Algorithm

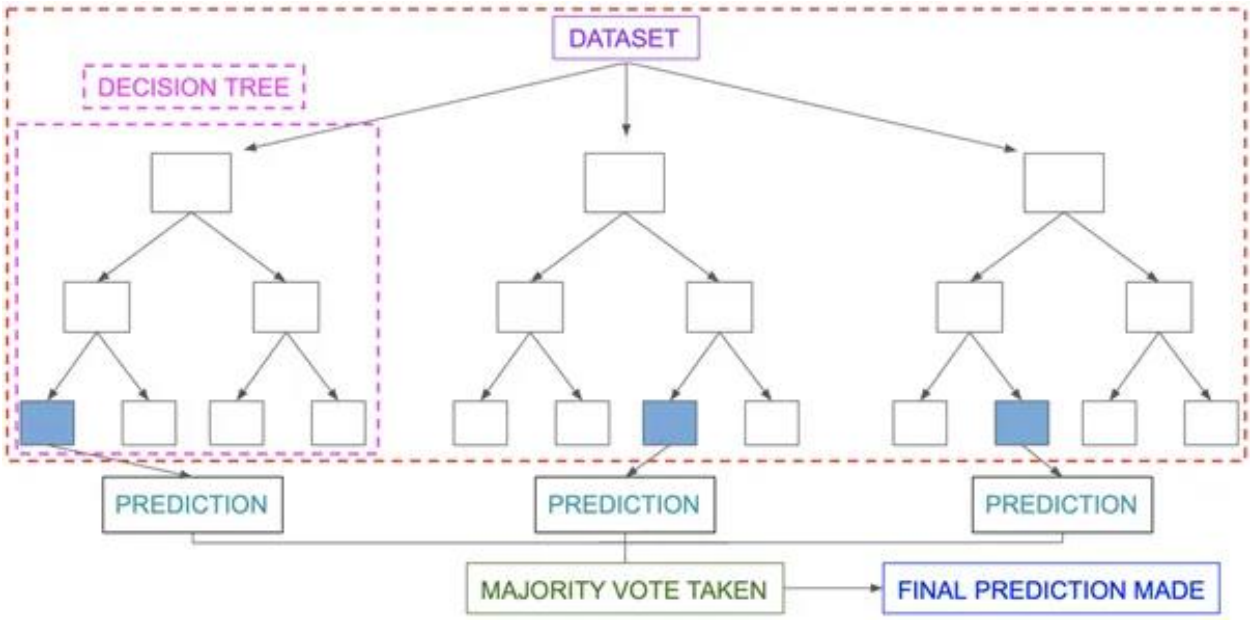

Figure 4: Representation of Random Forest Algorithm[6]

Random forest is a supervised learning algorithm used for both classification and regression problems. Similar to how trees make up a forest, random forest algorithms create decision trees from data samples and then obtain predictions from each one before selecting the best solution through voting. An ensemble method such as decision trees is superior to one single tree because it reduces overfitting by averaging the results. Random forest is a classification algorithm composed of many decision trees that utilize bagging and feature randomness when building each individual tree in order to form an uncorrelated forest of trees whose prediction by committee is more accurate than any individual tree's prediction.

1.2. Neural Network

Neural networks (NNs), also known as artificial neural networks (ANNs), are a type of machine learning tool that can efficiently process information, recognize common patterns, or discover new ones, and approximate complex processes. The structure of these networks is what

makes them so powerful. The way neurons work in our brains is well-known, as they are the fundamental building blocks of our thoughts and actions.

1.2.1. Biological Motivation

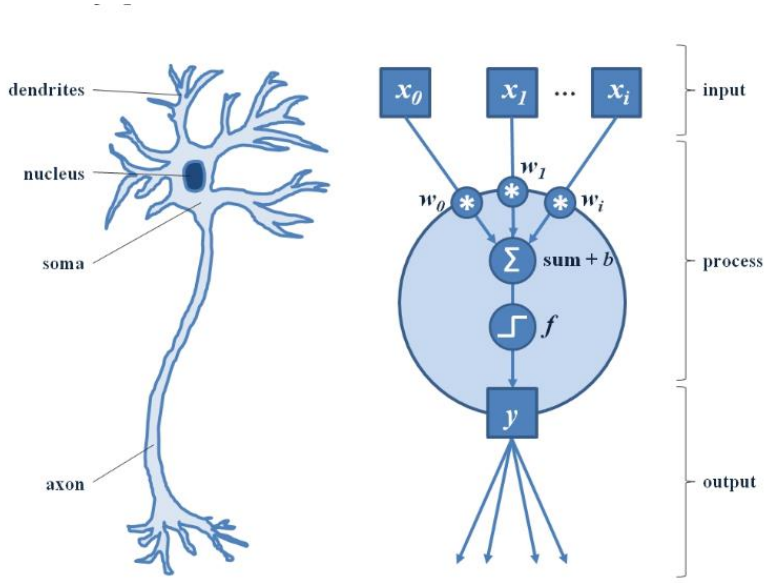

Figure 5: Structure of both biological neuron and NN model[5]

Neural networks are inspired by how animal brains work, as they consist of intricate networks of neurons that communicate with one another and process sensory input (in the form of electrical and chemical signals) to generate thoughts and actions. Dendrites, which are cell fibers extending from the cell body and carrying electrical signals from synapses (junctions with preceding neurons), provide each neuron with its electrical inputs. Once a certain threshold of electrical stimulation is exceeded, activation occurs and an electrical impulse travels along the neuron's axon to several synapses connecting other neurons. Each neuron acts as a simple signal processing unit which when combined produces complex thoughts and behaviors like those we are currently experiencing.

1.2.2. Mathematical Model

An artificial neuron is conceptualized based on its biological counterpart, which takes several input signals and generates an output by adding them together and applying an activation function. In figure 6, On the left, a simplified biological neuron is illustrated, while on the right its artificial counterpart. In an artificial neuron, inputs are typically weighted - that is, each input is multiplied by its unique weight. During the training phase, weights are adjusted so that a neuron responds appropriately to specific features. Furthermore, some neurons may possess a bias parameter which is also trained and used as an offset in the summation process. Mathematically, suppose we have a neuron that takes two input values, x0 and x_1, each of which is weighted by a factor w0 and w_1, respectively. The values are then summed together, and the optional bias b is added to this weighted sum. After adding them together, the values are summed together, and an optional bias b is added to this weighted sum. We can express the input values as horizontal vector x and weights as vertical vector w for simplicity's sake.

$$x = (x_0, x_1), \qquad \omega = \begin{pmatrix} \omega_0 \\ \omega_1 \end{pmatrix}$$

The dot product between two vectors provides the weighted summation:

$$x \cdot \omega = \sum_i x_i \omega_i \; = \; x_1\omega_1 + \; x_2\omega_2$$

1.3. Activation Functions

Activation functions are critical elements in image classification deep learning algorithms as they introduce nonlinearity into the network, enabling it to learn complex patterns and make accurate predictions. The activation function applies to each neuron's output, determining whether that neuron should be activated or not based on input received.

Some commonly used activation functions in image classification deep learning algorithms include:

1.3.1. Rectified Linear Unit (ReLU)

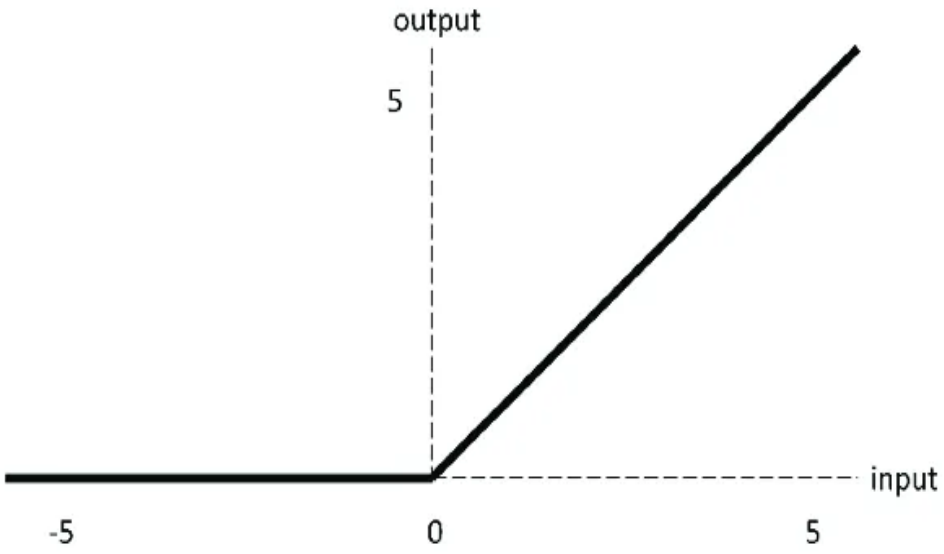

Figure 6: ReLU Activation function representation[9]

ReLU is the most commonly employed activation function and it is defined as f(x) = max(0, x). It offers an efficient computation method which helps reduce the vanishing gradient problem. ReLU stands for Rectified Linear Unit and it's a popular activation function in deep learning models.

This implies that for any input x, the ReLU function returns only the maximum of both 0 and x. Therefore, this non-linear activation function helps deep learning models learn complex non-linear relationships between inputs and outputs.

One of the key advantages of using the ReLU activation function is its computational efficiency, especially when compared to other activation functions like sigmoid or tanh. This is because only simple mathematical operations like addition and comparison must be performed, which can be quickly computed on modern hardware. Furthermore, using this activation function

helps prevent vanishing gradient problems that may arise in deep learning models when using other activation functions like sigmoid or tanh. That is because its derivative has either 0 or 1 which makes it easier for models to backpropagate errors and update weights during training.

1.3.2. Sigmoid

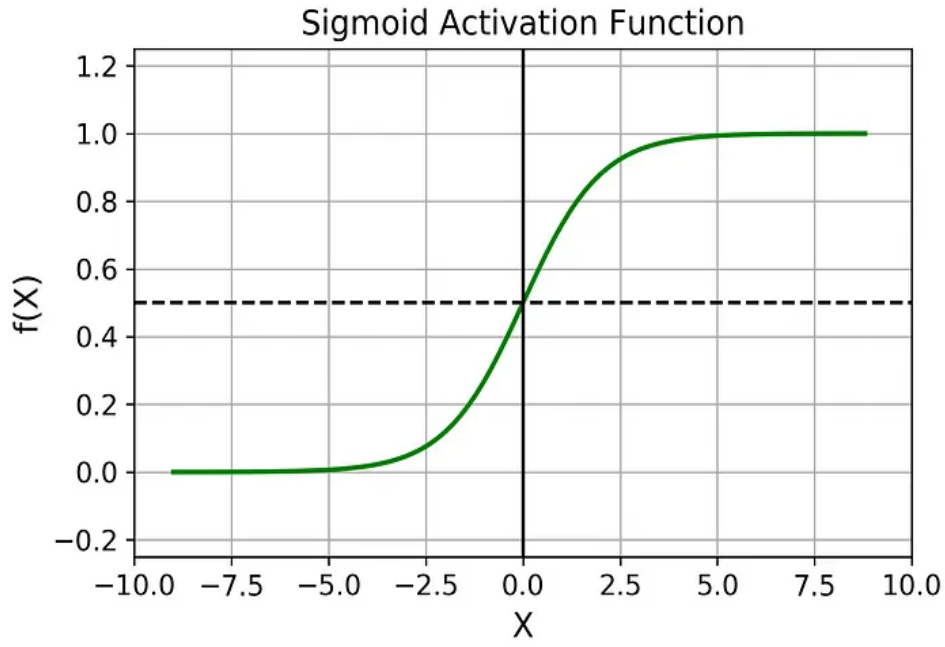

Figure 7: Sigmoid Activation function representation[9]

The sigmoid activation function is a widely used activation function in neural networks, particularly in binary classification problems. It maps any input value to a value between 0 and 1, making it suitable for producing binary outputs (0 or 1) based on a threshold. The sigmoid function has the following mathematical expression: f(x) = 1 / (1 + exp(-x)), where x is the input to the function. The output of the function is always between 0 and 1.

The sigmoid function has a characteristic S-shaped curve that starts at 0 when x is negative, rises sharply around x = 0, and approaches 1 as x becomes large. This makes the function ideal for binary classification problems where the output is either 0 or 1. The function is

differentiable, which allows for gradient-based optimization methods like stochastic gradient descent to be used during model training.

1.3.3. Tanh

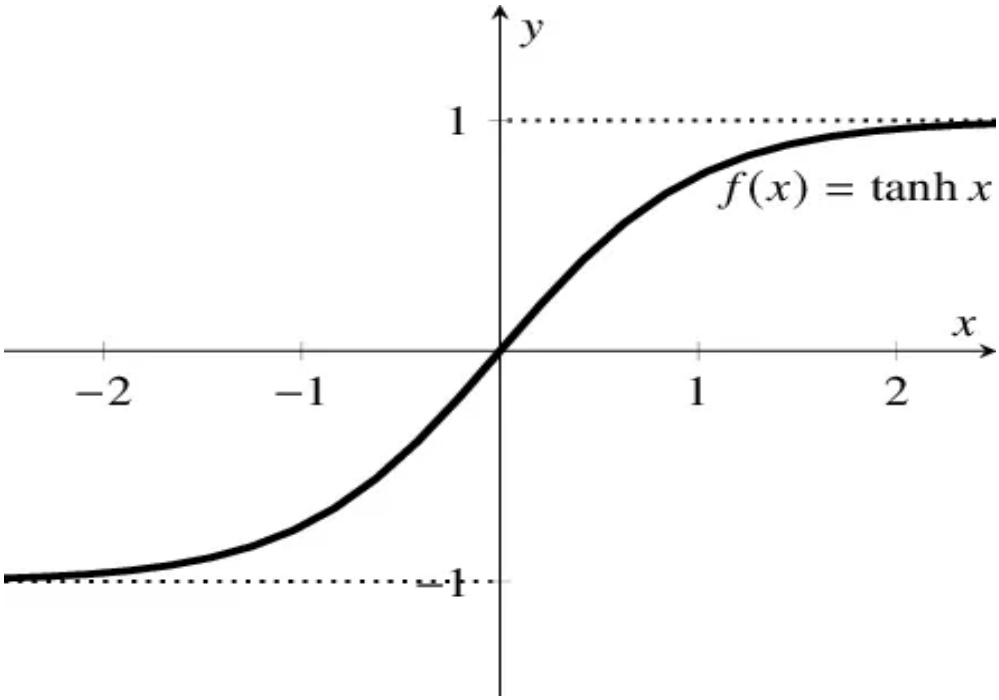

Figure 8: Tanh Activation function representation[9]

The hyperbolic tangent function, or tanh for short, is an activation function widely used in deep learning models. It's a mathematically smoothed version of the sigmoid function with a range of -1 to 1. This function maps input values into this range and helps solve vanishing gradient problems.

1.3.4. SoftMax

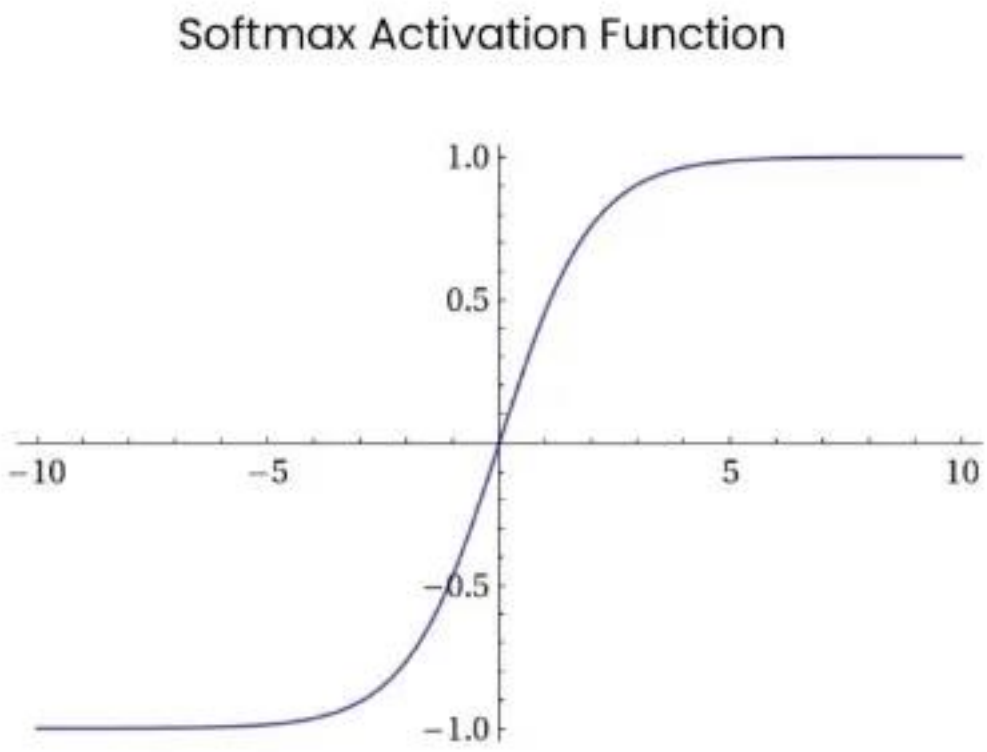

Figure 9: SoftMax Activation function representation[9]

The SoftMax activation function is often employed in the output layer of a neural network for multiclass classification problems. It takes a vector of real-valued scores and transforms it into an estimated probability distribution over classes. The SoftMax function normalizes these scores so they all add up to 1, making it simpler to interpret the output as probabilities.

Selecting the ideal activation function is critical as it can significantly impact the accuracy and convergence rate of a model. We chose ReLU as it best suits our neural network model while being computationally efficient due to its simpler computation compared to other activation functions like sigmoid or tanh. This efficiency enables faster training and inference times in deep learning models.

2. RELATED WORK

- In the study "Convolutional Neural Network (CNN) for Image Classification of Indonesia Sign Language Using Tensorflow", Olivia Kembuan, Gladly Caren Rorimpandey and Soenandar Milian Tompunu Tengker developed an image classification model using Convolutional Neural Network (CNN) architecture and Tensorflow library. The model was trained on 2659 images representing 26 letter categories from Indonesian Sign Language (BISINDO) dataset which was divided into training and validation sets. The aim of this research was to develop an image recognition system using CNN and Tensorflow algorithms. Google Colaboratory provided the model implementation, using Rectified Linear Unit (ReLU) activation function. After 5 epochs on both training and validation datasets, our model achieved an accuracy rate of 96.67% on both. In image classification testing with multiple images of alphabet characters, we observed 100% accuracy rates for each character tested. In order to achieve success with this model, it was necessary to decide on the number of epochs and batch size for training dataset. Furthermore, the research addressed the need for image recognition system for Indonesian Sign Language (BISINDO), which is commonly used by speech and hearing-impaired individuals in Indonesia. Finally, the Convolutional Neural Network System for Image Classification using Indonesian Sign Language (BISINDO) using Tensorflow was successfully implemented with high accuracy rates.
- Jubin Dipakkumar Kothari's study "A Case Study of Image Classification Based on Deep Learning Utilizing Tensor Flow" explores image classification using Deep Neural Network (DNN) or Deep Learning techniques using Tensor Flow system.

Python programming language was chosen due to its compatibility with Tensor Flow. The focus of the research was flower classification, using five types of flowers as testing subjects. DNN proved most accurate at producing 90.585% accuracy rate for roses while all other flowers achieved average accuracy rates close to 90% or higher for all types studied. The study achieved its three objectives, all of which are critical to the conclusion. The DNN method was extensively investigated in detail - from assembly and model training to image classification - with emphasis placed on its role in managing accuracy and avoiding issues such as overfitting. The implementation of deep learning using the Tensor Flow framework produced impressive results. The model was able to simulate, train and classify with up to 90% accuracy for three types of plants.

- Shefali Arora and M.P.S Bhatia's paper "Handwriting Recognition Using Deep Learning in Keras" describes an approach to classifying handwritten images from the MNIST dataset using two architectures: feedforward neural networks and convolutional neural networks. The Python library Keras is used for classification, while Stochastic Gradient Descent is employed for model optimization. After evaluating both models using various metrics, researchers concluded that convolutional neural networks outperformed feedforward neural networks in terms of accuracy. This study employed a feedforward neural network with two hidden layers with 512 neurons each, followed by an output layer of 10 neurons to classify digits 0-9. To train the model, researchers trained it using 60,000 examples and tested it on 10,000 samples - yielding an accuracy rate of 90% after five iterations. Contrastingly, a convolutional neural network achieves an accuracy rate of 95.63% in classifying

images. This paper highlights the effectiveness of CNNs for image recognition and suggests further exploration into promising neural network technologies such as RNN and LSTM. Furthermore, it suggests extending CNNs to recognize facial expressions and other biometric traits used for identification across various applications.

- Treesukon Treebupachatsakul and Suvit Poomrittigul's study "Bacteria Classification using Image Processing and Deep Learning" sought to apply image classification and deep learning methods to classify bacteria genera. They proposed implementing a recognition system using Python programming with Keras API with TensorFlow Machine Learning framework, testing it on two genera of bacteria: Staphylococcus aureus TISTR 746 and Lactobacillus delbrueckii TISTR 1339. Digital images of both bacteria were taken using an Optika B-292 Biological Microscope equipped with an Optikam B3, Italy; they created their own dataset with both types of bacteria images for comparison. This study employed LeNet Convolutional Neural Network (CNN) architecture to classify bacteria images. They separated each dataset into training and test datasets by an 80/20 split percentage, respectively. Results showed that when applying more epochs, Training Loss and Accuracy would improve. Overall, results demonstrated the proposed method could accurately recognize bacteria genera with high precision; both high-resolution and standard resolution bacteria images had training accuracy levels exceeding 75% when trained over 4 epochs at higher resolutions. Researchers suggested that future investigations could enhance the accuracy of standard resolution bacteria images datasets and compare other CNN methods such as ResNET and AlexNET for comparison.

- Analysis done by Lokesh Srikakolupa served as a good reference. The deep learning model was developed using a sequential approach and was run on one of the same datasets we used i.e., the GTSRB[10]. Python scripting language was utilized for the scripts and TensorFlow and Keras API were used for creating hidden layers and the study. The training was done for 15 epochs and achieved the training loss of 0.1951, training accuracy of 0.9479, validation loss of 0.0435, and validation accuracy of 0.9.
- Project published by Hossam Fakher on image classification algorithm used Chinese Traffic Dataset[11] for the algorithm to train. Again, a sequential approach to build the DNN was used. The scripts were written using python programming language and TensorFlow and Keras API libraries were used to build the DNN and conduct analysis on the dataset. The best model achieved during the training had training loss of 0.1355, training accuracy of 0.9673, validation loss of 0.0378, and validation accuracy of 1.000.

These papers and experiments serve as an excellent starting point for those interested in image classification using TensorFlow and Keras. They demonstrate the efficacy of deep learning approaches for image classification, offering insights into the architecture and implementation of NNs and learning algorithms.

3. IMPLEMENTATION

Prior to Image Classification, image preprocessing is necessary. This step reduces undesirable distortions and highlights important image features that can benefit computer vision models. Image pre-processing includes reading the image, resizing it, and performing Data Augmentation techniques like gray scaling, reflection, Gaussian blurring, Histogram equalization rotation and translation.

Next comes detection, which helps locate an object in an image by segmenting the picture and recognizing its position.

Following Detection is the feature extraction and training stage, which is essential for recognizing captivating image patterns with statistical or deep learning methods. Once these features are specific to a class, they are taught by the model for it to differentiate between different classes - this process is known as model training.

Finally, detected objects are classified into predefined classes using an appropriate classification technique. This involves comparing image patterns with target patterns and categorizing them correctly into their appropriate class. Overall, these four steps are essential for performing Image Classification with TensorFlow and Keras frameworks.

3.1. Gathering the Data

To analyze the deep learning neural network a couple of data sets were identified and worked on to train the neural network. The data was gathered from Kaggle which is a popular online community platform for data science and machine learning enthusiasts. It was founded in 2010 and was later acquired by Google in 2017. Kaggle provides a wide range of resources for data scientists, including access to datasets, cloud-based computational resources, and a community of over 4 million members who share their ideas, collaborate on projects, and

participate in competitions. After extracting the data from Kaggle the datasets were altered according to the requirement of the study. Following datasets were incorporated for the analysis of the neural network:

- GTSRB - German Traffic Sign Recognition Benchmark: The German Traffic Sign Benchmark is a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN) 2011. The benchmark has the following properties:

  - Single-image, multi-class classification problem

  - More than 40 classes

  - More than 50,000 images in total

  - Large, lifelike database

- Chinese Traffic Signs: This dataset is originating from Chinese Traffic Sign Recognition Database. It has been explored by Riga Data Science Club members to do some training on convolution neural networks. Dataset consists of 5998 traffic sign images of 58 categories. Each image is a zoomed view of a single traffic sign. Annotations provide image properties (file_name, width, height) as well as traffic sign coordinates within image and category.

3.2. Preparing the Data

After the two datasets, i.e., German Traffic Sign Dataset and The Chinese Traffic Sign dataset, were gathered from Kaggle, the data were pre-processed before training the neural network. The training dataset was split into two different folders, one for training and the other for validation. We used python scripts to create the data sets for training and validation of the neural network. The script runs through the training folder, and it splits it in ratio of 9:1 and puts

them in a newly formed training folder and the validation folder respectively. For the test dataset, a python script was used to map the images to the labels in the csv file and then put them into the folder of corresponding classes to make the test data look representative.

After the data was rearranged in the respective folders, a preprocessor was defined to process the data before we feed it to the deep learning model. In the pre-processor we rescale the images. This is to normalize the pixel values to a specific range. For 8-bit images, we generally rescale by 1/255 to have pixel values in the range 0 and 1.

After the pre-processor is defined, data generator utility function was defined to generate processed data for training, validation, and testing. For the data generators, class mode 'categorical' was chosen as we are using categorical cross entropy loss function while compiling the model, the target size of 60*60 was chosen as we wanted all our images resized to 60*60 pixels, and the color mode is set to be 'rgb' because the images are colored images. The shuffle parameter is set to 'true' for the training and validation datasets as images will be shuffled so that between the two epochs the order of the images will not be the same. This kind of randomness helps the model be more generalized, learn features and ignore the order between the images as we don't want our model to learn that.

3.3. Building the Model

The deep learning model is constructed through a functional method using TensorFlow and Keras. This approach offers great flexibility in designing the structure of the neural network. This allows the creation of complex models with multiple inputs and outputs, as well as models with shared layers or those featuring multiple inputs or outputs. Furthermore, it enables models that can be quickly reused and modified. By delineating the network structure as a graph of layers, it is possible to create an reusable template for network architecture that can be tailored

for various applications or needs. Doing this helps save time and effort when developing new models or adapting existing ones for new tasks.

This model was constructed with 15 hidden layers and an output layer. The hidden layer consists of Conv2D, MaxPool2D, BatchNormalization, GlobalAvg2D and Dense as sublayers.

The Conv2D layer performs a mathematical operation called convolution, which applies a set of filters to an input image in order to extract features. During training, these weights are adjusted for optimal performance on specific tasks like image classification or object detection. After Conv2D, ReLU is introduced as a nonlinear activation function which introduces non-linearity into the model and allows it to learn more complex features.

MaxPool2D is an algorithm used to reduce the spatial dimensions of input feature maps by down sampling them. After applying Conv2D, this layer further shrinks the output feature maps' spatial dimensions, making subsequent layers more computationally efficient. MaxPool2D may be applied multiple times in a neural network for even further reduction in feature map dimensions.

The GlobalAvg2D layer serves as a replacement for the fully connected layers at the end of a CNN. Instead of flattening out the feature map and passing it through multiple dense layers, GlobalAvg2D pools the feature map down into one-dimensional vector, which has several benefits including reducing model parameters while encouraging it to focus on crucial features.

The Dense layer, also referred to as the fully connected layer, is an essential building block in deep learning for image classification. It typically serves as the final layer of a neural network and produces its output. The Dense layer takes as its input a set of features extracted from the input image by previous layers in the network and applies a linear transformation to them. This transformation is weighted sum of input features, with each weight representing one

neuron in this layer. The weights in the Dense layer are learned during training through backpropagation, which adjusts them to minimize the loss function. After passing through this nonlinear activation function, which introduces non-linearity into the network and allows it to model complex patterns in data, the output of this layer becomes the Dense layer weights.

Table 1 and Table 2 provide detailed models, outputs and weights for German Traffic Sign and Chinese Traffic Sign, respectively.

Table 1: German Traffic Sign Dataset - Description of Model, the Outputs, and their Weights

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 60, 60, 3)]	0
conv2d	(None, 58, 58, 32)	896
max_pooling2d	(None, 29, 29, 32)	0
batch_normalization	(None, 29, 29, 32)	128
conv2d_1 (Conv2D)	(None, 27, 27, 64)	18496
max_pooling2d_1	(None, 13, 13, 64)	0
batch_normalization_1	(None, 13, 13, 64)	256
conv2d_2	(None, 11, 11, 128)	73856
max_pooling2d_2	(None, 5, 5, 128)	0
batch_normalization_2	(None, 5, 5, 128)	512
conv2d_3	(None, 3, 3, 256)	295168
max_pooling2d_3	(None, 1, 1, 256)	0
batch_normalization_3	(None, 1, 1, 256)	1024
global_average_pooling2d	(None, 256)	0
dense	(None, 128)	32896
dense_1	(None, 43)	5547
Total Params	428779	
Trainable Params	427819	
Non-Trainable Params	960	

Table 2: Chinese Traffic Sign Dataset - Description of Model, the Outputs, and their Weights

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 60, 60, 3)]	0
conv2d	(None, 58, 58, 32)	896
max_pooling2d	(None, 29, 29, 32)	0
batch_normalization	(None, 29, 29, 32)	128
conv2d_1 (Conv2D)	(None, 27, 27, 64)	18496
max_pooling2d_1	(None, 13, 13, 64)	0
batch_normalization_1	(None, 13, 13, 64)	256
conv2d_2	(None, 11, 11, 128)	73856
max_pooling2d_2	(None, 5, 5, 128)	0
batch_normalization_2	(None, 5, 5, 128)	512
conv2d_3	(None, 3, 3, 256)	295168
max_pooling2d_3	(None, 1, 1, 256)	0
batch_normalization_3	(None, 1, 1, 256)	1024
global_average_pooling2d	(None, 256)	0
dense	(None, 128)	32896
dense_1	(None, 58)	7482
Total Params	430714	
Trainable Params	429754	
Non-Trainable Params	960	

3.4. Training the Model

Once your model architecture is defined, use Keras' 'compile()' function to compile it. This involves specifying a loss function, optimizer and metrics for evaluation during training. The loss function measures how well the model performed on training data while the optimizer adjusts weights in the model to minimize loss function impacts; finally, metrics evaluate model performance on validation and testing sets.

We used Adam as the optimizer for training our model because it is an effective optimization algorithm for deep neural networks. Adam stands for Adaptive Moment Estimation and it is a type of stochastic gradient descent optimization algorithm. Adam effectively utilizes

two well-known optimization algorithms: AdaGrad and RMSProp. AdaGrad adjusts the learning rate based on each parameter, while RMSProp adjusts rates based on a moving average of the squared gradient. Adam utilizes both methods and adds bias correction to calculate adaptive learning rates. This makes him a highly efficient and effective optimizer for many deep learning tasks. Furthermore, Adam has several hyperparameters that can be adjusted individually in order to fine-tune its performance on any particular task.

The loss function used is 'categorical_crossentropy', as it has the capacity to handle multi-class classification, penalize incorrect predictions, and boast reliability and efficiency.

Accuracy is a useful metric when training the model, as it measures how accurately it classifies images correctly. Accuracy provides insight into how well-trained your model is at performing this task.

After compiling the model, the next step is to train it using Keras' 'fit()' function. This involves passing both training and validation data to the model along with specifying its number of epochs and batch size for training. During this process, the model updates its weights in order to minimize its loss function using backpropagation and stochastic gradient descent techniques.

Both German Traffic Sign and Chinese Traffic Sign Datasets were run for 10 epochs with a batch size of 32. At each epoch, Model Checkpoints were created which monitored validation accuracy; the model with maximum validation accuracy was saved as Tables 3 and 4. Table 3 and Table 4 illustrate how loss, accuracy, validation loss, and validation accuracy change after each epoch.

Table 3: German Traffic Sign Dataset – Training Logs

Epoch	Loss	Accuracy	Val. Loss	Val. Accuracy
1	1.1425	0.7205	0.3199	0.9342
2	0.1925	0.9637	0.1133	0.9768
3	0.0746	0.9891	0.0694	0.9852
4	0.0400	0.9955	0.0481	0.9895
5	0.0242	0.9982	0.0398	0.9918
6	0.0169	0.9984	0.0325	0.9941
7	0.0124	0.9994	0.0305	0.9941
8	0.0095	0.9993	0.0282	0.9941
9	0.0077	0.9996	0.0250	0.9954
10	0.0062	0.9998	0.0252	0.9946

Table 4: Chinese Traffic Sign Dataset – Training Logs

Epoch	Loss	Accuracy	Val. Loss	Val. Accuracy
1	2.4286	0.4668	4.0417	0.0295
2	1.0199	0.8000	3.8688	0.0750
3	0.5435	0.9172	2.6663	0.3205
4	0.3305	0.9579	1.1218	0.8205
5	0.2053	0.9788	0.4277	0.9432
6	0.1372	0.9885	0.2187	0.9750
7	0.0952	0.9946	0.1313	0.9886
8	0.0677	0.9976	0.0919	1.0000
9	0.0505	0.9992	0.0715	1.0000
10	0.0401	0.9992	0.0592	1.0000

3.5. Evaluating the Model

Once trained, we can evaluate on test datasets using the 'evaluate()' function of Keras API in TensorFlow. It takes both the trained model and test dataset as inputs and computes performance metrics for the model on that particular test set. The returned values include accuracy, precision, recall, F1 score and more metric types that were specified during model compilation. These value can then be used to track model progress over time.

4. RESULTS

The neural network was designed using the layers provided by the Keras libarary. The training data was run for 10 epochs with a batch size of 32 and after each epoch a validation dataset was run to test the accuracy of the neural network. The training dataset and validation dataset were re-shuffled for each epoch.

The graphs shown from Figure 11 and Figure 16 showcases how after each epoch model accuracy, loss, validation accuracy, and validation loss are getting affected for the German Traffic Sign Dataset whereas Figure 17 to Figure 22 showcases the same for the Chinese Traffic Sign Dataset. During each epoch four parameters are evaluated:

- Accuracy
- Loss
- Validation Accuracy
- Validation Loss

Val. accuracy of 0.9954 and val. loss of 0.0250 was identified while training the Model over German Traffic Sign Dataset, whereas in case of Chinese Traffic Sign, val. accuracy of 1.0000 and val. loss 0.0592 was noted.

An accuracy of 0.9495 and loss of 0.1626 was observed on the test dataset of German Traffic sign which is a good result for the developed deep learning neural network model, whereas in case of the Chinese Traffic Sign test dataset where we had small dataset for the training and evaluation, we detected an accuracy of 0.6038 and loss of 1.5379. The results demonstrate how of quality and size of the test data can impact the accuracy of the model. A small test dataset, like in case of Chinese Traffic Sign Dataset, may not provide enough diversity

in the images to allow the model to generalize well. This means that the model may be overfitting to the training data and not able to perform well on new, unseen images.

4.1. German Traffic Sign Plots

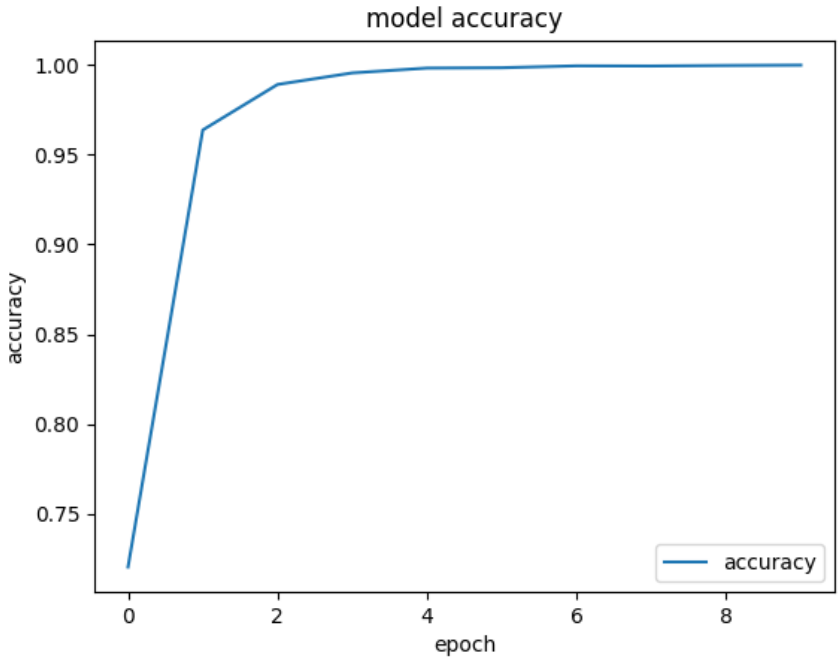

Figure 10: Accuracy v/s Epoch– German Traffic Sign Dataset

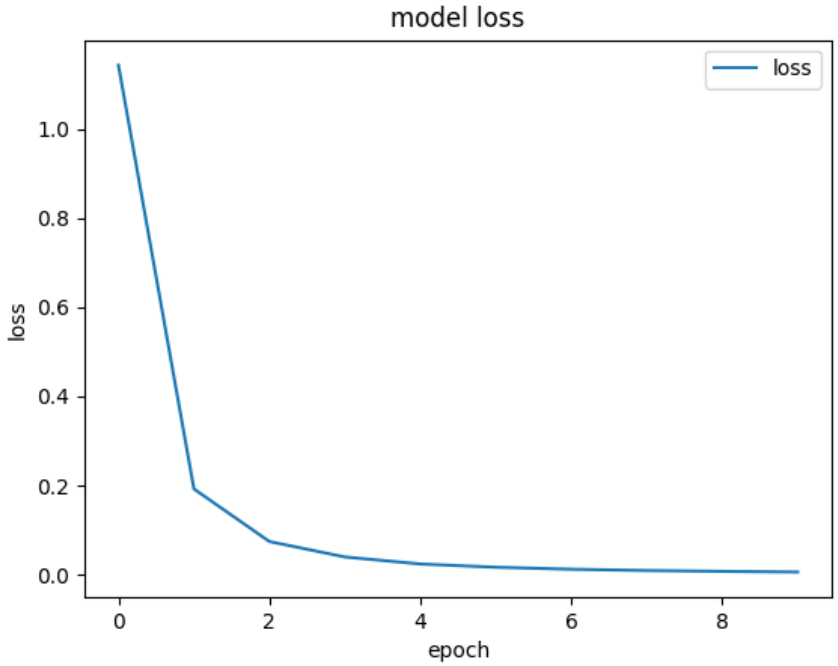

Figure 11: Loss v/s Epoch– German Traffic Sign Dataset

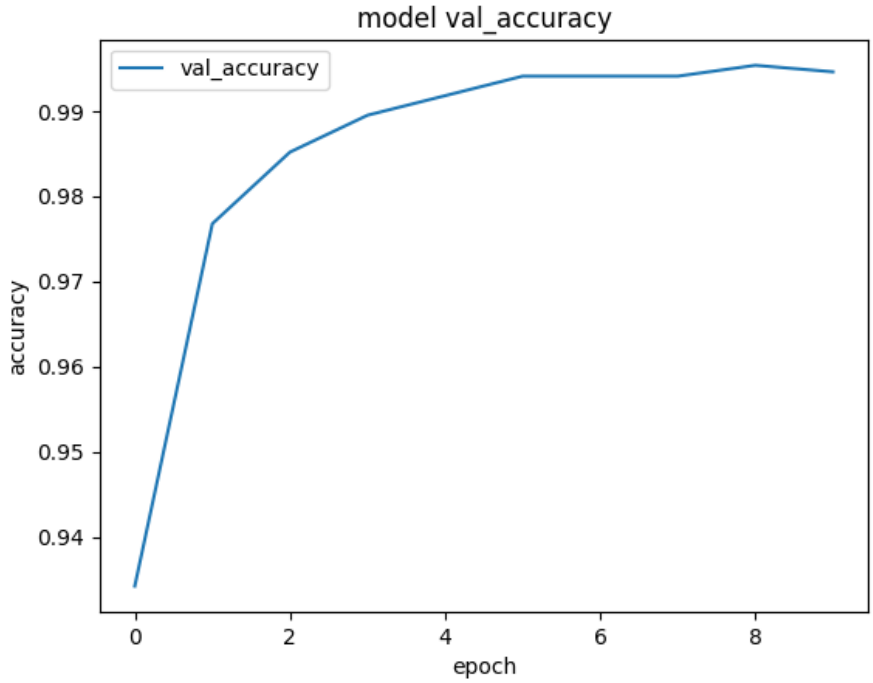

Figure 12: Validation Accuracy v/s Epoch– German Traffic Sign Dataset

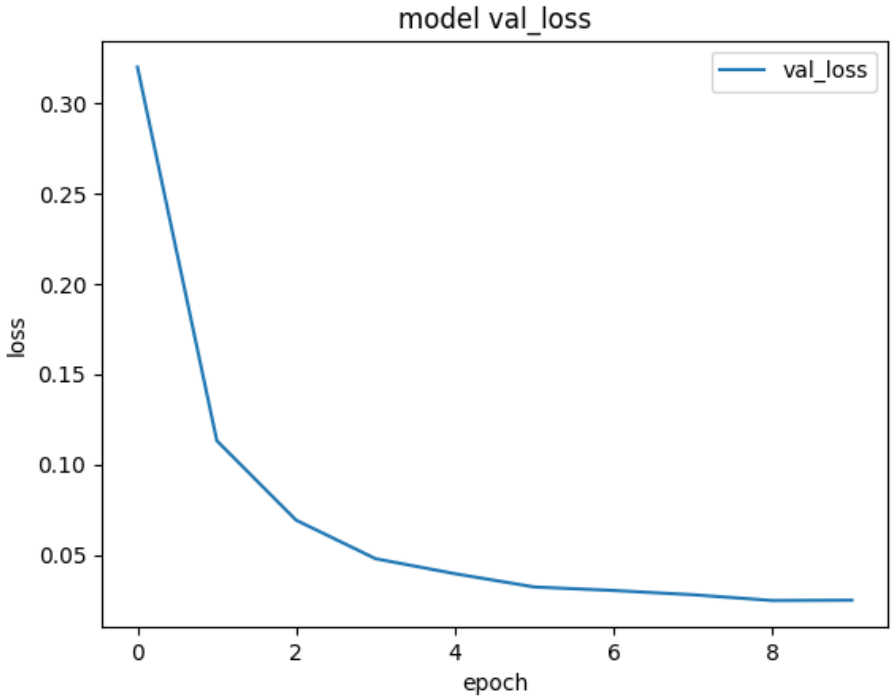

Figure 13: Validation Loss v/s Epoch– Chinese Traffic Sign Dataset

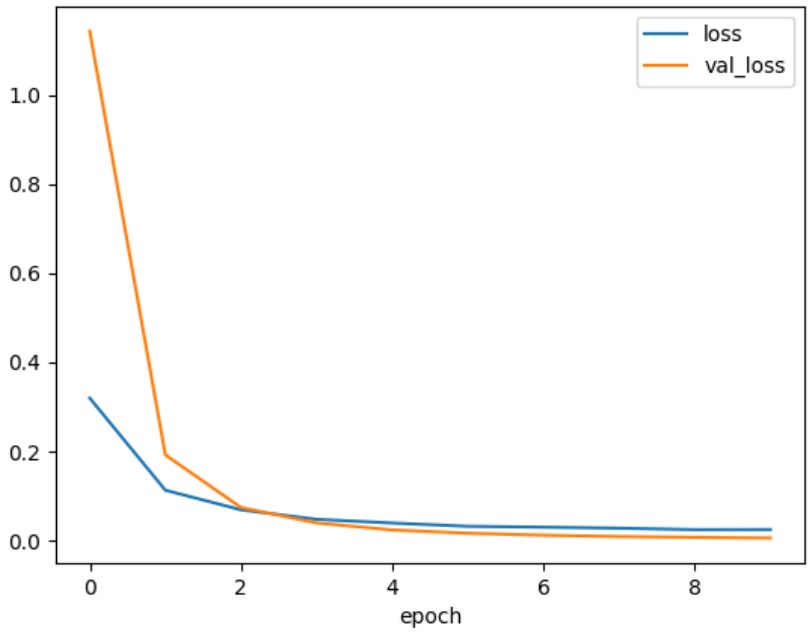

Figure 14: Loss and Validation Loss Comparison v/s Epoch– German Traffic Sign Dataset

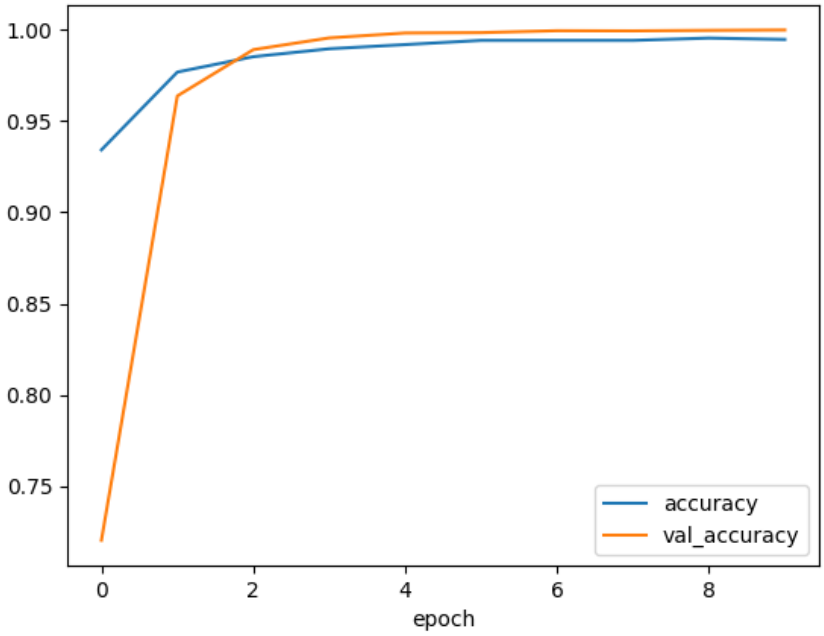

Figure 15: Accuracy and Validation Accuracy Comparison v/s Epoch– German Traffic Sign Dataset

4.2. Chinese Traffic Sign Plots

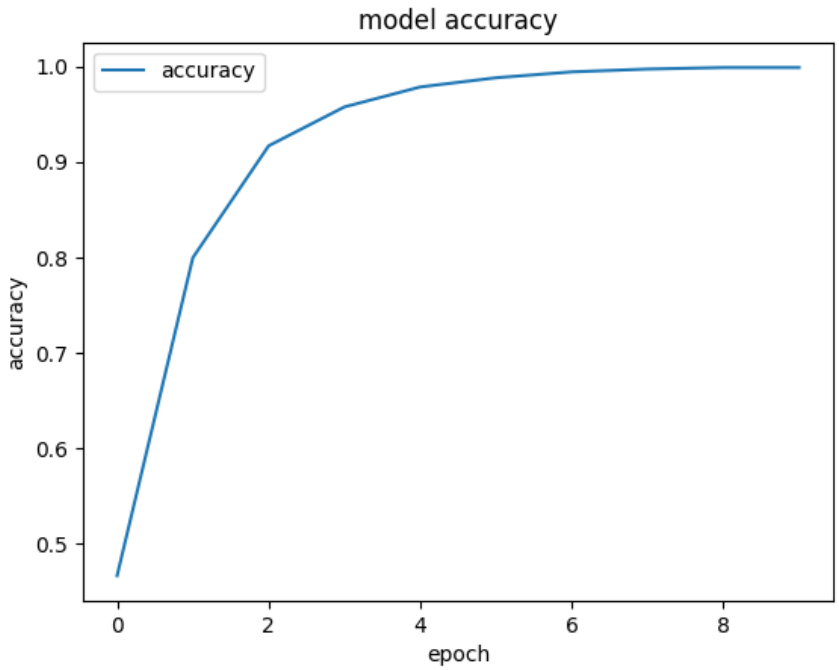

Figure 16: Accuracy v/s Epoch – Chinese Traffic Sign Dataset

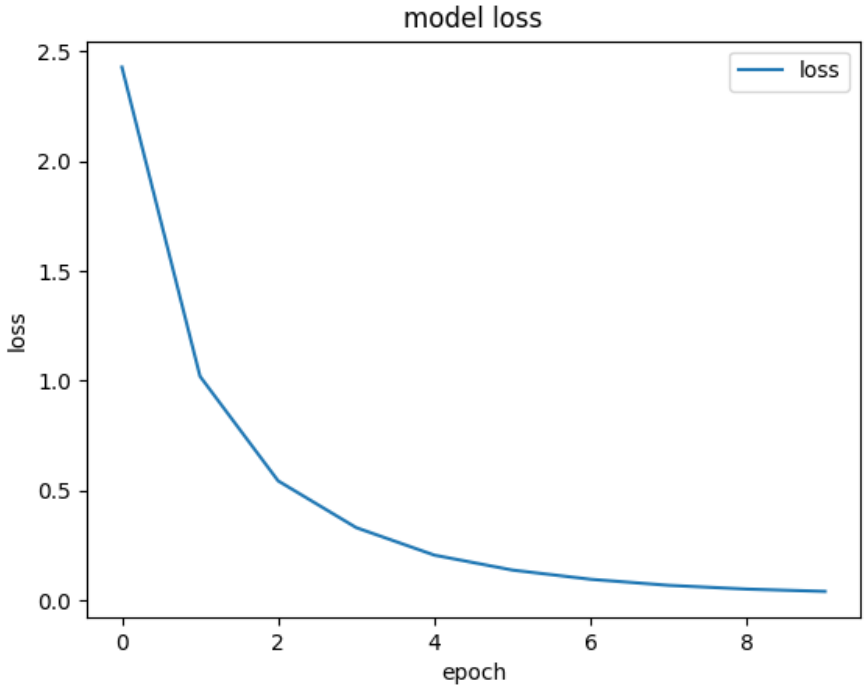

Figure 17: Loss v/s Epoch – Chinese Traffic Sign Dataset

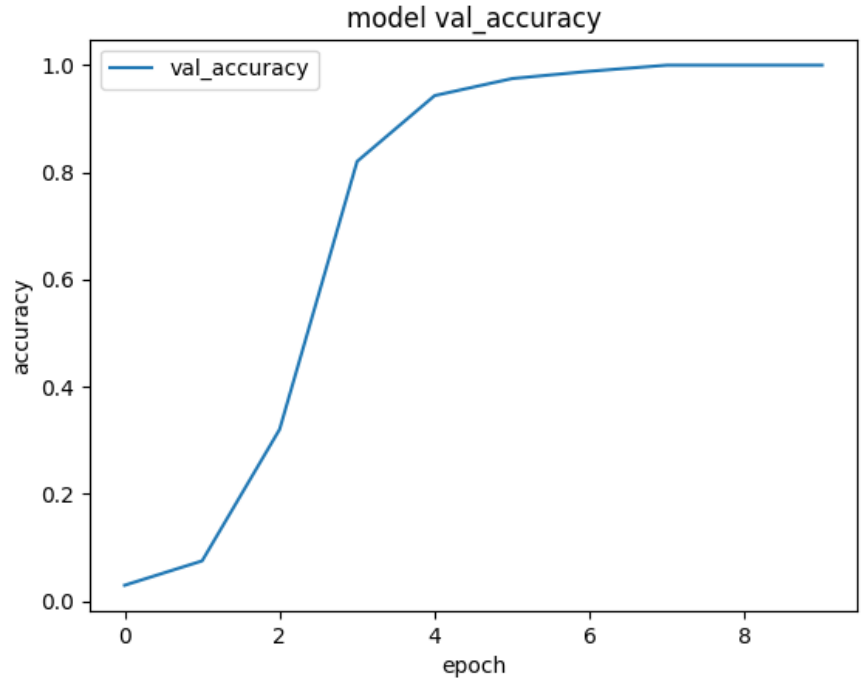

Figure 18: Validation Accuracy v/s Epoch – Chinese Traffic Sign Dataset

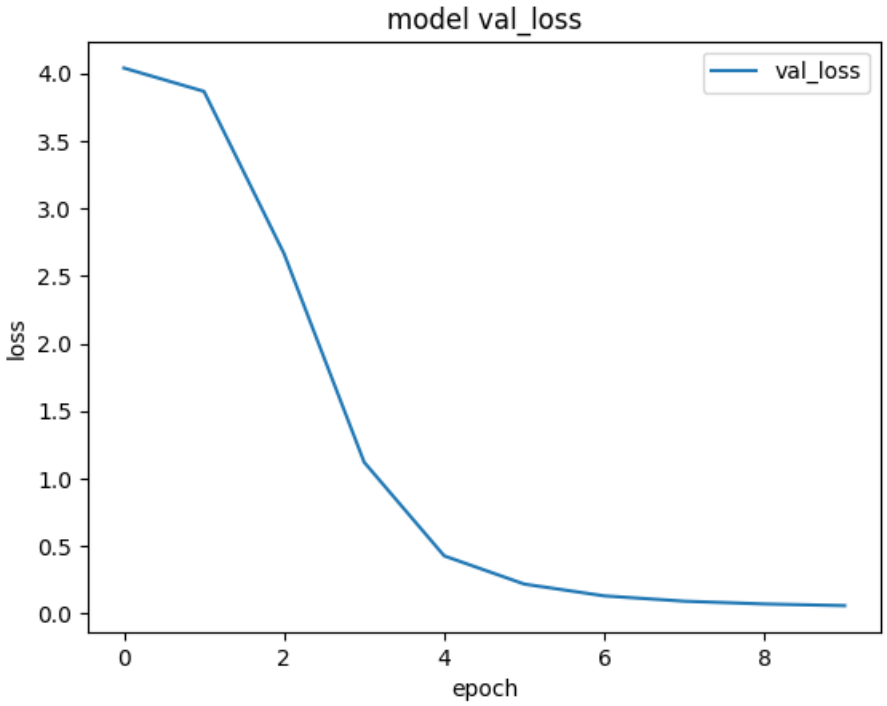

Figure 19: Validation Loss v/s Epoch – Chinese Traffic Sign Dataset

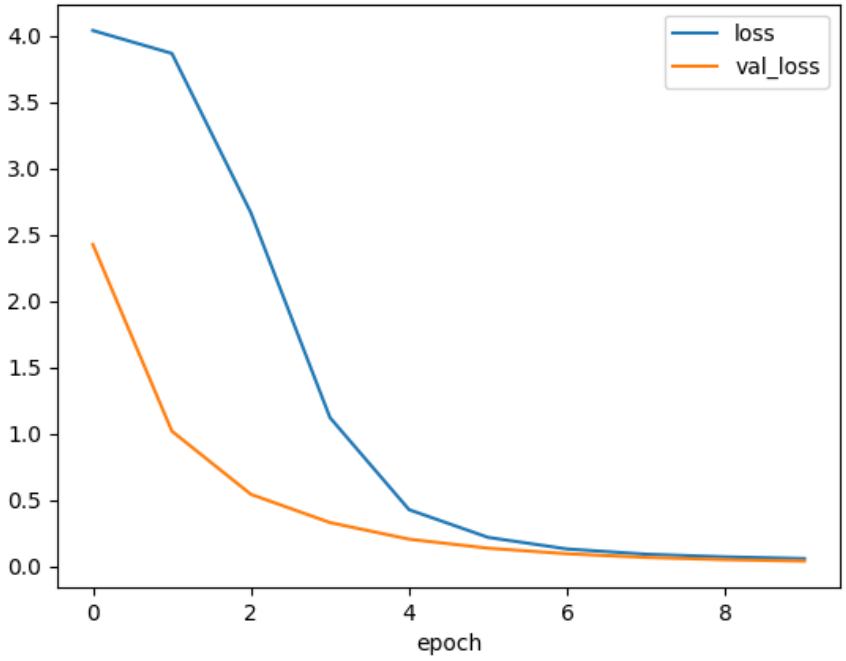

Figure 20: Loss and Validation Loss Comparison v/s Epoch– Chinese Traffic Sign Dataset

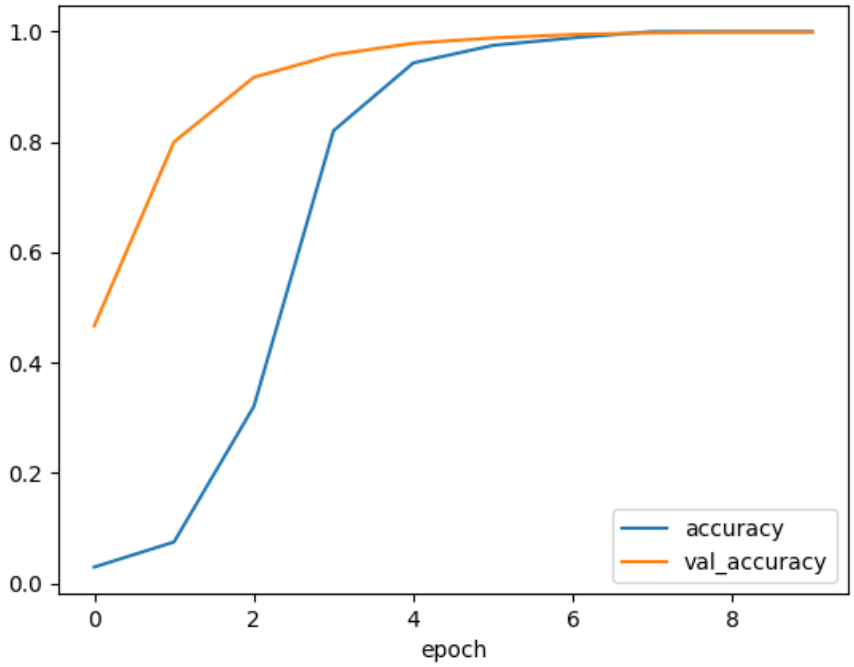

Figure 21: Accuracy and Validation Accuracy Comparison v/s Epoch – Chinese Traffic Sign Dataset

5. CONCLUSION

In this study, we focused on CNN as the primary image classification technology. We specifically examined the role of epochs, dataset size and hidden layer count in CNN to ensure accuracy and avoid overfitting issues. By applying deep learning with Tensor Flow framework, we achieved positive outcomes - it could train, classify, and predict with high validation accuracy rates up to 100%.

Our findings highlight the power of DNNs for image classification tasks and emphasize the significance of carefully tuning hyperparameters like epochs, batch size, activation function and number of hidden layers to optimize performance. Furthermore, our use of Tensor Flow underscores its effectiveness in implementing deep learning models for image classification tasks. Future work could explore application of DNNs and Tensor Flow beyond image classification tasks to other domains and assess the influence other hyperparameters have on model performance. Ultimately this study contributes to further development in deep learning techniques used for image classification by providing a framework to explore further research in this field.

5.1. Discussion

This study evaluated the feasibility of using deep neural networks for image classification tasks. Results demonstrated that this model performs well in terms of accuracy and speed when size and quality of dataset are taken into consideration. Nonetheless, further research and improvement are needed in this area.

One potential future direction is to investigate the effects of various pre-processing techniques on model performance. While this study utilized basic pre-processing methods like

resizing and data augmentation, more sophisticated ones like contrast enhancement or image segmentation may further enhance the model's accuracy.

Another potential research direction could be to investigate the application of transfer learning for image classification tasks. This method involves using an existing model trained on a large dataset as the starting point for new classification tasks, potentially decreasing training data requirements and improving model performance.

Furthermore, this study used only two datasets to train and validate the model. To evaluate its generalizability and robustness, additional tests with different degrees of complexity would be beneficial.

Overall, this study provides a solid foundation for future research in image classification using deep neural networks.

REFERENCES

[1] Kembuan, Olivia, et al. “Convolutional Neural Network (CNN) for Image Classification of Indonesia Sign Language Using Tensorflow.” 2020 2nd International Conference on Cybernetics and Intelligent System (ICORIS), 2020, https://doi.org/10.1109/icoris50180.2020.9320810.

[2] Kothari, Jubin Dipakkumar, “A Case Study of Image Classification Based on Deep Learning Using Tensorflow (2018).” International Journal of Innovative Research in Computer and Communication Engineering, 6(7), 3888-3892., Available at SSRN: https://ssrn.com/abstract=3729757

[3] Arora, Shefali, and Bhatia, Manvir. “Handwriting Recognition Using Deep Learning in Keras.” 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2018, https://doi.org/10.1109/icacccn.2018.8748540.

[4] Treebupachatsakul, Treesukon, and Poomrittigul, Suvit. “Bacteria Classification Using Image Processing and Deep Learning.” 2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), 2019, https://doi.org/10.1109/itc-cscc.2019.8793320.

[5] Planche, Benjamin, and Andres, Eliot. Hands-on Computer Vision With TensorFlow 2: Leverage Deep Learning to Create Powerful Image Processing Apps With TensorFlow 2.0 and Keras. 2019.

[6] Sanghvi, Kavish. “Image Classification Techniques - Analytics Vidhya - Medium.” Medium, 20 Mar. 2022, medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac.

[7] Abadi, Martín, et al. “TensorFlow: A System for Large-scale Machine Learning.” Operating Systems Design and Implementation, 2016, pp. 265–83. https://doi.org/10.5555/3026877.3026899.

[8] Carnes, Beau. “TensorFlow for Computer Vision – Full Course on Python for Machine Learning.” freeCodeCamp.org, Oct. 2021, www.freecodecamp.org/news/how-to-use-tensorflow-for-computer-vision.

[9] Bag, Sukanya. “Activation Functions — All You Need to Know! - Analytics Vidhya - Medium.” Medium, 2 Jan. 2023, medium.com/analytics-vidhya/activation-functions-all-you-need-to-know-355a850d025e.

[10] Kaggle. “GTSRB - German Traffic Sign Recognition Benchmark.” Kaggle, 25 Nov. 2018, www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign.

[11] Kaggle. “Chinese Traffic Signs.” Kaggle, 16 Apr. 2020, www.kaggle.com/datasets/dmitryyemelyanov/chinese-traffic-signs.

[12] GeeksforGeeks. “Artificial Neural Networks and Its Applications.” GeeksforGeeks, Apr. 2023, www.geeksforgeeks.org/artificial-neural-networks-and-its-applications.

[13] DeepAI. “Data Science and AI Glossary.” DeepAI, deepai.org/definitions.

[14] StackOverflow. “Keras - Plot Training, Validation and Test Set Accuracy.” Stack Overflow, stackoverflow.com/questions/41908379/keras-plot-training-validation-and-test-set-%20%20accuracy.

[15] Lokeshsrikakolapu. “Traffic Sign.” Kaggle, Sept. 2022, www.kaggle.com/code/lokeshsrikakolapu/traffic-sign.

[16] Hossamfakher. “Traffic Sign Classification Acc 96 %.” Kaggle, Feb. 2022, www.kaggle.com/code/hossamfakher/traffic-sign-classification-acc-96.