Sign language recognition based on CNN with optimized activation function

Wen Han Lee
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
TP070146@mail.apu.edu.my

Jia Ling Tan
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
TP068255@mail.apu.edu.my

Zailan Arabee Abdul Salam
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
zailan@apu.edu.my

Huey Ying Teoh
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
TP069734@mail.apu.edu.my

Qin Jun Lee
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
TP070170@mail.apu.edu.my

Lai Tzi Syuen Suzanne
School of Computing
Asia Pacific University of Technology & Innovation (APU)
Kuala Lumpur, Malaysia
TP068713@mail.apu.edu.my

***Abstract*—Sign language is a non-verbal language that is used mostly by individuals with hearing impairments, yet it is not a common language among people without hearing disabilities. A Convolutional Neural Network (CNN) model to recognize sign language is developed to overcome the barrier of communication between individuals. In this research, the model is trained with altered first, second, and third dense layers, their activation functions are changed from ReLU to other activation functions to find out the best activation function to create the most accurate model. The new activation function chosen in our research includes Sigmoid, Tanh, Softmax, Softplus, ThresholdedReLU, ELU, and PReLU. The comparison of the accuracy of models trained with different activation functions is provided.**

Keywords— Convolutional Neural Network (CNN), Activation Function, Sign Language Recognition Introduction

I. INTRODUCTION

In our increasingly interconnected world, communication is one of the most crucial aspects of human interactions. However, in many cases, verbal communication is not possible. This is especially true for people with speaking and or hearing disabilities. Typically, the members of the aforementioned communities communicate using sign language. This proposes a problem—what if their conversation partner does not speak sign language? What if a deaf person had to communicate to their colleagues in a live video conference meeting? A system is sorely needed to bridge this communication gap and eliminate the problem.

The purpose of this project is to make a sign language recognition model to help recognize the gestures made by an individual by using a convolutional neural network (CNN). The CNN architecture is made up of multiple layers like convolutional layers, max pooling layers, dense layers, and optimization functions. All of these layers work together to take in an image to eventually reduce it to it barest key features to label them according to pre-defined classes that were created based off of the dataset. In our paper, this concept is essentially applied to the fine-tuning of a sign language recognition system that detects sign language gestures in real-time on a live camera feed to convert them into written text.

There are several factors that affect how accurate the results of a sign language recognition system are—from the diversity in the dataset to the layers that compose the CNN architecture, down to the smallest of alterations in each individual layer of the CNN architecture. As various pieces of journals available already document their varying levels of success at raising the accuracy rate on their own models, we acknowledge that in most of these research papers, ReLu is the most used activation function.

Activation functions are an integral component to CNN, as these functions are responsible for introducing non-linearity into the model while normal layers only add linear functions. Non-linearity is essential to capturing complex relationships in a model that cannot simply be represented by straight lines and planes—and in the real world, a lot of relationships to be represented are not a straightforward as the ideal. Different types of activation functions are useful in different aspects: all are used to make decisions (deciding where an input belongs), some are particularly good at squashing inputs into a specific range to represent a probability, and newer activity functions like ReLu can introduce sparsity to the model. Sparsity is conducive to the optimization of a network since it produces an output of zero for any negative input, effectively leaving out the features in an input that are not essential to the processing. In short, activation functions are important because they shape the output of a neuron, impact the training process, and assist the network's ability to handle unseen data.

Our research will focus our efforts on determining the "best" activation functions for the CNN architecture for a sign language recognition system. This will be done by substituting the ReLu function in the dense layers of our CNN into various other activation functions that will be put to the test: Sigmoid, Tanh, Softmax, Softplus, ThresholdedReLU, ELU, and PReLU. The evaluation of each model's

performance will be based on their accuracy in predicting the sign languages using the test data from our dataset.

II. Literature Review

A. Similar works

Define a There is an abundance of prior research conducted on the topic of producing sign language recognition systems using Convolutional Neural Networks (CNN). These projects were carried out with varying methodologies as well as datasets that comprise of different signs to be identified and learned by the system—ranging from finger-spelled digits to American Sign Language (ASL) alphabets and simple ASL phrases.

In a study published in the Journal of Jilin University (Khan, et al., 2022), the researchers evaluated the performance on their model which used CNN to identify images based on depth and intensity data. This was done by calculating the weighted average of the backgrounds in each frame and image to identify the foreground objects when necessary, hence allowing the system to also recognize the threshold and contours of the foreground objects when they appear. The pre-processed images were then fed into the CNN which is made up of ReLu, the activation function, and the max pooling operation was used to retrieve the most prominent features from the map covered by the filter. A caption would be generated for the image from pre-determined classifications in the CNN architecture. This model achieved an outstanding 98.76% accuracy rate in its performance and accuracy demonstration, which marks it as a particularly efficient method at accomplishing the goal of the study.

Dwijayanti et al. (2021) detailed a different approach to recognizing sign language with CNN. This particular study aims to better bridge the communication gap between the deaf and non-deaf people by producing a system that fluently recognizes alphabets in the Indonesia Sign Language (BISINDO), which is more commonly used in daily communication efforts as compared to Indonesia Sign Language System (SIBI), the more formal alternative. The dataset that was fed into the CNN in this study consisted of 39455 data points split following the composition of 60% for training, 20% for validation, and the last 20% for testing. The dataset taken also comprised of a large variety of signs: 26 alphabets and digits 1-10 adhering to BISINDO standards. These images were standardized by solely using a LogiTech webcam which records in a resolution of 1080p and 30 frames per second, which is considered high resolution by industry standards. This standardization was crucial in creating a model that is more fairly evaluated on its performance since it was proven in a 2021 JATI paper that noise affects the accuracy of CNN image classification (Lau et al., 2021). The dataset was also inclusive such that each alphabet was taken under conditions of varying lighting intensity as well as first and second-person perspectives. Model A of this system recorded the highest accuracy rate among the three tested models, which stands at 98.7%.

This 2018 conference paper (Chauhan, Ghanshala, & Joshi, 2018) explored a sign-recognition system using CNN, the deep-learning algorithm which was applied on the well-known MNIST and CIFAR-10 datasets. The models in this study explored both the system's accuracy in image recognition as well as object recognition. The MNIST dataset is applied specifically in systems that are trained to recognize handwritten digits. The standardization of the dataset ensures that each image is 28x28 pixels. Out of the 70000 images in the dataset, 60000 was used for training purposes while the remainder is utilized in the testing phase for this system. CIFAR-10 is a dataset specialized for object detection systems. Its images are 32x32 pixels and are all in color. Contained within this dataset is 60000 images for each class that it represents (airplane, bird, cat, automobile, deer, dog, frog, horse, truck and ship) and 50000 of these images were reserved for training while the remaining 10000 was used for testing. The CNN architecture for this study differs from the model used on the MNIST dataset and the CIFAR-10 dataset. Changes in the CNN architecture can lead to big disparities in results, as seen in a study conducted by Seng et al. (2017), documented in a JATI paper. For MNIST, the first two convolutional layers would pass the images through the ReLu activation function, max pooling layer, and then droupout, which is a technique used to prevent overfitting. The third convolutional layer consists of an added fully_connected layer that allows the model to learn all key features identified in previous layers before dropout. In the case of the CNN used on CIFAR-10, Conv1 uses a 32 filters (5x5) size with a stride of 1. When this output is passed through ReLu, all negative values are replaced with zero. Conv2 convolves the image following (62,5x5, stride = 1) before passing through ReLu and a max-pooling layer that moves 2 pixels at a time. Dropout is used to regularize the output. Conv3 uses a configuration of (64,3X3, stride=1) before passing the output through the ReLu function. The output of Conv4(64,3x3, stride=1) is passed through both ReLu and droupout. Finally, Conv5(64, 3x3, stride=1) produces an output that passes through the sequence of these layers: ReLu -> Max_pool -> dropout -> Flatten -> Dense(512) -> ReLu -> dropout -> Dense -> softmax. The CN model of MNIST achieved a superior accuracy rate of 99.6%, while CIFAR-10's model only achieved 80.17% on the test set.

B. Methodology

The dataset used in this paper was created specifically for this research. Using a webcam, 340 images were taken for each number ranging from 0-9. The whole dataset comprised of a total of 3400 images, which were split into 3000 for the training phase and 400 for the testing phase. The purpose of creating the training dataset is that we are able to establish a set of images that can be fed into our CNN architecture for the system to learn what each digit looks like, hence gaining to ability to successfully label them with high accuracy. The test data is a set of data that is not revealed to the system until it has finished the training phase. The testing dataset evaluates the model's ability to label each digit in the static images accurately.

The purpose of our research is to determine which activation function, upon being integrated into the CNN, can build the model that performs best in labelling the signed digits in real-time. Convolutional neural networks generally operate by taking input images fed to them (through datasets) and eventually convolve them with activation functions and filters to extract the key features in each frame and use shared weights and biases to produce a post-processed image that they can label according to predefined classes.

The general architecture of our CNN will be made up of three 2D convolutional layers that are combined with max pooling layers to be flattened and added with multiple fully-

connected layers. The first convolutional layer will be applied with 32 filters of 3x3 size for an input image of 64x64 pixel dimensions with 3 RGB channels. The output is run through a max_pooling layer applied on each 2x2pixel region which takes 2 strides at a time. The second convolutional layer applies 64 filters of the same size, the output of which is run through another max_pooling layer that bears the same specifications as the last. The third convolutional layer is applied with 128 filters, each of 3x3 size, and is followed by yet another max_pooling layer.

The same dataset will be fed into each CNN architecture which varies from one another by their activation functions in the dense layer. The activation functions subbed into the place of "ReLu", the original tested activation function, are Sigmoid, Tanh, Softmax, Softplus, ThresholdedReLU, ELU, and PReLU.

C. Conclusion

The CNN is widely used in many pieces of research pre-dating ours, but different CNN architecture produces results of varying success. With the many choices of activation functions available, it is difficult to pinpoint which develops the ideal CNN architecture that can eventually be adopted into sign language recognition systems that can be applied to all sign language systems. Hence, our research is crucial to identifying the activation function that makes up the best CNN architecture for similar projects.

III. MATERIALS AND METHODS

The project code used in this project is from DataFlair (n.d.).

A. Data

Fig. 1: Example of the data

The database used in both training and testing is created by us using the help of cv2 and numpy library. The codes for the creation of data are run in Visual Studio Code and the data is saved into files on our laptop. We created 300 images for each number for the training and 40 each for the testing, the images are saved in respective folders to be further utilized. The images created are in black and white so the machine can understand the gesture.

B. Preparation

```
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Activation, Dense, Flatten, BatchNormalization, Conv2D, MaxPool2D, Dropout
from keras.optimizers import Adam, SGD
from keras.metrics import categorical_crossentropy
from keras.preprocessing.image import ImageDataGenerator
import itertools
import random
import warnings
import numpy as np
import cv2
from keras.callbacks import ReduceLROnPlateau
from keras.callbacks import ModelCheckpoint, EarlyStopping
warnings.simplefilter(action='ignore', category=FutureWarning)
import matplotlib.pyplot as plt
```

Fig. 2. Library needed for the training of model

Before we start training the model, some preparation needs to be done. Fig.2 above shows the libraries that need to be imported before training.

After that, we import the database created earlier and process it using the Keras library for it to fit in our training. The function ImageDataGenerator() and flow_from_directory()is used in our preprocessing, the class_mode is 'categorical' since we are using the model to classify gestures into 10 different classes and the image size was set to be 64*64.

C. Training

```
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='ReLU', input_shape=(64,64,3)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='ReLU', padding = 'same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='ReLU', padding = 'valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Flatten())

model.add(Dense(64,activation ="ReLU"))
model.add(Dense(128,activation ="ReLU"))
#model.add(Dropout(0.2))
model.add(Dense(128,activation ="ReLU"))
#model.add(Dropout(0.3))
model.add(Dense(10,activation ="softmax"))
```

Fig. 3. Layers of the network

Convolutional Neural Network (CNN) will be used to train the model, the layers implemented include Conv2D layers, max pooling layers, flatten layer and dense layer. The activation function for Conv2D layers and dense layers are all set to be ReLU except for the last dense layer. Softmax is used in the last layer because it is used for classification.

```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0001)
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0, mode='auto')



model.compile(optimizer=SGD(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=1, min_lr=0.0005)
early_stop = EarlyStopping(monitor='val_loss', min_delta=0, patience=2, verbose=0, mode='auto')
```

Fig. 4: Model compilation

The model is compiled using two optimizer which is Adam and SGD, and the minimum learning rate for early stopping is set to be 0.0001 and 0.0005 respectively.

D. Evaluation

{'loss': [1.3327414989471436, 0.06851650774478912, 0.011676767840981483, 0.004266440402716398, 0.002386561129421818], 'accuracy': [0.6930000185966492, 0.9813333153724677, 0.9990000128746033, 1.0, 1.0], 'val_loss': [0.4016987979412079, 0.3145190477371216, 0.2639421820640564, 0.2753584384918213, 0.2677660286426544], 'val_accuracy': [0.8700000047683716, 0.8974999785423279, 0.9375, 0.9350000023841858, 0.9375], 'lr': [0.001, 0.001, 0.001, 0.001, 0.0005]}

Figure 5: History of model

The history of the model is shown in the figure above using the history function of Tensorflow to callback the event

recorded into the 'history2' object during the training of the model, the training accuracy is 100% and the validation accuracy is 93.75% during the last epoch.

E. Prediction

Fig. 6. Random images for prediction

Four	Eight	Nine	Four	One	Three	Nine	Eight	Six	Two
Four	Eight	Nine	Four	One	Three	Nine	Eight	Seven	Two

Fig. 7. Predicted label and correct label of images

Some prediction on the data from the test dataset is made to simulate the working of the system in real life. The figure above shows the predicted label of the image predicted by the model and the correct label of the image. The average accuracy of the model trained using ReLU as the activation function in the three dense layers is 93.25%.

IV. RESULT AND DISCUSSION

A. Discussion on implementing

Evaluating how different activation functions affect the accuracy of the CNN model is our research topic. Each different activation function would be applied to the train and test the image data set which is 300 for training and 40 for testing for getting the accuracy. The different activation functions used are ReLU, Sigmoid, Tanh, Softmax, ThresholdedReLU, ELU, PReLU. The table below shows the summary result of the different activation functions that affected the accuracy of the CNN model:

B. Discussion on the result

TABLE I. ACCURACY OF MODELS (3 RUN)

Activation Function	1st Run Attempt	2nd Run Attempt	3rd Run Attempt	Average Accuracy
ReLU	95.24999857	91.75000191	93.75	93.58333349
Sigmoid	62.00000048	54.75000143	44.49999928	53.7500004
Tanh	90.24999738	89.24999833	92.75000095	90.74999889
Softmax	10.00000015	10.00000015	10.00000015	10.00000015
Softplus	94.24999952	92.50000119	91.50000215	92.75000095
ThresholdedReLU	93.75	93.99999976	91.25000238	93.00000072
ELU	93.00000072	92.25000143	92.25000143	92.50000119
PReLU	94.49999928	94.49999928	93.75	94.24999952

As the table above shows, the ReLU activation function demonstrated the highest accuracy for the first run attempt, but the result of the 2nd run, and 3rd run showed a significant decrease leading to the average accuracy lower than the PReLU. The average accuracy of Sigmoid and Softplus is the lowest compared to other activation functions implying that they might not be the best option in CNN models in image classification tasks. Softplus and Thresholded ReLU also performed well and demonstrated their dependability. ELU activation function is demonstrated the most stable compared to others although it is not the highest average accuracy. The last, the PReLU activation function has the highest average accuracy compared to others although in the first run, accuracy is not the highest, but it still performs well in the 2nd and 3rd runs.

Fig. 8. shows the average accuracy for each of the activation functions. The Red horizontal line represents the activation function that this CNN model author used. As the result shown in the histogram below, the PReLU has the highest average accuracy, and it represents this activation function has the best performance for handling CNN models in the image classification task than ReLU.

In summary, the choice of activation function will affect the training accuracy of the CNN model. Based on our results, ReLU, Tanh, SoftPlus, ThresholdedReLU, ELU, and PReLU are viable choices for image classification tasks, among them, PReLU has the highest average accuracy.

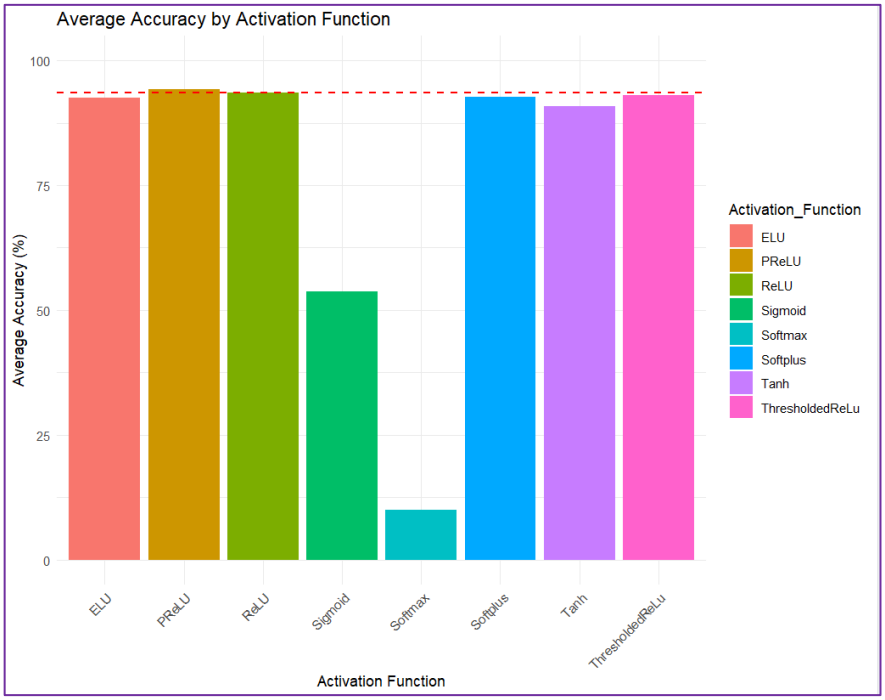

Fig. 8. The histogram of average accuracy based on activation functions

C. *Validation of result*

TABLE II. ACCURACY OF MODELS RUN FOR 10 TIMES

Number of Trial	ReLU	PReLU
1st Run Attempt	93.00000072	94.99999881
2nd Run Attempt	92.25000143	93.99999976
3rd Run Attempt	94.49999928	93.50000024
4th Run Attempt	90.49999714	95.24999857
5th Run Attempt	93.25000048	92.75000095
6th Run Attempt	92.00000167	92.50000119
7th Run Attempt	94.24999952	92.75000095
8th Run Attempt	94.74999905	93.25000048
9th Run Attempt	92.75000095	89.99999762
10th Run Attempt	92.25000143	93.25000048
Average Accuracy	92.95000017	93.2249999

To reduce the impact of randomness in testing and validate the previous finding in section B, we conducted 10 more trials on ReLU and PreLU. Table II shows the outcome of the trial attempt. With more attempts, the average accuracy of ReLU and PReLU become more comparable.

D. *Further explanation of findings*

The difference between ReLU and PReLU is that ReLU only produces 2 outputs, which are 0 and positive input from the previous layer, whereas PReLU allows a small portion value for the slope for negative (Khandelwal, 2021).

However, in our case, the input image is in RGB form (range from 0 to 255), which will not be passing negative numbers into the activation function. Without any negative input passed to the activation function, ReLU and PReLU work similarly, and both will pass the input as output. Hence, the accuracy of ReLU and PReLU is reasonable to be similar.

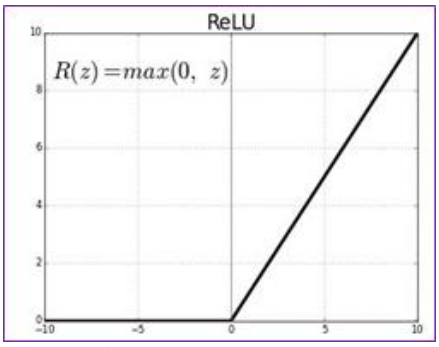

Fig. 9. The formula of ReLU

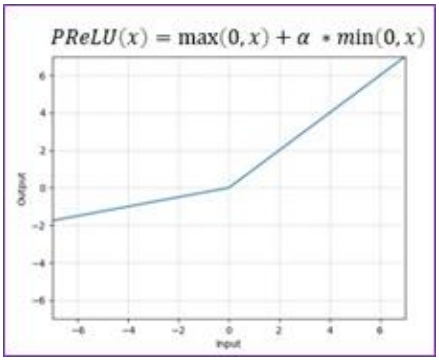

Fig. 10. The formula of PReLU

V. CONCLUSION

In short, the activation function has an impact on the accuracy of the sign-recognition model. Based on the obtained results, the activation function of "PReLU" has the highest average accuracy among the other activation functions. In addition, we can see that based on the results of gesture recognition attempts, we can observe that the results fluctuated in different attempts. The attempts do not guarantee the accuracy to be higher than the previous attempts although it already went through another loop of training.

Despite the research attaining an objectively better result in terms of accuracy when PReLU is used instead of ReLu in the dense layers, it is still important to note that there are some further improvements that can be made to the training of the model. This section will explore our research's limitations and what should be changed to improve the validity in our results in later efforts. Firstly, our dataset is comparably small when it comes to training a Convolutional Neural Network model. The existing dataset created by us consists of a total of 3,400 images, which contain 3,000 images of train data and 400 images of test data. The dataset can be further diversified with data considering the size of hands, distance between hands and camera, first and second-person perspective as well as the cultural differences in representing numbers with hand gestures. For instance, Chinese hand gestures for numbers differ from how they are usually represented universally.

Other than that, in order to get a higher average accuracy, we should train more than 3 times for each activation function so that the randomness in the accuracy data can be decreased.

Additionally, it should be noted that the dataset can further be enhanced by implementing solutions within the system to "denoise" the images. CNNs can be trained to map noisy images to clean ones by using deeper architectures that feature multiple convolutional layers to capture the key features of the images that are retained. This increases the chances of non-key features in an image being reduced significantly since noise would generally fall under the category of non-key features. Batch normalization can also be used in the CNN architecture. By normalizing the activations of a layer, it can help reduce the internal covariate shift. This contributes to denoising because it makes the network training more stable. Moreover, another improvement that can be made would be standardizing the quality of the webcam used to collect images for the dataset since it would ensure uniformity in the images.

REFERENCES

Chauhan, R., Ghanshala, K. K., & Joshi, R. (2018). Convolutional Neural Network (CNN) for Image Detection and Recognition. *First International Conference on Secure Cyber Computing and Communication (ICSCCC)*. Jalandhar: Institute of Electrical and Electronics Engineers (IEEE).

DataFlair. (n.d.). *DataFlair*. Retrieved from Sign Language Recognition Using Python and OpenCV: https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/

Dwijayanti, S., Hermawati, Taqiyyah, S. I., Hikmarika, H., & Suprapto, B. Y. (2021). Indonesia Sign Language Recognition using. *(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 12*.

Khan, I., Rana, S., Ansari, N. M., Iqbal, R., Tariq, T., Awan, M. U., & Waqar, A. (2022). DESIGN AND IMPLEMENTATION OF CNN FOR SIGN LANGUAGE RECOGNITION. *Journal of Jilin University (Engineering and Technology Edition) Vol. 41*, 135-145.

Khandelwal, R. (2021, March 16). *Different Activation Functions for Deep Neural Networks You Should Know*. Retrieved from medium: https://medium.com/geekculture/different-activation-functions-for-deep-neural-networks-you-should-know-ea5e86f51e84

Lau, Y., Sim, W., Chew, K., Ng, Y., & Salam, Z. A. (2021). Understanding how noise affects the accuracy of CNN image classification. *Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 5*.

Seng, L. M., Chiang, B. B., Salam, Z. A., Tan, G. Y., & Chai, H. T. (2017). MNIST handwritten digit recognition with different CNN architectures. *Journal of Applied Technology and Innovation (e -ISSN: 2600-7304) vol. 5*.