CSE676 Final Project
Chirologia Recognition By Deep Learning

Venkateswararao Para
UB Number: 50442199/ University at Buffalo
vpara@buffalo.edu

Akshey Ram Murali
UB Number: 50442206/ University at Buffalo
aksheyra@buffalo.edu

1 Introduction

The individuals have their own way of communicating with each other. This includes actions like body postures, facial aspects, oral sentences. But people who are deaf-dumb and not physically challenged are limited to communicate with a hand gesture. In this scenario it is impossible for them to address the persons who are usual and not impaired. A tool developed using the deep learning frame will become beneficial to ease the sign language communication.

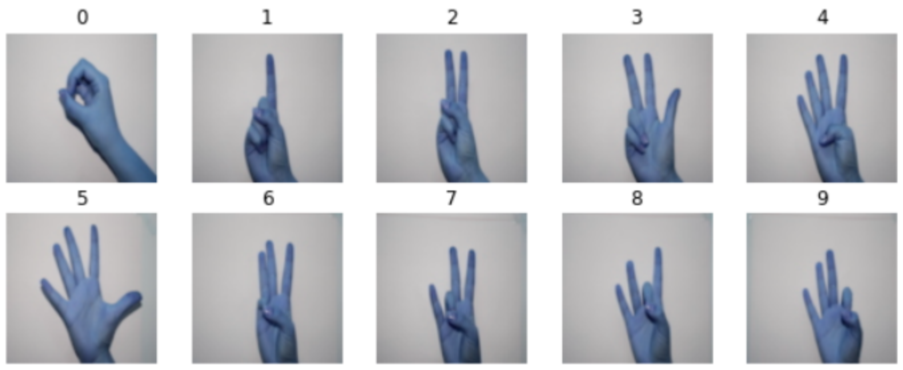


Figure 1: Images for Sign Language Representing the digits 0-9

[1] In reality the frames of the images making up the video is detected by a deep learning classification model. By this way the communication happens. However there are some limitations of using this system as far the image is considered:

1. The pixel clarity which defines the model training and model performance.
2. The image must populate the hand atleast until wrist with fingers.

In this paper, we classified the sign language for the digits varying from 0-9 as shown in figure 1. Although it can be extended to alphabets as shown in figure 2. [2] With this purpose the convolutional neural network (CNN) deep learning framework is utilized.

2 Problem Statement

In the nature to socialize with individuals it is important to have a communication in a meaningful manner and that is in usual cases a oral communication. However people with oral and hearing

[1]Adaptive from [7]
[2]Adaptive from [3]

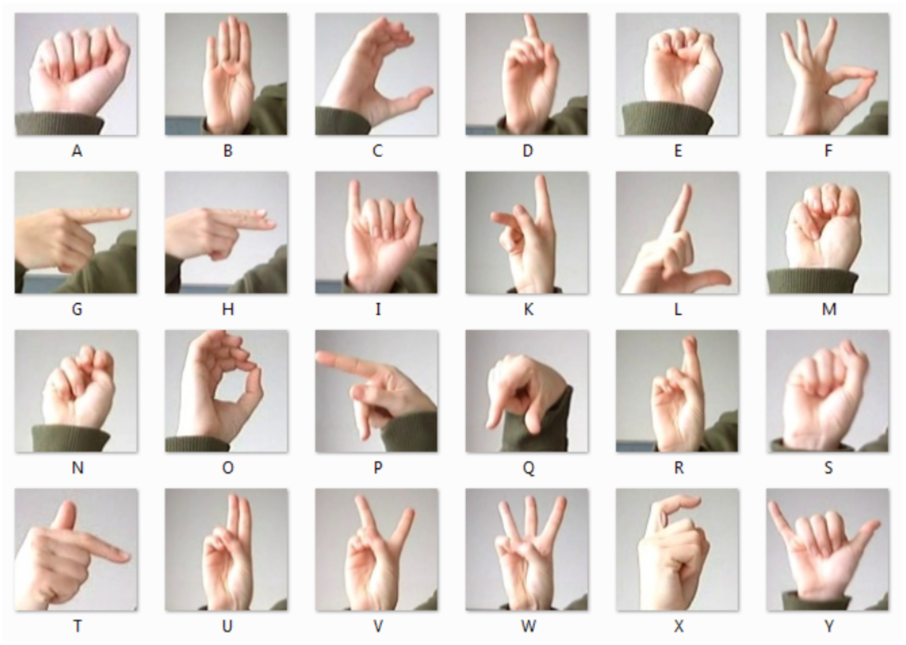


Figure 2: Images for Sign Language Representing the digits A-Y

disability need an alternative way to communicate other than vocal channel. And it is important to understand the importance of each gesture, for this purpose a lot of training is required. The one another important factor is the time complexity of the model and the working architecture that also defines the space complexity. The technology would be a mediator that express the thoughts to other individual in a potential manner. This paper is dedicated to address this problem.

3 Data Preparation and Feature Engineering

The training data set consists of 185 images for each of the digit from 0 to 9 individually. The testing data set consists of 20 images again for each of the digit individually. The images are plain and as per the necessity data augmentation had been implemented to create the robust training methodology. There are also other techniques i.e more than 60 in practice such as erosion, edge detection, dilation, fourier-transform, etc to name a few.

The generalization cannot be achieved with the datasets having no variations. For the purpose to reduce training error and generalization error the more data is required. This is achieved through data augmentation. There are two types of data augmentation namely synthetic data augmentation and real data augmentation.

The Keras ImageDataGenerator with the arguments re-scale, horizontal flip, vertical flip, rotation range, zoom range is used as a generator to extend and implement the variation data-set. And the precision and accuracy of the model training and its behavior was improved. It must be interpreted that the project execution happened successfully with about 45% being dedicated to particular section.

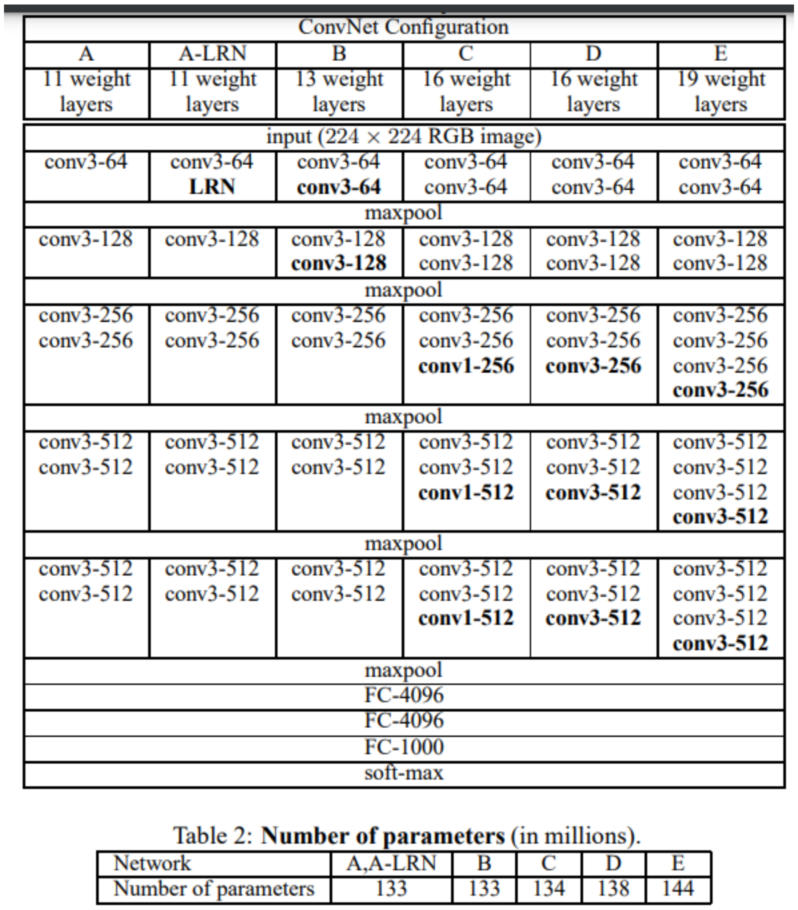
4 Deep Learning Network For Chirologia Classification

The deep learning architecture used for the purpose of image recognition has the following configuration and the associated blocks:

- A RGB input image with 224 x 224 dimentionality pixels
- Two convolution layers with 64 number of filters each and ReLU activation function
- Two convolution layers with 128 number of filters each and ReLU activation function
- Three convolution layers with 256 number of filters each and ReLU activation function
- Six convolution layers with 512 number of filters each and ReLU activation function

• Five pooling layers
• Three fully connected layers and ReLU activation function

[3] In total there are 16 weight layers. The fully connected layers have 4096 channels each for two and the other 1000 channels. 3D convolution layer is convolved with the input layer to result a tensor of outputs. The outputs then pass through the activation function which again pass through the stack of convolution layers involving the same concept. The maxpooling blocks attesting the architecture is to perform dimentionality reduction of the produced output in the previous layer. Max-pooling is performed over a 2×2 pixel window, with stride 2. Three Fully-Connected (FC) layers follow a stack of convolutional layers with the corresponding depths: the first two had 4096 channels each, the third performs 1000-way classification and thus contains 1000 channels one for each of the digit. The ultimate layer is the soft-max layer. The configuration of the fully connected layers follows a similar fashion in all networks.



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figure 3: ConvNet Configuration

The kernels of the convolution layer defines the modulation to the input that is of fixed size 224 x 224 RGB image. For instance a 3D convolution layer with 64 kernel size will follow the demonstration as show in the figure 3. The few of the arguments defining the kernel operation are the kernel size, strides, padding that transforms the input. A stack of convolution layers with the respective depths uses a standard stride of 1 pixel. The convolution stride is fixed to 1 pixel; the spatial padding of convolution layer input is such that the spatial resolution is preserved after convolution. Hence, the padding is 1-pixel for 3×3 convolution layers. Spatial pooling is carried out by each of the max-pooling layers [4], which follow some of the convolution layers. All hidden layers are collaborated with the ReLU non-linearity activation function and only one follow local response normalization

[3]Adaptive from [5]
[4]Adaptive from [2]

(LRN). However such a normalization does not benefit the model performance, but leads to increased time complexity and space complexity of over 533MB. Thus the model performance downgrade due to slow training due to large network architecture weights.

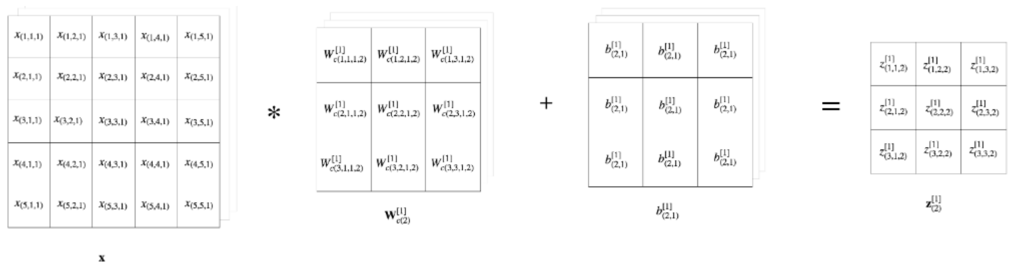


Figure 4: Convolution Kernel

Let number of training samples be 256, J be the cost function,x be the input vector, y be the actual output vector and ŷ be the prediction vector. The cost function formulae is demonstrated in the figure below. [5] The notice is that the 'imagenet' architecture default weights are used as initialization.

$$\hat{y}_i = P(y = i|\mathbf{x})$$

$$log\ p(\mathbf{y}|\hat{\mathbf{y}}, \mathbf{x}) = \sum_{i=1}^{1000} log\ \hat{y}_i^{y_i}$$

$$\mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}) = -log\ p(\mathbf{y}|\hat{\mathbf{y}}, \mathbf{x})$$

$$J = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$$

$$\frac{\partial J}{\partial \mathbf{W}^{[j]}} = \frac{1}{256}\sum_{i=1}^{256}\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[j]}}$$

Figure 5: Cost Function

[6] The backpropogation procedure follows the policy to minimize the loss function 'L' , with an intention to maximize the probability of ŷ which must be equal to 'y' given the feature vector 'x'. The y belongs to a categorical distribution/class. Also let 'W' and 'b' be the trainable parameters.

[7] The gradients and the partial derivative of L with respect to both W, b and with respect to z are calculated to estimate the error propogated from the preceding layer.

[8] The 138 parameters are in total for the neural network architecture and the performance metric of multiple test scale is highlighted in Figure 8. Majorly it depends on the algorithm and the processing speed of the keras GPU. Also, parameter sharing is one major aspect of defining the model efficiency.

[9] Because the necessity of image recognition were realized with the particular custom architecture that was discussed in the above sections. The respective design principles in the deep learning development framework. In our case, it is tensorflow keras. The number of epochs had been five and the early stopping and drop out phenomena are used to efficiently reduce the model training time. Also, controlled are the gradient descent vanishing problem and avalanche optimization.

5 Observations

The validation data set has twenty images for each of the digit from 0-9. As in figure 9 the model predicts the class of an input image.

[5] Adaptive from [1]
[6] Adaptive from [10]
[7] Adaptive from [8]
[8] Adaptive from [5]
[9] Adaptive from [6]

$$\left(d\mathbf{W}^{[16]}, d\mathbf{b}^{[16]}, \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[16]}} \right)$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[16]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[16]}} \frac{\partial \mathbf{a}^{[16]}}{\partial \mathbf{z}^{[16]}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[16]}} = \begin{bmatrix} -\frac{y_1}{a^{[16]}_{(1,1)}} & -\frac{y_2}{a^{[16]}_{(2,1)}} & \cdots & -\frac{y_{1000}}{a^{[16]}_{(1000,1)}} \end{bmatrix} \begin{bmatrix} a^{[16]}_{(1,1)}(1 - a^{[16]}_{(1,1)}) & -a^{[16]}_{(1,1)}a^{[16]}_{(2,1)} & \cdots & -a^{[16]}_{(1,1)}a^{[16]}_{(999,1)} & -a^{[16]}_{(1,1)}a^{[16]}_{(1000,1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -a^{[16]}_{(1000,1)}a^{[16]}_{(1,1)} & -a^{[16]}_{(1000,1)}a^{[16]}_{(2,1)} & \cdots & -a^{[16]}_{(1000,1)}a^{[16]}_{(999,1)} & a^{[16]}_{(1000,1)}(1 - a^{[16]}_{(1000,1)}) \end{bmatrix}$$

Figure 6: Gradient Descent Loss Function

In our case the activation function is ReLU

$$g(z) = \begin{cases} z & \text{if } z > 0 \\ 0 & \text{if } z =< 0 \end{cases} \qquad g'(z) = \begin{cases} 1 & \text{if } z > 0 \\ \text{Undefined} & \text{if } z = 0 \\ 0 & \text{if } z < 0 \end{cases}$$

$$\frac{\partial \mathbf{a}^{[13]}}{\partial \mathbf{z}^{[13]}} = \begin{bmatrix} g'(z^{[13]}_{(1,1,1)}) & 0 & \cdots & 0 & 0 \\ 0 & g'(z^{[13]}_{(2,1,1)}) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & g'(z^{[13]}_{(13,14,512)}) & 0 \\ 0 & 0 & \cdots & 0 & g'(z^{[13]}_{(14,14,512)}) \end{bmatrix}$$

Figure 7: Gradient Descent Activation Function

The accuracy measured over train data and the validation data are demonstrated using figure 10. The model is trained with five epochs only and the performance will be much better when the epochs would be more than 30 [10], which actually is the normal scenario.

[10]Adaptive from [4]

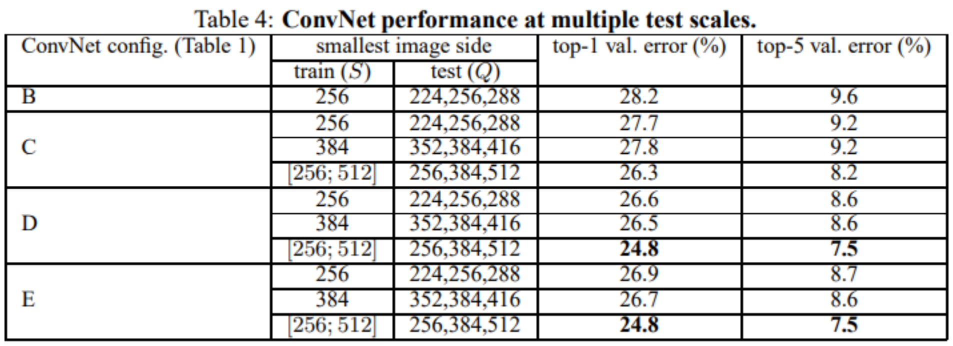
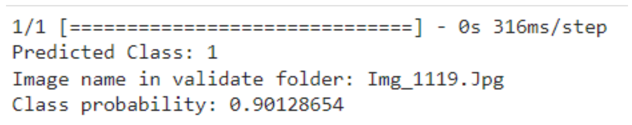


Table 4: **ConvNet performance at multiple test scales.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	**24.8**	**7.5**

Figure 8: Neural Network Performance



```
1/1 [==============================] - 0s 316ms/step
Predicted Class: 1
Image name in validate folder: Img_1119.Jpg
Class probability: 0.90128654
```

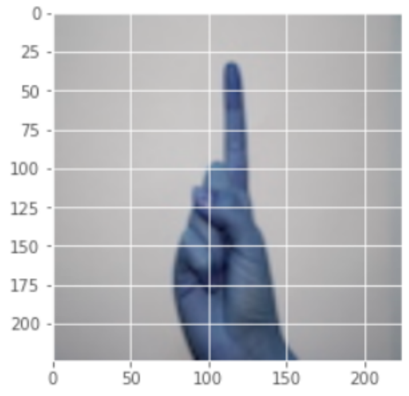


Figure 9: Prediction For Digit 1

6 Further Scope

Background subtraction (BS) is a common technique to mask off the back-end pixels by populating the distinguishing edges and therefore the necessary image. [11] In reality the live video having the moving objects in capture will be populated using the frame subtraction principle. Background image processing process involves two steps listed below and the figure 10 shows the results.

• Initiation where the background is identified and manipulated;
• Update to adapt to background suppress.

The image erosion is another concept where-in the pixel value is calculated to minimum. The estimated value will be the new pixel replacement. By this idea of erosion the dark regions elevates. While the brighter shade decreases. Dilation is a technique where we expand the image. The number of pixels around the object will be enhanced. The structuring element is a matrix of 1's and 0's controls this operation.

All of the generated images using the above concepts will be used for training the deep learning network for the purpose of image classification. This time the epochs must be more just to make sure that the accuracy reaches to the optimal level by making sure of the concerns with over fitting. As an extension to this paper the further scope could be to make the model much robust by adapting to

[11] Adaptive from [9]

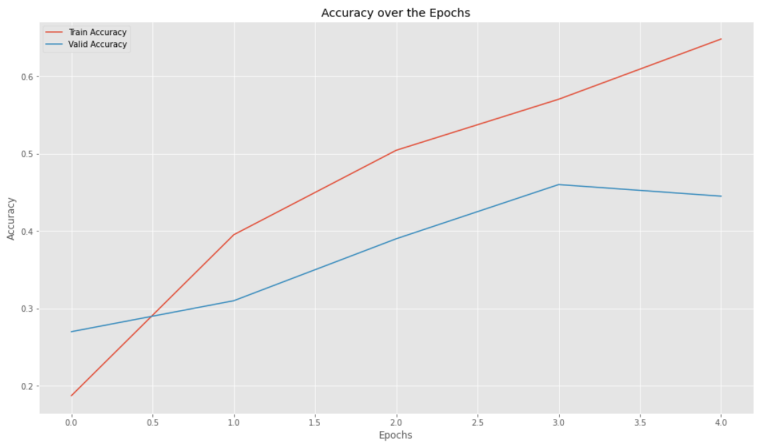


Figure 10: Accuracy Metric

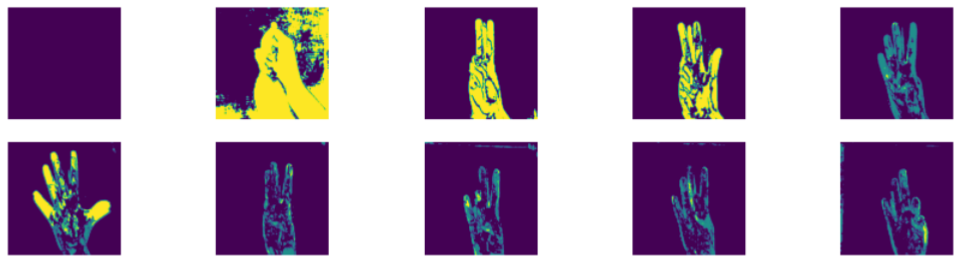


Figure 11: Evaluation using Background Substraction Images

images classification of under the criteria of varying inputs. Also one another aspect is to using the boosted deep learning network which improves the parameter tuning and model precision.

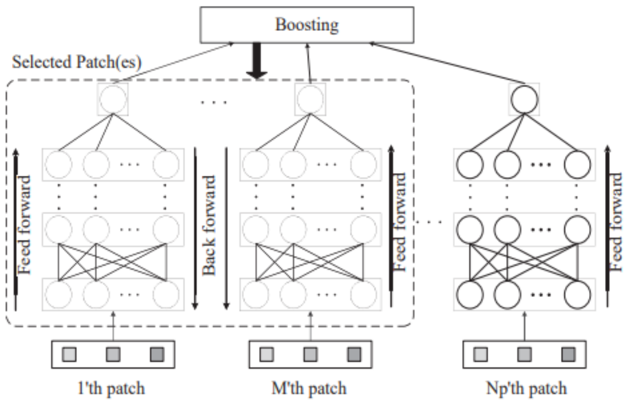


Figure 12: Boosting in Deep Learning

References

[1] I. Sutskever A. Krizhevsky and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:pp. 1097–1105, 2012.

[2] Nazri Mohd Nawi Ali Arshad Saman Riaz Abdulrahman Alruban Ashit Kumar Dutta Afia Zafar, Muhammad Aamir and Sultan Almotairi. A comparison of poolingmethods for convolutional neural networks. *Applied Sciences Digital Object Identifier*, 2022.

[3] Faliang Huang Kaili Wang Liqiong Lu, Yaohua Yi and Qi Wang. Integrating local cnn and global cnn for script identification in natural scene images. *IEEE 10.1109/ACCESS.2019.2911964*, 2019.

[4] Dr. Smitha Rao Saahil Afaq. Significance of epochs on training a neural network. *International Journal Of Scientifc and Technology Research*, 9(6), 2020.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2015.

[6] Jingkuan Song Xuanhan Wang, Lianli Gao. Beyond frame-level cnn: Saliency-aware 3-d cnn with lstm for video action recognition. *IEEE Signal Processing Letters*, 24(4), 2017.

[7] Shuo Wang Xuedan Du, Yinghao Cai and Leijie Zhang. Overview of deep learning. *31st Youth Academic Annual COnference of Chinese Association of Automation*, 2016.

[8] J.S. Denker D. Henderson R.E. Howard W. Hubbard L.D. Jackel Y. Le Cun, B. Boser. Handwritten digit recognition with a back-propagation network. *In Advances in neural information processing*, 1990.

[9] Fu Jie Huang Yann LeCun and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *The Courant Institute, New York University*, 21:pp. 197–115, 2011.

[10] Özgür Ceyhan. Back propogation. *Algorithmic Complexities in Backpropagation and Tropical Neural Networks*, 2021.