American Sign Language and Digits recognition using 2D Neural Network

Muhammad Sabih ul Hassan
Department of Robotics and Intelligent Machine Engineering
National University of Science and Technology (NUST)
Islamabad, Pakistan
mhassan.rime22smme@student.nust.edu.pk

Abstract

Sign Language is an important mode of communication for the deaf population and the general public. Proposed system is a sign language and digit detection system based on convolutional neural network. This research focuses on applying Deep Learning methods to analyze sign language symbols and create helpful predictions. We used ASL dataset for English alphabets and sign language digit dataset from Ankara Ayranc Anadolu High School in Turkey. We used deep learning techniques, computer vision, neural network, etc. This model continuously classifies alphabets (a-z) and digits (0-9). The proposed model has a training accuracy of *99.16%* and validation accuracy of *97.23%*.

Keywords—*ASL; deep learning; computer vision; Convolution Neural Network*

I. INTRODUCTION

Millions of people around the world utilize Sign Language, a rich and complicated visual language. The major language of the Deaf and Hard-of-Hearing community in the United States and Canada is American Sign Language and is also used in other parts of the world. While there have been significant advances in automatic sign language recognition (ASLR) in recent years, accurate recognition of ASL and digits remains a challenging task due to the high variability and complexity of sign movements.

Traditional machine learning algorithms for ASLR rely on handcrafted features, which may not capture the full range of variation in sign language. To solve this issue, we offer a unique method for ASL and digit recognition based on a 2D neural network. Our technique is based on recent breakthroughs in deep learning and convolutional neural networks (CNNs), which have proven cutting-edge performance on a variety of computer vision applications. By using the power of deep learning and CNNs, we aim to develop a robust and accurate system for ASL and digits recognition that can be deployed in real-world settings. In this paper, we present the design and implementation of our 2D neural network for sign language and digits recognition. We evaluate our system on a large dataset of ASL and digits. Our experiments demonstrate that our system achieves high performance highlighting the potential of deep learning for ASLR [1].

The rest of this paper is structured as follows. Section II contains a review of related work on ASLR. and deep learning, highlighting the limitations of traditional approaches and the potential of deep learning for ASLR. Section III describes our methodology and the architecture of our 2D neural network, including details on data pre-processing, network architecture, and training parameters. Section IV presents our experimental results and finally, concludes the paper.

II. REALTED WORK

In the past, neural networks have been used to recognize ASL letters with accuracies of more than 90% [2-11]. However, they have complex functionality. Our system takes live video as input and extracts separate video frames and generate probabilities for each letter in 2D Convolutional Neural Network (a through z and numbers 0 to 9).

Over the course of the past several years, ASL recognition has been studied extensively as a computer vision problem. Researchers have employed various categories of classifiers including Linear classifiers, neural networks, and Bayesian networks are examples of neural networks [2-11]. Multi-column deep CNNs that use several parallel networks considerably enhance recognition tasks.

Classifiers are very straightforward models to work with, but they require sophisticated feature extraction and preprocessing approaches to be effective. Using Karhunen-Loeve Transform, Das and Singha achieved over 95% accuracy on hand gesture model with 10 classes. After preprocessing the images, they employed a linear classifier to predict hand gestures such as peace, ok, etc. Sharma et al. achieved accuracy of 62% using SVM on color channel model. After background subtraction and noise removal [4], they employed SVM and KNN models to characterize each color channel model. Anjaly S. Menon, C J Sruthi, A. Lijiya achieved an accuracy of 99.67% on Indian sign language dataset using CNN which they compared with InceptionV3 and VGG16 models with 98.67% and 99% accuracy, respectively, which are among the top models for sign language recognition [12].

In the past, neural networks were utilised to solve translation problems. However, they require considerable time and resources to train and most of them are shalllow. Mekala et al. used advanced feature extraction and 3-layer Neural Network to classify video of ASL letters into text [8]. Before ASL classification, they identify hand points of interest: palm center and fingertips. They claim to be able to classify 100% of photographs using this method, although there is no proof of this.

Using a feedforward Neural Network Admasu and Raimond correctly classified 98% of cases. They used

preprocessing and extracted features using Gabor Filter and Principal Component Analysis [9].

Hidden Markov Models (Bayesian Network) have also achieved good results. To track hand movement, Starner and Pentland used a Hidden Markov Model (HMM) and a 3D glove. Because the glove can acquire 3D data from hand movements, they attained an accuracy of 99% on the test set. Their model detects and classifies hand movements based on past hand postures. [5].

L. Pigou et al.'s implementation of CNNs to categorize 20 Italian gestures from the ChaLearn 2014 conference. One of the most notable works in this sector is the Looking at People gesture spotting competition. [11]. They achieved validation accuracy of 91% using Microsoft Kinect was used to capture full-body images of people executing hand movements. Kinetic captures depth features that aid in the classification of ASL signs.

Das et al. achieved an accuracy of 90% by training an InceptionV3 CNN on 24 classes of alphabets except for 'J' [13]. They developed a deep learning SLR system that makes use of processed static photos of ASL gestures. Researchers discovered that if an image dataset is properly preprocessed, the InceptionV3 model is suitable for static sign language identification.

III. METHODOLOGY

The proposed model uses a 2D Convolution Neural Network for my model to predict alphabets and digits using hand gestures. To do this, we use a multi-step method that includes obtaining the dataset., preprocessing the images, training and testing the model. Next sections explain these steps in detail.

A. DATASET:

The model is trained using two distinct datasets. One for alphabets and other for digits. By Turkey Ankara Ayranc Anadolu High School Students (SLD), the American Sign Language Dataset (ASL) is used for alphabets and the Sign Language Digits Dataset (SLD). ASL dataset contains 3000 images of each of 29 classes (87000) of which 26 are alphabets (a-z) and other three are space, nothing and delete which are helpful in real time applications. Each image is 200x200 pixels. Sample images of some hand gestures are shown in Fig. 1.

SLD contains 205 images of 10 classes (0-9). Each image is 100x100 pixels in RGB color space. This dataset was collected by students from Turkey's Ankara Ayranc Anadolu High School. 218 students participated in this and 10 samples were collected from each student. Fig. 2 shows the sample images of digits.

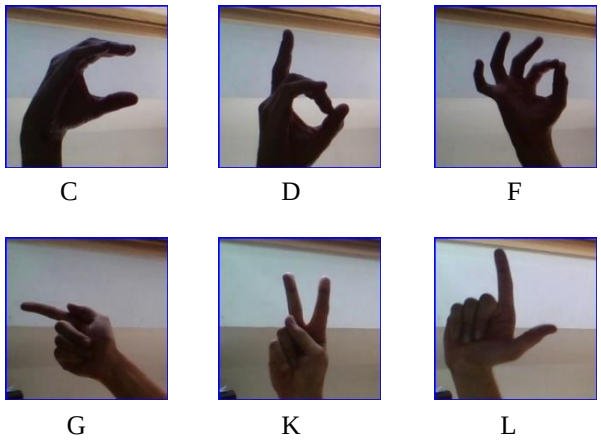

Figure 2. Some letters from ASL Dataset

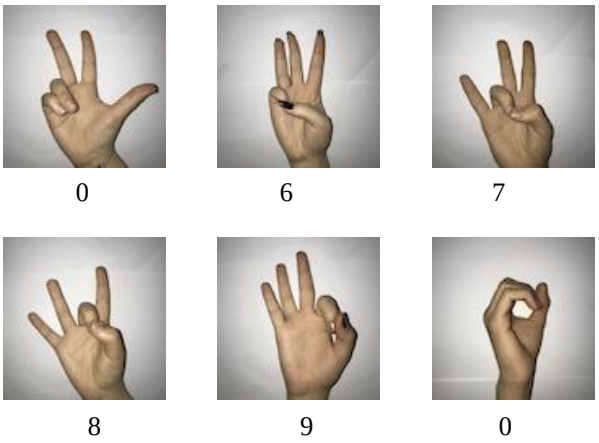

Figure 1. Some numbers from SLD Dataset

B. PREPROCESSING:

The first step in preprocessing the images is to resize them to a standardized size of 224x224 pixels. This ensures that all images have the same dimensions, and this can be easily loaded into the machine learning model.

Once the images are resized, they are saved into a new directory for further processing. To use the images for classification, they need to be encoded into a suitable format. In this case, the images are encoded in a one-vs-all fashion, which means that each image is assigned a label corresponding to the gesture it represents.

The labels are assigned in a one-vs-all fashion, which means that each label corresponds to a unique gesture and all other gestures are treated as negative examples. This enables the machine learning model to learn the patterns unique to each gesture and distinguish it from others.

Once the images are encoded, they are mapped against their labels and saved into a .csv file for further use in training and testing the machine learning model. The .csv file contains the encoded images as well as their corresponding labels, and this can be used to train and test the machine learning model.

C. CLASSIFIER:

Convolutional Neural Networks (CNNs) are a sort of deep learning model that is commonly employed for image categorization tasks. A CNN classifier is employed in this case to classify photos of sign language motions. The CNN architecture is made up of four convolutional layers, followed by two fully linked layers. The first convolutional layer uses a 3-channel input image and applies 16 5x5 filters to generate 16 feature maps. This indicates that the layer concatenates the input image with 16 separate 5x5 filters to extract 16 different features from the image. The second convolutional layer takes the first layer's 16 feature maps as input and applies 32 5x5 filters to produce 32 feature maps. Similarly, the third convolutional layer takes as input the 32 feature maps created by the second layer and applies 64 3x3 filters to produce 64 feature maps. Finally, the fourth convolutional layer takes the 64 feature maps generated by the third layer as input and applies 128 5x5 filters to generate 128 feature maps.

After each convolutional layer, a ReLU activation function is applied. The ReLU function helps to introduce non-linearity into the network by converting any negative values in the feature maps to zero. A max pooling layer is used after each convolutional layer to reduce the spatial size of the feature maps by taking the maximum value in each 2x2 region, in addition to the convolutional layers. This reduces the number of parameters in the model and helps to avoid overfitting.

There are two completely linked layers after the convolutional layers. The first layer takes the flattened feature maps from the previous convolutional layer as input and outputs a hidden representation of 256 units. The second layer takes the hidden representation of 256 units and produces the output logits, where the number of output units is the same as the number of classes in the dataset.

Finally, a SoftMax function is applied to the logits to convert them into class probabilities. The SoftMax function is a form of activation function that translates the last layer's output into a probability distribution over the dataset's classes.

Overall, the architecture of the CNN is designed to extract features from the input images and learn to classify them into the correct sign language gestures. Convolutional layers, max pooling layers, and fully linked layers are used, CNN can learn complex patterns in the input images and produce accurate classifications. The model summary in Fig. 3 provides a detailed overview of the architecture of the CNN.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 16, 220, 220]           1,216
         MaxPool2d-2         [-1, 16, 110, 110]               0
            Conv2d-3         [-1, 32, 106, 106]          12,832
         MaxPool2d-4           [-1, 32, 53, 53]               0
            Conv2d-5           [-1, 64, 51, 51]          18,496
         MaxPool2d-6           [-1, 64, 25, 25]               0
            Conv2d-7          [-1, 128, 21, 21]         204,928
         MaxPool2d-8          [-1, 128, 10, 10]               0
            Linear-9                 [-1, 256]          33,024
           Linear-10                  [-1, 29]           7,453
================================================================
Total params: 277,949
Trainable params: 277,949
Non-trainable params: 0
```

Figure 3. Summary of the model

The above figure shows the architecture of the Neural Network Model proposed in this paper. There are four Convolutional Neural Network layers and below each layer, a MaxPooling layer is used to minimize the spatial size of the feature maps. There are two totally connected layers at the end.

D. DATA AUGMENTATION:

Data augmentation is a popular technique in computer vision and machine learning for increasing the size of a training dataset by modifying existing data. It is especially useful when there is a limited amount of training data available because it helps the model to learn from more diverse examples without having to acquire fresh data.

One popular library for implementing data augmentation in computer vision is Albumentations. Albumentations is a fast and flexible Python library for image augmentation, which supports a wide range of transformations including random crops, rotations, flips, color jittering, and many others. The library is highly optimized for speed, making it ideal for use in deep learning pipelines. After applying data augmentation to the images, they are resized to the same dimensions as the images in the training dataset. This is important to ensure that the model can handle images of the same size during training and inference.

Overall, data augmentation is a valuable method for boosting machine learning model accuracy and resilience, especially in scenarios where there is limited training data available. It can help prevent overfitting and improve the model's generalization capabilities.

E. PARAMETER OPTIMIZATION:

Data optimization is an important step in designing the model which can play an important role in achieving maximum accuracy of the model. ‘Adam’ is an optimizer that implements the Adam algorithm for stochastic optimization of neural network models. The Adam optimization approach is a variation on the well-known Stochastic Gradient Descent (SGD) method that uses adaptive learning rates and gradient updates. Adam is an adaptive learning rate optimisation algorithm that computes specific adaptive learning rates for various parameters based on estimates of the gradient's first and second moments. The algorithm combines the benefits of AdaGrad and RMSProp, two major stochastic optimization strategies.

The Adam optimizer has several hyperparameters that can be set by the user, including beta1, beta2, and epsilon and learning rate is set to 0.001. These control the momentum, scaling, and numerical stability of the algorithm, respectively. By default, the beta1 parameter is set to 0.9, beta2 to 0.999, and epsilon to 1e-8. During training, the optimizer computes the gradients of the loss function with respect to the model parameters and uses the Adam method to update the parameters. The optimizer keeps track of the first and second moments of the gradients for each parameter and uses these estimates to adaptively adjust the learning rate for each parameter. Adam uses exponentially moving averages to estimate the moments, which are computed on the gradient assessed on a current micro batch. Fig. 4 shows averages and squared gradient where m and v are moving averages, g is gradient on current mini batch [14].

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Figure 4. Moving averages of gradient and squared gradient

F. LOSS FUNCTION:

A loss function is a mathematical function that measures the difference between the predicted output of a machine learning model and the actual output, typically given in the form of a training dataset.

A machine learning algorithm's purpose is to minimize the loss function during the training process by altering the model's parameters to make better predictions. The loss function measures how well the model is currently functioning, and the optimization procedure seeks the set of parameters that minimizes this function. Choosing an appropriate loss function is an important aspect of designing a machine learning model, as it can significantly affect the performance and the model's ability to generalize. The loss function utilized is determined by the individual problem being addressed and the type of model being employed.

This model employs the Cross-Entropy Loss function. It calculates the difference between the expected and actual probability distributions of the target classes.

Fig. 5 shows cross entropy loss function.

$$L_{\text{CE}} = -\sum_{i=1}^{n} t_i \log(p_i)$$

Figure 5. Mathematical definition of Cross-Entropy Loss

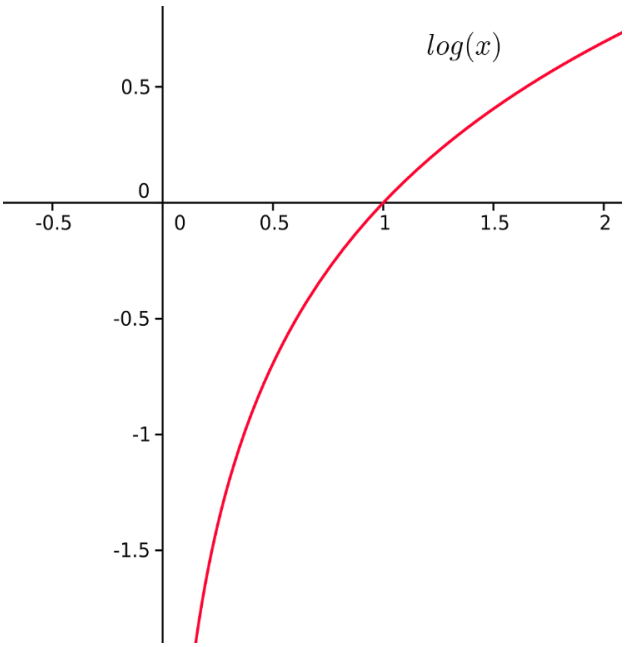

Figure 6. A plot of log(x). For x values between 0 and 1, log(x) <0

Where t_i is the truth label, p_i denotes the SoftMax probability for the i^{th} class, and n denotes the number of classes. The SoftMax activation function and the cross-entropy loss are combined into a single function called by the Cross-Entropy Loss function., which can simplify the implementation of a multiclass classification task. The loss is the negative log-likelihood of the true class, which encourages the model to assign high probabilities to the correct class. If the predicted probability distribution is a perfect match for the true distribution, then the loss is zero.

Fig. 6 shows the negative log of Cross Entropy Loss function [15].

G. TRAINING AND TESTING THE MODEL:

Dataset is split 85 to 15 into training dataset and testing dataset. The dataset contains 3000 images for each of 29 classes of English alphabets (a-z, space, nothing, backspace), and 205 images of 10 classes of numeric digits (0-9). The model is set to train on system's GPU if available else, CPU is used for training. The use of a GPU for training can significantly speed up the training process, as GPUs are designed to perform large-scale parallel computations. However, if a GPU is not available, training can still be performed on a CPU, although it may take longer to train the model. The use of a GPU is often determined by the available hardware resources and the size and complexity of the model being trained.

The model is updated during training using an optimizer, which modifies the model's parameters to minimize the loss function. The optimizer utilized is determined by the individual problem being addressed and the type of model being employed. The Adam optimizer with a learning rate of 0.001 is utilized to train the model in this scenario.

After training the model on the training dataset, the accuracy of the model is evaluated using the testing dataset. The testing dataset is used to assess the model's performance on data that it has never seen before. If the model performs well on the testing dataset, it is likely to perform well on fresh, previously unseen data as well.

H. OUTPUT:

Once the proposed architecture is trained on the dataset, it can be used for the classification phase, where the goal is to predict the corresponding sign language gesture for a given input. During the classification phase, the trained Neural Network takes the input features and creates a probability distribution over different labels corresponding to different sign language motions. The output of the Neural Network is a set of probabilities, where each probability represents the likelihood of the input gesture belonging to a particular gesture class. To determine the final output, the gesture class with the highest probability is chosen as the predicted gesture. The predicted gesture is then mapped to the corresponding

letter or digit using a lookup table. In Fig. 7, the output is shown as a text string representing the predicted letter or digit corresponding to the input gesture. This output can be further processed and used in various sign language translation applications, such as real-time translation for people with hearing impairments. It is important to note that the performance of the model during the classification phase depends on the quality and relevance of the input features, as well as the accuracy of the trained Neural Network. Therefore, it is important to carefully design and preprocess the input features and optimize the Neural Network architecture to achieve high classification accuracy.

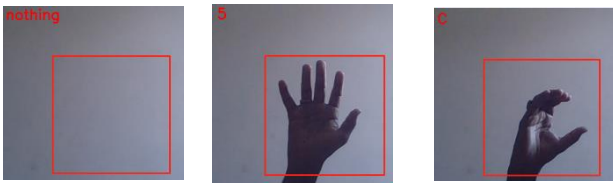

Figure 7. Real time output

IV. Results and Conclusion

The proposed model achieved a training accuracy of 99.16% and validation accuracy of 97% on training data and loss eventually decreases during training. The accuracy of a machine learning model is an important metric that measures the percentage of correctly classified instances in a given dataset and the loss function is also an important metric that measures how well the model is fitting the training data during the training process. Fig. 8 and Fig. 9 show the model accuracy and loss during training respectively. The loss function represents the difference between the model's expected and actual output in the training dataset.

The model's purpose during training is to minimise this loss function in order to obtain higher performance. Fig. 8 shows the accuracy of the proposed model during the training process. As can be seen, the accuracy of the model increases steadily during the first few epochs and then plateaus after a certain point. This means that the model has learned the majority of the patterns in the training data and is not significantly increasing in terms of accuracy. Fig. 9 shows the loss function of the proposed model during the training process. As can be seen, the loss function decreases steadily during the training process, this implies that the model is accurately fitting the training data and minimizing the difference between the expected and actual output.

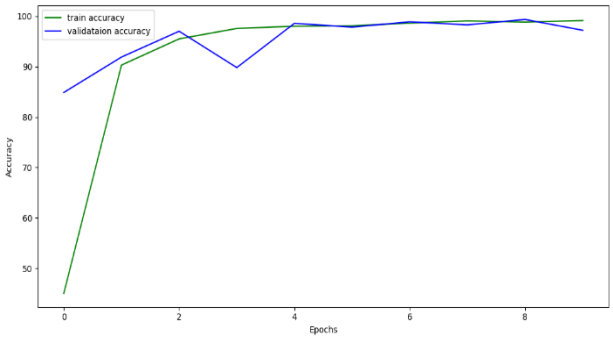

Figure 8. Training and validation accuracy of model

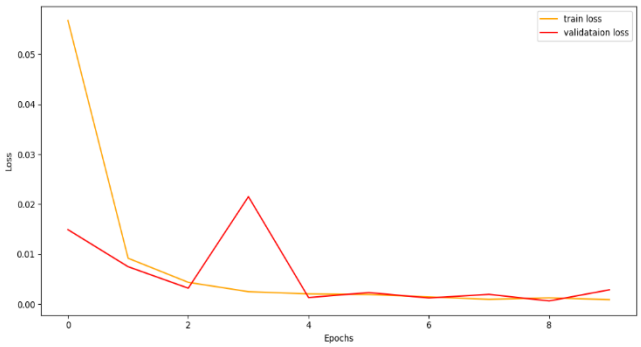

Figure 9. Training loss and validation loss

The fact that the loss function decreases during training is a positive sign, as it means that the model is learning and improving. However, it is critical to track both accuracy and loss during training to verify that the model is not overfitting the training data and can generalize well to new, previously unknown data.

The high accuracy achieved by the proposed model on the training data is a promising result and indicates that the model has the potential to perform well on new, unseen data.

REFERENCES

1. Zhao, R., Zhang, Y., Nan, Y., Wen, W., Chai, X., & Lan, R. (2022). Primitively visually meaningful image encryption: A new paradigm. Information Sciences, 613, 628-648. https://doi.org/10.1016/j.ins.2022.08.027
2. Singha, J. and Das, K. “Hand Gesture Recognition Based on Karhunen-Loeve Transform”, Mobile and Embedded 232 Technology International Conference (MECON), January 17-18, 2013, India. 365-371.
3. D. Aryanie, Y. Heryadi. American Sign Language-Based Finger-spelling Recognition using k-Nearest Neighbors Classifier. 3rd International Conference on Information and Communication Technology (2015) 533-536.
4. R. Sharma et al. Recognition of Single-Handed Sign Language Gestures using Contour Tracing descriptor. Proceedings of the World Congress on Engineering 2013 Vol. II, WCE 2013, July 3 - 5, 2013, London, U.K.
5. T.Starner and A. Pentland. Real-Time American Sign Language Recognition from Video Using Hidden Markov Models. Computational Imaging and Vision, 9(1); 227-243, 1997.
6. M. Jeballi et al. Extension of Hidden Markov Model for Recognizing Large Vocabulary of Sign Language. International Journal of Artificial Intelligence & Applications 4(2); 35-42, 2013
7. H. Suk et al. Hand gesture recognition based on dynamic Bayesian network framework. Patter Recognition 43 (9); 3059-3072, 2010.
8. P. Mekala et al. Real-time Sign Language Recognition based on Neural Network Architecture. System Theory (SSST), 2011 IEEE 43rd Southeastern Symposium 14-16 March 2011.
9. Y.F. Admasu, and K. Raimond, Ethiopian Sign Language Recognition Using Artificial Neural Network. 10th International Conference on Intelligent Systems Design and Applications, 2010. 995-1000.

10. J. Atwood, M. Eicholtz, and J. Farrell. American Sign Language Recognition System. Artificial Intelligence and Machine Learning for Engineering Design. Dept. of Mechanical Engineering, Carnegie Mellon University, 2012.
11. L. Pigou et al. Sign Language Recognition Using Convolutional Neural Networks. European Conference on Computer Vision 6-12 September 2014
12. A. S. Menon, C. J. Sruthi and A. Lijiya, "A 2D Fast Deep Neural Network for Static Indian Sign Language Recognition," 2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON), Faridabad, India, 2022, pp. 311-316, doi: 10.1109/COM-IT-CON54601.2022.9850589.
13. Das, S. Gawde, K. Suratwala and D. Kalbande. (2018) "Sign language recognition using deep learning on custom processed static gesture images," in International Conference on Smart City and Emerging Technology (ICSCET).
14. Bushaev, V. (2018, October 24). Adam - latest trends in deep learning optimization. Medium. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.
15. Koech, K. E. (2022, July 16). Cross-entropy loss function. Medium. https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e.
16. S. Ikram and N. Dhanda, "American Sign Language Recognition using Convolutional Neural Network," 2021 IEEE 4th International Conference on Computing, Power and Communication Technologies (GUCON), Kuala Lumpur, Malaysia, 2021, pp. 1-12, doi: 10.1109/GUCON50781.2021.9573782.