CS550 - Machine Learning

Major Project Final Report

A Convolutional Neural Network approach for Sign Language Translation

2023-24 - M semster

P V Damodaram - 12141200
Muttana Jashraj - 12141110

Contents

1 Github Repository

The github repository for this project: `https://github.com/damodaramPv/ML_project.git`

2 Data Preprocessing

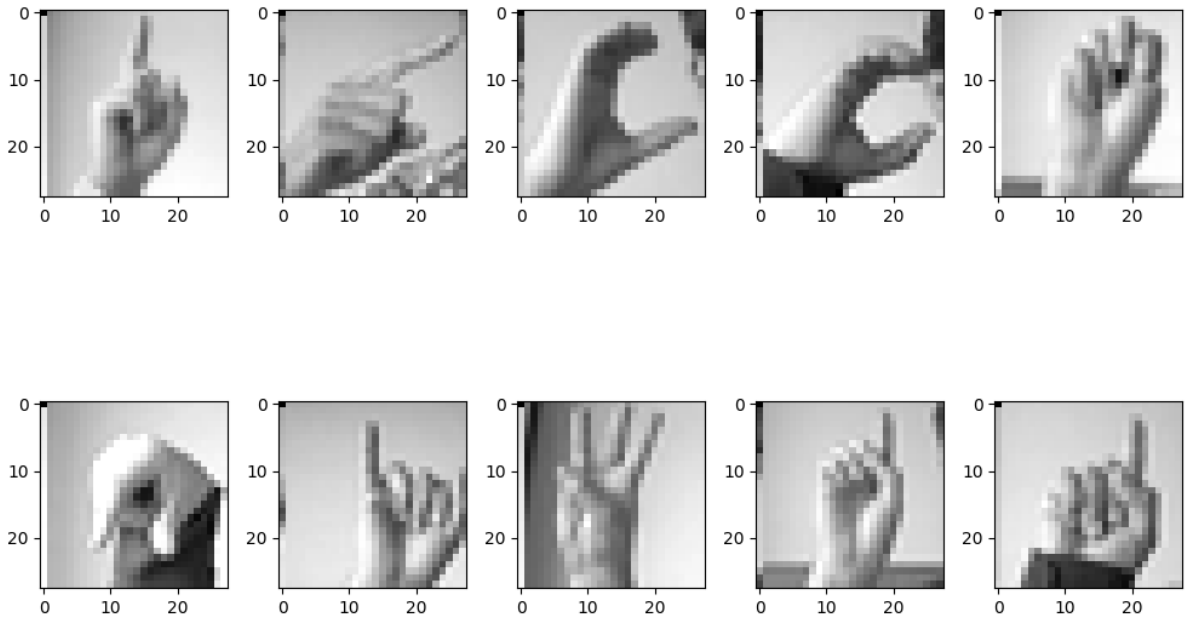
In our project, we adopted several crucial data preprocessing steps to enhance the effectiveness of our machine learning model. Initially, we utilized the LabelBinarizer to transform categorical labels into a binary format, facilitating easier processing and interpretation during the model training phase.

Our next step involved normalizing the pixel values of our image data. By dividing both the training and test datasets by 255, we scaled the pixel intensities to a range between 0 and 1, a common practice to ensure consistent data representation without compromising the inherent structure of the images.

Furthermore, we reshaped the datasets into a standardized format by utilizing a (28, 28, 1) shape. This reshaping ensured that our image data was structured appropriately for compatibility with convolutional neural network architectures, with the '-1' parameter dynamically inferring the batch size based on the input data.

To introduce variation in our training data, we applied various image augmentation techniques. These techniques, such as random rotation (up to 10 degrees), random zooming (0.1 zoom range), and random shifts both horizontally (10% of the total width) and vertically (10% of the total height), aimed to diversify the dataset by generating modified versions of the existing images. Additionally, we refrained from applying horizontal and vertical flips to maintain the integrity of the images without altering their orientation.

By implementing these preprocessing techniques, we aimed to enhance the robustness, generalizability, and variability of our dataset, which ultimately contributed to the overall performance and accuracy of our machine learning model.



3 Modeling

The model was built using the keras library only by getting the following layers - Dense, Conv2D, MaxPool2D , Flatten , Dropout , BatchNormalization, MaxPooling2D layers.

The description of these layers:
Dense Layer: The Dense layer is a fully connected layer where each neuron or node in the layer is connected to every neuron in the preceding layer.
Conv2D Layer: The Conv2D layer is a convolutional layer that is used for 2D spatial convolution over images. It applies a set of filters to the input data, extracting features from the input by sliding the filters over the input's width and height.
MaxPooling2D Layer: MaxPooling2D is a pooling layer that performs down-sampling by taking the maximum value over a set of values within a defined window. It helps reduce the spatial dimensions of the input volume, reducing the number of parameters and computations in the network.
Flatten Layer: The Flatten layer is used to convert the input data into a one-dimensional array or vector.
Dropout Layer: Dropout is a regularization technique where randomly selected neurons are ignored during training. It helps prevent overfitting by reducing the reliance on specific neurons and promotes a more robust network.
BatchNormalization Layer: Batch Normalization is a technique to improve the training of deep neural networks by normalizing the inputs of each layer. It helps stabilize and accelerate the training process.
First we have built the model using these layers to check if the model will work for the data. As expected the CNN model works pretty well for the dataset. The details of the models trained are as follows:

3.1 Model 1

Number of convolution layers = 1
Number of filters in convolution layer = 20
Learning rate = 0.02

Remaining layers are some of the above mentioned layers in the above said layers to maintain the shape appropriately and to drop some of the layers to reduce the parameters.

Result: We have got an accuracy of 86%, which is not that much sufficient. From the validation curves we can infer that this model has validation accuracy greater than training accuracy, which can infer there is an error in the prediction, and training should be improved.

3.2 Model 2

Number of convolution layers = 2
Number of filters in convolution layer = 20, 10 respectively
Learning rate = 0.02

Remaining layers are some of the above mentioned layers in the above said layers to maintain

the shape appropriately and to drop some of the layers to reduce the parameters.

Result: We have got an accuracy of 94%, which is an improvement by comparing it to the previous model. But on seeing the validation curves we can see that the validation accuracy is fluctuating high and low for every epoch which can infer that the model is not trained properly and the model is more dynamic to changes. We have to reduce this fluctuation in the next models.

3.3 Model 3

Number of convolution layers = 2
Number of filters in convolution layer = 32, 64 respectively
Learning rate = 0.0115

Remaining layers are some of the above mentioned layers in the above said layers to maintain the shape appropriately and to drop some of the layers to reduce the parameters.

Result: We have got an accuracy of 90%, which is less than the previous model. But the fluctuations in the validation curves are lesser than the previous curves. So this model is almost ok but we have to improve the accuracy. One thing to note is that this model has more parameters than the previous models, which can be a factor affecting these results.

3.4 Model 4

Number of convolution layers = 3
Number of filters in convolution layer = 32,64,128 respectively
Learning rate = 0.00155

Remaining layers are some of the above mentioned layers in the above said layers to maintain the shape appropriately and to drop some of the layers to reduce the parameters.

Result: We have got an accuracy of 93%, which is better than the previous model. This model has the highest parameters of all models since the number of convolution layers are more in this. So to mitigate this we have added more dropout layers with more dropout so that parameters are reduced. By doing this we got this better accuracy and also the validation curves are smoother compared to previous models.

3.5 Ensemble Model

We have also tried doing the ensemble of all these models created. To perform ensembles we have taken into account of hard voting, where the class is assigned to an input based on the maximum occurred predicted class of all the models in the ensemble.

Result: We actually got a validation accuracy in this case of 98% which is much higher than all the previous models. This can be due to the fact that some models may be detecting some kind of pattern and other models can be detecting other patterns. So the whole ensemble models can detect all the patterns.

4 Hyper-parameter Tuning

From all the models created from the above, We have selected the key learnings from the results and created the final model.

4.1 Final Model

Number of convolution layers = 3
Number of filters in convolution layer = 70,50, 20 respectively
Learning rate = 0.002

Result: From the final model we have got an accuracy of 97%, which is pretty good. In this final model we have taken into account the number of convolution layers to have which is 3, which gave better accuracy for us in the previously built models. And also the fact the learning rate is to be kept around 0.002 gave better results in this model.

4.2 Final Model after tuning

For the final Hyper tuning, we have changed some of the parameters such as the number of filters in the convolution layers and changing the dropout in the dropout layer. We have set the learning rate to 0.01 and letting it to reduce on plateau to get the perfect learning rate and get better results. We have also increased the number of neurons in the dense layer so that the model becomes accurate in detecting relations from the features extracted by the neurons in previous layers.

Number of convolution layers = 3
Number of filters in convolution layer = 75,50, 25 respectively
Learning rate = 0.01 to 0.000015 based on plateau.

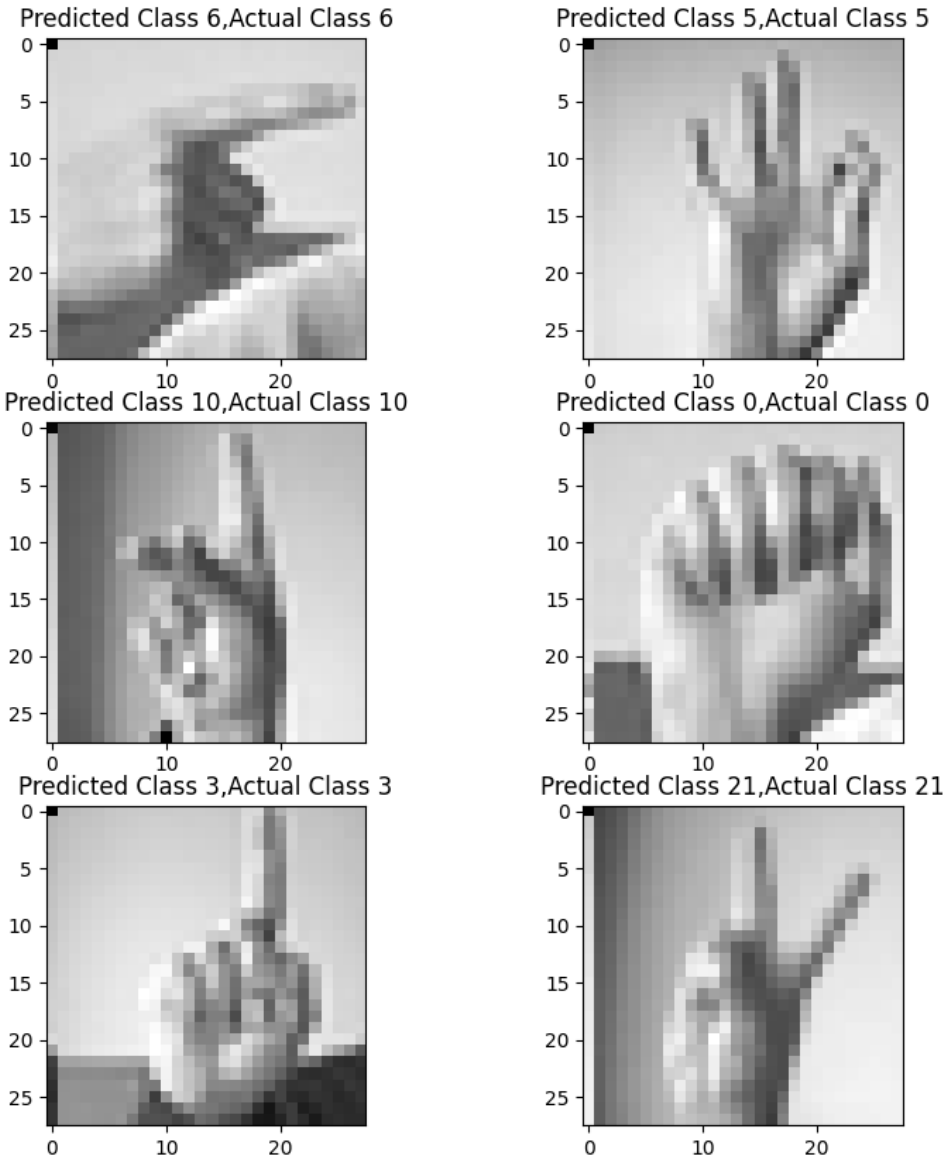
Note: We have tuned the hyper parameters in the modeling stage also where we have experimented with different values for the number of filters, number of layers, number of dropouts and the learning rates also. By doing all these we have to come to the conclusion of the final model with better hyper parameters.

5 Interpretation of results and conclusions

5.1 Results from the final model

We have run the final tuned model upto 20 epochs, and got very good results. The model has been able to achieve the best accuracy possible 1.0. The model has reached the best accuracy after 20 epochs and then after that it just trained for the next epochs to fix these parameters so that there are fluctuations in the validation curves.

Here are some results of the model:



From this we can see that the model is predicting the same as the actual class from the test data.

5.2 Conclusion

We have achieved the goal we stated, that is to successfully classify the Sign - Language images to alphabets (classes). We went through a journey of selecting various models by seeing the metrics of the models such as the validation accuracy and fluctuations in the validation curves. We have successfully got 1.0 accuracy for the test data given in the data set.

We want to conclude that we have successfully classified this data set. This classification may help us normal people to say which alphabet is which from the images. We have presented the models of this project in the github repository, which can be used to predict the new images without training the whole neural network once again.

6 Contributions

P V Damodaram : Done data augmentation in data pre-processing part,Worked with creating model 1 and model 2, created the ensemble model and the final model before tuning.

Muttana Jashraj : Done Data loading and visualization, Done the remaining in data pre-processing, Worked with creating model 3 and model 4, Worked with tuning the hyper parameters for the final model, Done the final output visualization.