AMERICAN SIGN LANGUAGE IMAGE CLASSIFICATION

INDIVIDUAL PROJECT REPORT

Ishita Jain

INTRODUCTION

Sign language is used by the hearing-impaired individuals to communicate within their own community or with others. Sign languages are expressed through hand gestures, that sometimes involve motion. American Sign Language (ASL) is the natural sign language used by the majority of North Americans who are hard of hearing. ASL is expressed by movements of the hand and face. Therefore, identifying and interpreting sign language is a crucial application of computer vision. We attempted to use a Neural Network model implemented in PyTorch for the purposes of this project.

INDIVIDUAL WORK

MODEL ARCHITECTURE

I tried to create custom-built Convolutional Neural Network (CNN) in PyTorch to classify hand images into letters. A CNN is a type of deep neural network that is used for analysis of images and visual data.

I created a CNN subclass with the torch.nn.Module as base class. My CNN model contained 2 convolutional layers, each followed by ReLU activation and 2D Batch Normalization. The output of BatchNorm layers was fed into a Max Pooling layer. As the input images were of size 28x28, padded with zeros to give images of size 38x38 after initial preprocessing, 2 convolutional layers seemed like a good enough number for extraction of important features from the images. The final output of last linear layer had only 24 neurons since we have only 24 classes in our dataset, excluding J and Z. Additionally, since the output classes were integers between 0 and 25, excluding 9=J and 25=Z, the class numbers after 8 had to be reduced by 1, to maintain continuity and compatibility with the model. This was done with the following lines of code:

```
y_train = torch.tensor([y if y<=8 else y-1 for y in y_train]).to(device)

y_test = torch.tensor([y if y<=8 else y-1 for y in y_test]).to(device)
```

ReLU activation is usually considered a good choice for neural networks, especially CNNs.The purpose of applying a ReLU activation function is to increase nonlinearity in the images, as images naturally contain non-linear features. Activation, followed by Max Pooling layer, helps to extract important features from the image.

The formula to calculate the output size of image after convolution or pooling layer is equal to:

$O = (W-K+2P) / S + 1$

Where W: Width of the input image

K: Kernel Size

P: Padding

S: Stride

A combination of different values of hyperparameters were used, but the final custom model was trained with learning rate of 0.1, epochs equal to 15 and batch size of 1000.

The basic architecture of model was as follows.

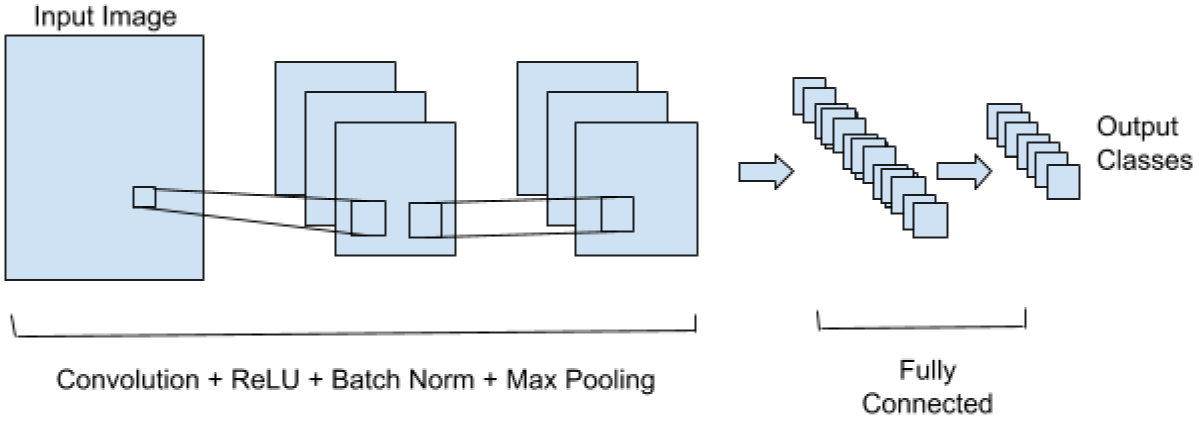

Fig.1 Custom Model Architecture

Layer	Input Image Size	Output Image Size
1. Conv2d	1, 38, 38	20, 36, 36
2. MaxPool2d (Kernel size = 2)	20, 36, 36	20, 18, 18
3. Conv2d	20, 18, 18	20, 14, 14
4. MaxPool2d (Kernel size = 2)	20, 14, 14	20, 7, 7
5. Linear	20*7*7	100
6. Linear	100	24

Fig.2 Input and Output Image Sizes for each layer

RESULTS

The model gave a training accuracy of 36.23%, test accuracy of 86.45%. The F1-Score for test set came out to be 0.86, and Cohen's Kappa score was 0.86. Though the model gave acceptable results but the metrics were not getting improved even after trying multiple combinations of hyperparameters.

SUMMARY AND CONCLUSIONS

The custom-built Convolutional Neural Network worked quite well on the train and test data, however, it could not achieve any further improvement in its performance. Therefore, it was finally decided to employ a pre-trained model for our project.

REFERENCES

1. https://www.nidcd.nih.gov/health/american-sign-language.

2. https://pytorch.org/docs/stable/nn.html.