ASL Recognition using Deep Convolutional Neural Network

Ahmed Rageeb Ahsan
CISE
University of Florida
Gainesville, Florida
ahmedrageebahsan@ufl.edu

Hajymyrat Geldimuradov
CISE
University of Florida
Gainesville, Florida
geldimuradovh@ufl.edu

Md Jabir Hossain
CISE
University of Florida
Gainesville, Florida
mdjabir.hossain@ufl.edu

Abstract—**Recognition tasks have become one of the most remarkable and notable applications of computer vision with the advancement of Convolutional Neural Networks (CNNs). These classification tasks can range from handwritten digit recognition to sign language recognition, the latter of which has seen significant attention due to the possibility of easing difficulties in communication and social interaction for those with visual impairment. Deriving from this task, our work focuses on image classification for nine different American Sign Language (ASL) characters. We propose the use of a convolutional neural network with residual blocks and 18 convolutional layers, alongside necessary data augmentations and preprocessing, to classify these characters based on assigned classes. The pipeline implemented by this paper resulted in an overall accuracy of 98.8% on our test set based on the average accuracy of all classes. We discuss the diverse data augmentation techniques utilized to improve our model's generalization, which resulted in an improvement in accuracy of approximately 6% to 7%.**

I. Introduction

Convolutional neural networks, in particular those with a significant number of layers and hence great complexity, have led to notable breakthroughs in various domain tasks of computer vision, including image classification, object detection and neural style transfer. By performing convolutions at each layer and learning to detect edges and lines of complex content in data, these networks are able to effectively extract high-level, medium-level and low-level features from input data. Consequently, these architectures have also proven to be effective for problems involving the classification of symbols, digits and hand signals.

We propose the use of a deep convolutional neural network for the classification of nine different American Sign Language (ASL) characters, where labels range from 0 to 9. Without including augmentations, the dataset used to train the model consisted of two .npy files: one file contained the training examples that represented images of different ASL characters, while the other file contained the corresponding labels. As part of preprocessing, some images were removed from the dataset, either due to poor quality or incorrect labeling. For the data splitting scheme, we opted to use 80% of the data for training, while 20% was allocated to the validation set. The test set consisted of a separate dataset that was procured through additional data collection and was not obtained through a split of the original dataset. Data augmentation schemes used included the use of additional data obtained through additional images procured through the aforementioned data collection process and transformations, which included rotation and color transformations.

For classification tasks, various deep CNNs have seen extensive usage in the past, where increasingly complex datasets have benefited from deeper networks. For example, VGG nets have seen usage for the ImageNet and CIFAR-10 datasets, both of which are considered cornerstones of basic image classification schemes. At the same time, however, a dilemma arises when considering the number of layers that should be stacked to produce a more complex and capable model. It has been found that a "degradation" problem becomes more acute as increasing convolutional layers are stacked, whereby accuracy of the training and test error begins to rapidly increase, leading to a significant drop in accuracy. As such, shallower neural networks can often produce no higher performance than their deeper counterparts.

To remedy the degradation problem, our work implemented the ResNet-18 architecture, which makes use of residual blocks and contains a total of 18 layers consisting of 3x3 and 1x1 convolutions. The input is eventually convolved to 512 features with the increased number of filters. Residual connections allow for the activation from one layer to be fed to another, deeper layer, which serves as the building block for the ResNet architecture. Based on past works, it has been shown that using residual connections allows for much deeper CNNs to be trained without suffering from generalization issues brought by the degradation problem. Previous uses of the ResNet architecture have demonstrated that it can be implemented to be up to eight times deeper than the VGG nets while still maintaining a lower number of parameters and superior generalization capabilities.

Section II of this work describes the architecture used, as well as hyperparameter tuning schemes and data augmentation techniques. Section III provides an overview of experimental results and a discussion of the network's performance, as well as possible improvements that could be made.

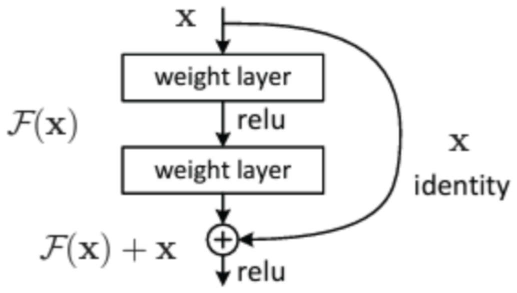
x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Fig. 1. Overview of residual learning. The identity function becomes easier to learn for a residual block, which means that adding more layers should not result in degradation seen with other nets without such a scheme. A deeper network without residual blocks will struggle to choose parameters even for the identity function, thereby leading to diminished performance.

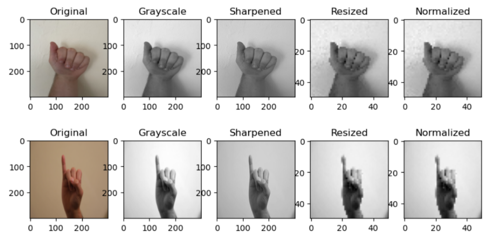

Fig. 2. Different stages of preprocessing. From RGB to normalized image

II. Implementation

We preprocessed input data before feeding data to our models. We use Residual Networks and a hybrid of CNN and FCN. Data augmentation is used to increase diversity, and generalization, reduce overfitting, and enhance robustness. In the later section, we give the details.

A. Data Preprocessing Pipeline

In the data processing, we convert the image RGB to grayscale so that the model can understand the pattern better and reduce the computational complexity.Then an image-sharpening kernel $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 6 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ is used to enhance edge visibility and improve image details. Finally, we converted the image to a 50x50 size and normalized the data (0-1) as the network works better on normalized pictures. Figure 2 shows the process.

The data set is then organized into the custom format for Pytorch.

B. Data Augmentation

Data augmentation is used mainly to increase diversity and make the model to learn about different types of images. As collected data has different variances like lighting condition, zooming factor, shifting factor, rotation factor, etc. Based on these conditions, we have decided to augment our database. We have applied four augmented techniques to the dataset. They are rotation, width shift, height shift, and shear. We use these variables alone or altogether randomly(sometimes any two variables altogether, sometimes three and sometimes four) on an image and create three different images from one image. Figure 3shows the process.

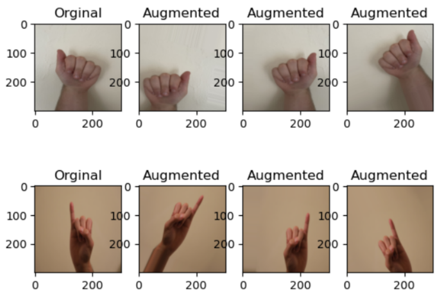

Fig. 3. Showing the augmented pictures from original pictures.

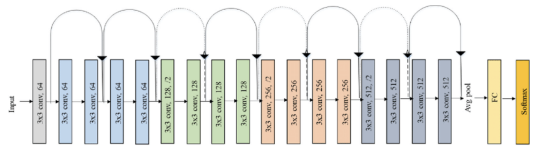

Fig. 4. ResNet-18 architecture used in this paper. In this architecture, a total of 18 layers are implemented: 17 convolutional layers and one fully-connected layer. The convolutional layers are organized into residual layers, where each residual layer contains multiple residual blocks. .

C. Proposed Network Architecture

The implementation of the ResNet architecture[3] centers around a series of 3x3 and 1x1 convolutional layers, with each residual layer composed of multiple residual blocks and max pooling layers used to downsample the image dimensions. A dropout layer, where the probability of a given activation to be zeroed is 0.2, is included after every residual layer. A residual block, as implemented in our architecture, consists of two consecutive 3x3 convolutional layers accompanied by batch normalization and Rectified Linear Unit (ReLU) activation functions. Skip connections with a stride of 2 implemented at the first residual block of every other layer. The 3x3 convolutions focus on extracting important features from the input. The function of the 1x1 convolutional layers is to act as "bottleneck" layers; that is, using a convolution with a 1x1 filter reduces the number of channels of the input such that subsequent convolution operations are more computationally efficient by reducing the number of parameters while maintaining the dimensions of the current input. The final layer comprises an adaptive average pooling layer and a fully connected layer, which will output the class probabilities for the given training example given the multi-class classification problem. Figure 4 shows the architecture of Resnet.

D. Objective Function

The categorical cross-entropy loss is used as the objective function for the architecture.

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

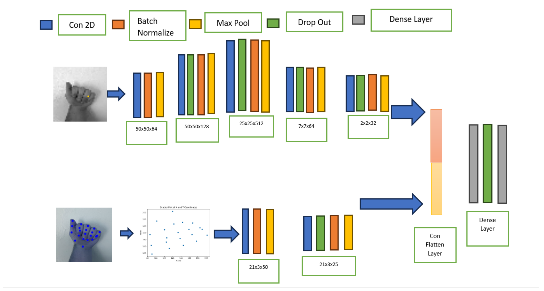

Fig. 5. Picture is showing multithreaded CNN.On one head preprocessed image is fed where as on another head google hand land mark is fed.

E. Training Scheme

The mini-batch optimization technique has been used with a batch size of 32. Adam optimizer [2] is used with an initial learning rate of 0.0001. For traning, a100 GPU has been used. Number of epochs used for training is 50 with a patience of 10. We have also used dropouts after every layer in our resnet with a rate of 20%.

III. EXPERIMENTS

A. Evaluation Metrics

In this project, the performance of our resnet implementation has been evaluated using the metrics of ROC. We have calculated the number of accurately classified samples in the validation set for each class. Accuracy focuses on instances of true positives and negatives. We have also visualized the classification performance with a confusion matrix. The computational complexity of the model is also estimated in order to analyze and compare the performance of different networks.

B. Dataset

The total number of 8443 data was given. After the augmentation mentioned in section IIB we have increased our data to 33772. So before feeding the data to a model we split the data to 80 percent for training and 20 percent for validation.

C. Comparing Methods

We have used two different models for training and testing. They are ResNet and CNN. For CNN we have used multi-headed CNN with 3 FCN layers [1] from previous research. In this architecture, one head of CNN is used to feed a 50x50 preprocess resize image described in section IIA. Another head is used to feed Google hand landmark coordinates. After that, we have combined those in a flattened layer. Finally, we use 3 FCN layers to train the model. So in the CNN layer, we use Relu activation function, and last layer of FCN we use the Softmax activation function. Other details like CNN and FCN channels are given in figure 6.

For the ResNet architecture, we did not use any handmark land coordinates. We augmented the original data and fed the large data set into the ResNet model and trained with a sufficiently large number of epochs with some regulaization.

Model	Structure	Acc	Complex
Model-1 (Resenet)	7x7 CNN+ 4x2x3x3 CNN+ 1 FCN	98.8%	42.7MB
Model-2 (CNN+FCN)	5xCNN+2xCNN+3FCN	89%	17.43MB

TABLE I
COMPARSION TABLE.

D. Results

The initial CNN model achieved an accuracy of 89%. To acheive this accuracy, a significant amount of preprocessing and augmentation have been applied. For preprocessing, we have used hand tracking techniques to track the coordinates of the hands in the images. We then passed the images in the convolutional layer of the network along with their associated hand coordinates values. The model then learned the hand pixels values and along with the coordinates. This increases in the number of parameters for the model to learn proved to be effective. Without hand tracking, the model's performance reaches a threshold of around 75%. After hand tracking, the accuracy increases by 14%. Yet, the accuracy reaches a threshold with this network despite experiencing an increase in the input parameters. Even with data augmentation, the network does not improve further. This can be due to the poor quality or wide range of variable conditions present among the images. This leads to us to explore a more complex architecture such as ResNet.

The ResNet model achieved higher accuracy of 98.77% on the validation set after performing data augmentation. Before augmentation, the accuracy also reached a threshold of 92% for this network. We have collected a set of small test images and the model's performance on that was only 89%. Thus, our goal was to bring the validation accuracy by around 4-5%. We have applied PCA and SVD to reduce the size of the data and select the most prominent features for the network to train on. Although around 4100 vectors of size 270000 explained around 90% variance, this method failed to increase the accuracy. Reducing the size of the images to 100X100X3 seemed to deteriorate performance. We have tuned the hyper-parameters and added several regularization techniques which minimally increased the performance. Ultimately, applying data augmentation to get a large set of images and feed those to this robust architecture significantly increased the accuracy on validation and test sets.

Table I shows the comparisons of the two models in terms of the number of parameters they used and the accuracy they obtained. In figure 6, we show the confusion matrix we obtained for ResNet. We see that the most number of misclassifications occurred for class 6. However, the percentage of misclassification error is so small that it cannot be concluded that the model is performing poorly for that class. The model performing better for some classes than others can change with different random splits of the data. Regardless of the splits, the confusion matrix mostly shows bright diagonals with a large number of samples being correctly classified.

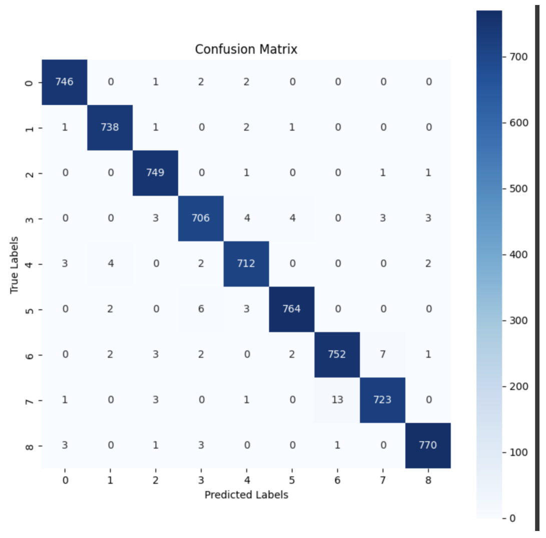

Fig. 6. Confusion matrix of the Resnet model

IV. Extra Credit

We take the probability matrix from our resnet model. If any class is determined by the network with a probability less than 0.5 we consider it as negative class.

Conclusion

Sign language serves as a crucial means of communication for individuals who are deaf and mute. The accuracy of sign language recognition is paramount for effective communication. In this study, our focus revolves around the classification of nine distinct characters within sign language. To enhance the robustness of our model, we implement data preprocessing techniques to mitigate noise, and data augmentation methods to increase the size of the dataset, thereby improving the model's generalization. Upon identifying the limitations of conventional CNN networks and even multiheaded CNNs, we try a ResNet architecture. Remarkably, our ResNet model achieves an impressive accuracy of 98.8 %, accurately classifying each letter with near perfection. This outstanding result highlights the efficacy of the ResNet network in addressing the challenges faced by other architectures.Looking ahead, our future research endeavors will explore advanced image processing techniques to further enhance the capabilities of our system.

References

1. Pathan, R.K., Biswas, M., Yasmin, S. et al. Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network. Sci Rep 13, 16975 (2023). https://doi.org/10.1038/s41598-023-43852-x

2. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

3.He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.