# GROUP24: SIGN LANGUAGE CLASSIFICATION USING DEEP LEARNING

*Minting Chen, Tianyi Gao, Xiaoyang Pan*

University of California San Diego, La Jolla, CA 92093-0238

## ABSTRACT

**Sign language is a useful tool for people with listening and speaking difficulties. In this paper, we used deep learning methods for sign language recognition to obtain comparable results with limited training data. We build three models to perform the task, Convolutional Neural Network (CNN), VGG16, and Resnet50. We also investigate ways to improve the model performance like data augmentation, batch normalization, merge datasets, etc. Overall, ResNet-50 has the worst performance whereas the CNN model performs the best with $99.65\%$ accuracy on ASL dataset. The VGG-16 learning also shows some success on the classification task via transfer learning.**

## 1. INTRODUCTION

Sign language can be a very effective way to communicate with people who have hearing or speaking difficulties. American Sign Language (ASL) is a natural language that is widely used around the world. Since sign language is a skill that need takes time to learn, it is not well understand by the general public outside the deaf community. Sign language recognition is a problem that has been studied for many years, however, we have not yet found a complete and suitable solution. At present, most of the proposed solutions to this problem involve wearing sensors or applying a vision-based device such as a camera. Although the accuracy of recognizing the sign language is high by wearing sensors for recognition, the equipment is still expensive and not easy to carry. With the advent of deep learning, datasets can make better use of models and improve accuracy by entering images. In this paper, we compare the performance of three models with the image datasets as input.

## 2. RELATED WORK

American sign language (ASL) classification is not a novel topic; it has been studied for over two decades. As mentioned in [1], there are three main methods for solving the classification problem: Bayesian Classifier, Linear Classifier, and Neural Networks. In 1997, Starner, Thad, and Alex Pentland [2] used the Hidden Markov Model and a pair of gloves to captures 3d motion of the hand. Then they fed the spatial information from the gloves over time into the model. It was able to achieve $99.2\%$ accuracy. In 2013, Sharma, Rohit, et al. [3] used support vector machine and K-Nearest Neighbor classifier after background subtraction and noise removal on the input image. This model achieved $90.1\%$ accuracy. In 2016, Garcia, Brandon, and Sigberto Alarcon Viesca [1] utilized a pre-trained GoogLeNet model to do ASL classification and achieved around $70\%$ accuracy for five of the letters. They claimed that the poor performance was due to a lack of data. However, the paper showed some success in classifying with a pre-trained model. In 2018, Huang, J. [4] proposed Hierarchical Attention Network with Latent Space (LS-HAN) which does not require any preprocessing and segmentation on the input images. This is the state-of-the-art method and its model contains Convolutional Neural Network (CNN) layers for generating feature representations, a Latent Space (LS), and a Hierarchical Attention Network (HAN). As the result, LS-HAN was able to achieve $61.6\%$ accuracy.

In this project, we are re-implementing the Convolutional Neural Network (CNN) mentioned in [5]. The authors first did background subtraction on the data and data augmentation on the train set by rotating the images by 20 degrees and flipping the images horizontally. The CNN model described in the paper has a simple architecture, which only contains several convolutional blocks and dense layers. And this model was able to achieve $82.6\%$ accuracy on the test set, which is a good result for such a simple model without complicated architecture and the use of transfer learning.

## 3. DATASET

Two datasets utilized in this projects, American Sign Language(ASL) [6] Dataset, and Sign Language Gesture Images(SLGI) Dataset [7] are from Kaggle, "Interpret Sign Language with Deep Learning" and "CNN on Sign Language Gesture Images dataset" respectively. They consist of images of 37 classes, alphabet A-Z, and the number 0-9. ASL dataset has $20216$ images, $400 \times 400$ pixels, and SLGI dataset has $55500$ images, $50 \times 50$ pixels. Images from two datasets are shown in Figure 1.

There is no feature selection process since the CNN models can identify the most important features for classification. To reduce the training time and same input size for the CNN model, images from the ASL dataset are resized to $50 \times 50$ pixels, and the RGB values of each image are normalized, di-

**Fig. 1**: Images from ASL and SLGI datasets. Top two rows are images from ASL dataset [6], and bottom two rows are images from SLGI dataset [7].

vided by 255. To include both left and right-hand gestures in our model, we flip train images horizontally. After data augmentation, the ASL dataset has 40432 images, and the SLGI dataset has 111000 images. All codes are implemented using Python and Tensorflow. Both datasets are divided by $8:2$ into the training set and the test set. Based on the training set, the verification set is divided by $8:2$.

## 4. METHODS

### 4.1. CNN Model

The method/model we implement in this section is the CNN model discussed in [5] as shown in Figure 2. It contains three convolutional blocks and two groups of dense layers.

**Convolutional Block** is widely used for classification tasks. It can learn different features by sampling with filters on the input image. Each block has two convolutional layers followed by a maxpool layer and a dropout layer.

**Batch Normalization** helps prevent the model from overfitting and improve the performance. We add a batch normalization layer after each convolutional layer and dense layer of the CNN model shown in Figure 2.

**Max Pool** is used to reduce the feature map size by downsampling.

**Dropout** layer makes the model more robust and less overfit. During training time, a certain percentage of the nodes will be dropped. However, the use of dropout layers makes the model harder to do predictions during training time, which results in higher training loss.

**Activation Functions.** Activation function, which is used after each convolution and dense layer, defines the output of a neural based on the input value. In this project, we use two activation function. **ReLU** activation function defines the positive part of the input. The ReLU function is defined as

$$f(\boldsymbol{x}) = max(\boldsymbol{x}, 0),$$

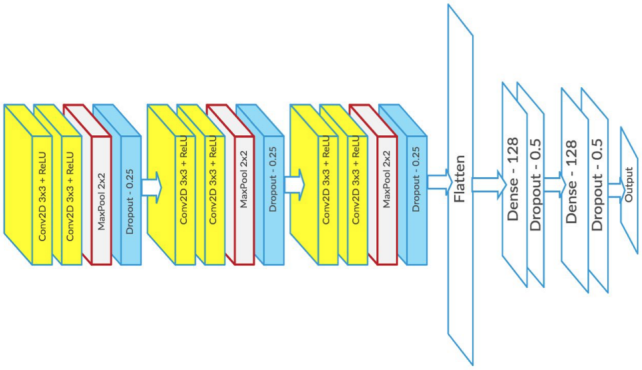


**Fig. 2**: CNN model architecture without batch normalization from [5].

where $\boldsymbol{x}$ is the input value. **Softmax** activation function is commonly used for multiclass classifier, which outputs the probability distribution of all possible classes. The softmax function is defined as

$$softmax(\boldsymbol{z})_i = \frac{exp(z_i)}{\sum exp(z_j)},$$

where $\boldsymbol{z}$ is the input vector represents the unnormalized log probabilities.

### 4.2. Transfer Learning - VGG16 Model

Transfer Learning utilized storing information from another model to train for a new and similar problem. A pre-trained VGG16 model learns from over 14 million images, which contains 1000 classes. Then, the information can be utilized with the ASL and SLGI datasets to make predictions.

VGG16 model is a more complex version of the CNN model described above. It contains five convolutional blocks without the dropout layers. Then, one flatten layer is followed by a dense layer to output the prediction value.

### 4.3. Transfer Learning - ResNet50 Model

The full name of ResNet is residual network. It refers to the VGG19 network and is modified on the basis of it. ResNet adds the residual block through the shortcut connection[8]. Figure 3 is a block proposed for the deep network, and it is called a "bottleneck" block, whose main purpose is to reduce dimensionality. The three layers are 1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions. First, reduce the 256-dimensional channel to 64 channels through a 1x1 convolution, and finally restore it through a 256-channel 1x1 convolution[8]. As we can see here from

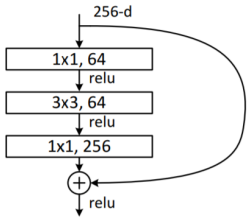


**Fig. 3**: Residual block from [8]

| layer name | output size | 18-layer | 34-layer | 50-layer | 101-layer |
|---|---|---|---|---|---|
| conv1 | 112×112 | 7×7, 64, stride 2 | | | |
| | | 3×3 max pool, stride 2 | | | |
| conv2_x | 56×56 | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$ |
| conv3_x | 28×28 | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ | $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$ |
| conv4_x | 14×14 | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$ | $\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$ |
| conv5_x | 7×7 | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$ | $\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ | $\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$ |
| | 1×1 | average pool, 1000-d fc, softmax | | | |
| FLOPs | | $1.8\times10^9$ | $3.6\times10^9$ | $3.8\times10^9$ | $7.6\times10^9$ |

**Fig. 4**: Architectures of ResNet from[8]

the table in Figure 4, for the 18-layer and 34-layer ResNet, we replace each 2-layer block in the 34-layer net with this 3-layer bottleneck block, resulting in a 50-layer ResNet. The three-layer convolution kernels are what we see here. It is worth noting that the number of feature maps in the hidden layer is relatively small, and it is 1/4 of the number of output feature maps.

## 5. EXPERIMENTS

### 5.1. CNN Model

**Training Details.** The convolutional layers in the first two blocks have 32 filters and the ones in the third block have 64 filters. Kernel size is $3 \times 3$ and stride size is 1. After these three blocks, we reduce the size of the feature map to $2 \times 2 \times 64$. We use add batch normalization because it helps minimize the gap between validation loss and training loss. A dropout layer is used at the end of each convolutional block with a $0.25$ drop rate and each dense layer with a $0.3$ drop rate to prevent overfitting. Relu is used as the activation function for all convolutional layers and softmax is used in the output layer. We use cross-entropy as the loss function and adam optimizer with a learning rate of $0.001$ to prevent the model from learning too quickly or too slowly. The batch size is 32 to minimize generalization error. We train the CNN model with the ASL dataset for $100$ epochs and train the same model but with the merged dataset (ASL+SLGI) for $50$ epochs.

**Experiment on CNN Model.** To investigate whether data augmentation improves our model performance, we compare the models that are trained by the ASL dataset and the augmented version of the ASL dataset, where each image in the train set is flipped horizontally. In addition, we add batch normalization to further boost the model performance. The accuracy and loss plots are shown in Table 1. The accuracy decreased by $1\%$ after augmentation because there are no left-hand gesture images in the test set, so those additional images may make the model harder to do predictions. However, batch normalization help solves the issue. It boosts the accuracy to $99.65\%$ and reduces the loss to half.

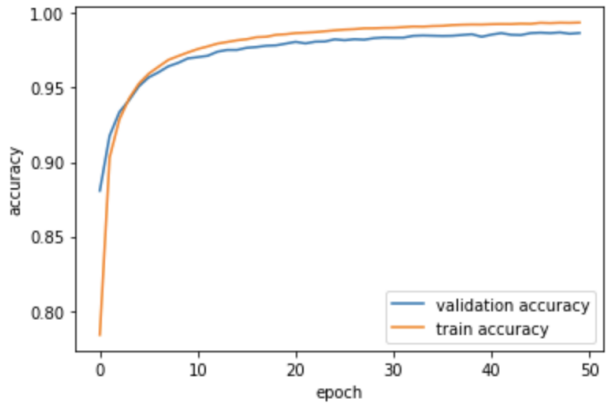


**Fig. 5**: Accuracy of VGG-16 Model with the merged dataset (ASL+SLGI)

### 5.2. Transfer Learning - VGG16 Model

**Training Details.** All convolutional layers in the VGG16 model have a kernel size of $3 \times 3$, and a stride size of 1. There is no dropout layer and batch normalization. Same activation function, loss function, and optimizer as the CNN model. The batch size is set to 64. We train the VGG16 model with the ASL dataset for 100 epochs and train the same model with both datasets for 50 epochs.

**Experiment on VGG-16 Model.** The VGG16 model presents a high accuracy when train and predict using images from the ASL dataset. However, the model might not be very general to all kinds of images. Therefore, we merged the ASL dataset and the SLGI dataset to make our model even more robust. We choose the VGG model for this experiment because it has better performance than the ResNet50 model and it is more efficient than the CNN model. Figure 5 and Figure 6 show the accuracy and loss, respectively. The simplified confusion matrix is shown in Figure 7. The model performs well on classifying the gestures, and the number of misclassification is small compared to the number of correct classifications. The loss curve implies no sign of overfitting. The new testing accuracy is 0.97.

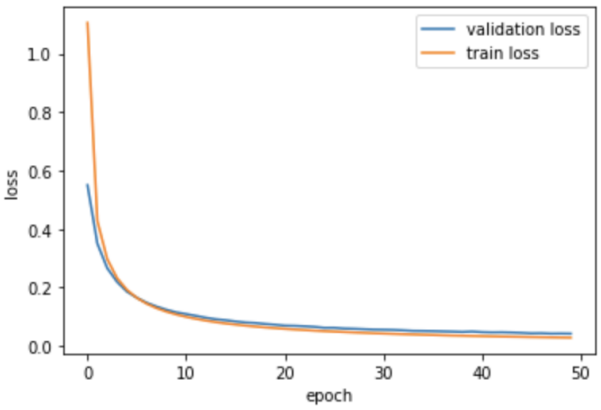


**Fig. 6**: Loss of VGG-16 Model with the merged dataset (ASL+SLGI)

### 5.3. Transfer Learning - ResNet50 Model

**Training Details.** ResNet50's pre-training model on ImageNet was used for training. The training batch size was 64 and we trained 100 epochs. ResNet50 was built without the original fully connected network. The model retains the last 8 layers for training. The optimization algorithm used in the training process is adam, and the Loss function is "categorical cross-entropy".

**Experiment on ResNet Model.** Again, we perform the augmentation experiment on the ResNet model. The accuracy and loss value on the test set is shown in Table 1. Data augmentation is less effective than it without data augmentation because, in the training process, the model without augmentation has higher accuracy than it with the data augmentation at the initial stage of training. From the final result, there is a higher accuracy in the validation set without Augmentation. This may be because images differ significantly from the original image after augmentation, leading to a decrease in model accuracy.

### 5.4. Discussion

We use accuracy as our primary metric. The testing accuracy and loss for each model are shown in Table 1. The CNN model with Data Augmentation and Batch Normalization yields the best accuracy and lowest loss. On the other hand, VGG16 also yields high accuracy and low loss. A potential problem with the CNN model is that it might only fit a dataset similar to ASL, while VGG16 might perform better at predicting datasets other than ASL through information from the pre-trained model. In comparison, ResNet50 has low accuracy comparing to the other models. Therefore, VGG16 and CNN models could be used for a more general model in the future.

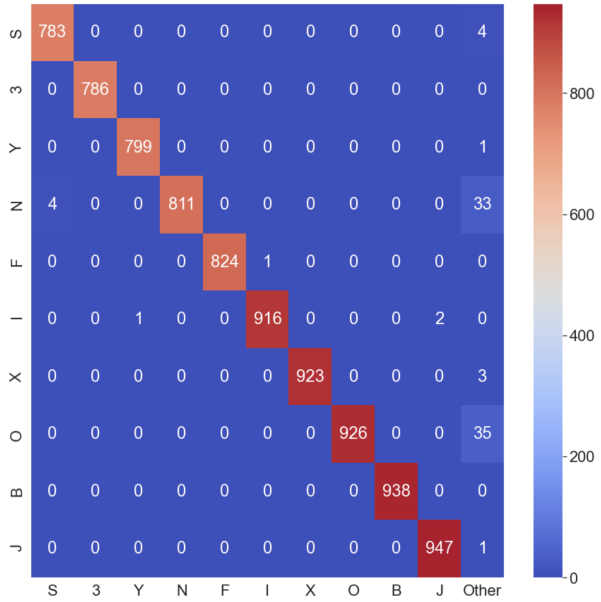


**Fig. 7**: Confusion Matrix of VGG-16 Model with the merged dataset (ASL+SLGI). Top five rows represent the five classes with the highest accuracy. Bottom five rows represent the five classes with the lowest accuracy. Right most column represents the total number of misclassification of all other classes.

| Model | Loss | Accuracy |
|---|---|---|
| CNN w/o Data Augmentation | 0.0233 | 0.9921 |
| CNN w/ Data Augmentation | 0.0448 | 0.9813 |
| CNN w/ Data Augmentation + BN | **0.0142** | **0.9965** |
| VGG16 w/o Data Augmentation | **0.0510** | **0.9854** |
| VGG16 w/ Data Augmentation | 0.0979 | 0.9699 |
| ResNet50 w/o Data Augmentation | **0.7434** | **0.8823** |
| ResNet50 w/ Data Augmentation | 0.7973 | 0.8733 |

**Table 1**: The model performance w/o and w/ data augmentation on ASL dataset. (BN: Batch Normalization)

## 6. CONCLUSION

In this project, we implement the CNN model, VGG-16, and ResNet-50 to classify the American Sign Language gestures. For all three models trained with the ASL dataset, data augmentation actually degrades the performance. However, the CNN model with additional batch normalization layer shows some success on recovering the loss from data augmentation. Overall, the CNN model achieve the highest score on testing the ASL dataset. To make our model more robust, we combine the ASL dataset with the SLGI dataset and train the VGG model with the merged dataset. As the result, we are able to achieve 97% accuracy on the test set, which include both hands gestures, and various lighting condition and skin colors.

## 7. CONTRIBUTIONS

Since we analyze three models in this project, each team member work on one model implementation which includes loading data, implementing model architecture, training, and testing, and documenting the results. Minting Chen is responsible for the CNN model, Xiaoyang Pan is responsible for the VGG-16 model study, and Tianyi Gao is responsible for the ResNet-50 model. In this final report, Tianyi Gao wrote the abstract and introduction sections; Minting Chen wrote the related work and conclusion sections; Xiaoyang Pan wrote the dataset and discussion sections. Then each of the members writes an analysis of the corresponding model and experiment results.

## 8. REFERENCES


[1] Brandon Garcia and Sigberto Alarcon Viesca. Real-time american sign language recognition with convolutional neural networks. *Convolutional Neural Networks for Visual Recognition*, 2:225–232, 2016.

[2] Thad Starner and Alex Pentland. Real-time american sign language recognition from video using hidden markov models. In *Motion-based recognition*, pages 227–243. Springer, 1997.

[3] Rohit Sharma, Yash Nemani, Sumit Kumar, Lalit Kane, and Pritee Khanna. Recognition of single handed sign language gestures using contour tracing descriptor. In *Proceedings of the world congress on engineering*, volume 2, pages 3–5, 2013.

[4] Jie Huang, Wengang Zhou, Qilin Zhang, Houqiang Li, and Weiping Li. Video-based sign language recognition without temporal segmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[5] Vivek Bheda and Dianna Radpour. Using deep convolutional networks for gesture recognition in american sign language. *arXiv preprint arXiv:1710.06836*, 2017.

[6] American Sign Language Dataset. https://www.kaggle.com/ahmedkhanak1995/cnn-on-sign-language-gesture-images-dataset.

[7] Sign Language Gesture Images Dataset. https://www.kaggle.com/paultimothymooney/interpret-sign-language-with-deep-learning.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Group 6:
Comments: You mention that the way to improve the Resnet50 results would be to make the dropout higher, but I think there are a variety of other things applicable? Batch size/ learning rate / freezing / optimizer
Reply: Making the dropout higher is one of the way to improve the model. I just use it as an example when we do the presentation.The other ways you said also work for the optimization.
Comments: You mention that using the other low lighting data set would be 'very different' from ASL dataset. Maybe it would be helpful to be more specific. It would the new dataset be harder or easier to classify?
Reply: Please refer to the data section for more details on the new dataset we used. In general, the new dataset is harder to classify due to various lighting and skin colors.
Comments: A lot of the plots in the presentation just show train vs val. What about plots of the results vs the test?
Reply: the plots showing validation and training loss is to evaluate the model and tuning the hyperparameters. After tuning the model, we should see how the model work, and we think compare the plots with accuracy might not give much meaningful information. Therefore, we do not include it in the report.
Comments: Why not also train data augmentation with ResNet 50?
Reply: We have trained data augmentation with ResNet50, maybe I didn't mention it clearly.
Comments: Overall I like the 3 model approach.. I think it would be nice to have other metrics besides accuracy, maybe precision/recall? Also a final figure showing how each of the 3 models compared to one another would be nice too.
Reply: The precision/recall value are very closed to accuracy, and compare accuracy and loss should be sufficient to describe the performance of the model. In our report, we add the section to compare all three models. Thank you for pointing out.

Group 9:
Comments: Why in the CNN model, the train loss is higher than the validation loss?
Reply: Please refer to the explanation for dropout layer in section 4.1.
Comments: What's the number of the data after data augmentation? Did you rotate all the data or just some data?
Reply: We only do augmentation on the train set; so each train image is flipped horizontally. The train set size is doubled.
Comments: I see that it also contains testing data, so what's the prediction accuracy of the testing data by the 3 mentioned models?
Reply: The testing accuracy is shown in the table of the experiment section.
Comments: I'm not familiar with transfer learning. Can the setting or parameters be changed in the pre-trained model? If so, maybe you can try to get some improvements with the ResNet50 model.
Reply: Before the presentation, I kept the last four layers for training, after that I kept the last eight layers for optimizing the model, it has more learning ability than keeping 4 layers.
Comments: Will you guys think the feature extraction will improve the work?
Reply: The convolutional neural networks are able to identify the most important features from the images. We also had try to convert the images into black and white images to obtain the edges of hand, but the training on the black and white images presents poor accuracy. Therefore, We did not include any feature extraction process.

Group 15:
Comments: What is the reason for choosing a current variable combination, such as learning rate, discount rate, and initial conditions?
Reply: Please refer to section 5 where we give detail explanation on that.
Comments: Since you splitted the data to train, validation and test, it would have been useful to also report the test loss and see how those models perform on test data.
Reply: Please refer to the table that shows the loss and accuracy on the experiment section.
Comments: Could have used different models like ResNet101 with pre-trained weights to have a better accuracy?
Reply: I agree. ResNet101 will perform better.
Comments: How exactly does your model compare to the previous works?
Reply: Our CNN model architecture is very similar to the one implemented in [**?**]. However, we changed some parameter setting. We add batch normalization and changed the dropout ratio. Based on our result, our model could perform better and achieve higher accuracy. In addtion to that, we also investigate two other models, VGG-16 and ResNet-50, in this project.