# ML 2021 Final Project Report

Juhyeon Park
Seoul Nati' Univ
1 Gwanak-ro, Seoul
parkjh9229@snu.ac.kr

## Abstract

*In this project, I implemented an alphabet recognition and sorting model with **CRNN** architecture, which combined CNN and RNN. I referenced the ResNet-18 model for CNN and used the LSTM cell for RNN.[1][2] I could get a test accuracy of 87 percent with a given test set. To alleviate the over-fitting problem, I adopted the "Voting Ensemble" approach.[3] With the ensemble approach, I could get a test accuracy of 89 percent with the same test set, which is an improvement of about 2 percent.*

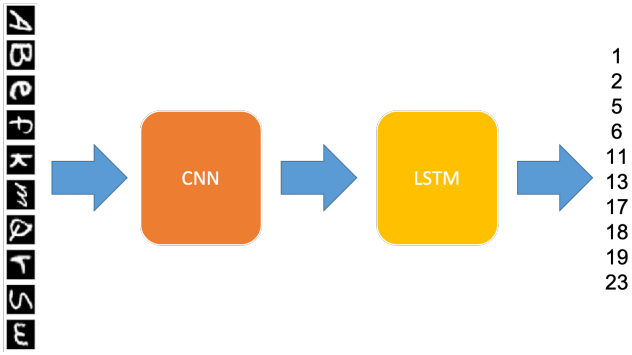


Figure 1. Overall Learning Flow Diagram

## 1. Introduction

I thought that this project can be divided into 2 big problems. The first one is the recognition of a single alphabet from EMNIST dataset.[4] To solve this problem, I could build CNN architecture, which is similar to the ResNet model with skip connection. The second problem is sorting those recognized alphabets. I used the LSTM cell to solve this problem. Combining these two separate models, I made a united model named **ConvLSTM**.

And then I adopted ensemble learning to improve validation accuracy. I used Voting Classifier, which trains multiple models and averages their outputs, because I observed tendency of over-fitting.

### 1.1. CNN Architecutre With Skip Connection

The ResNet architecture significantly improves the performance of CNN by using residual blocks and skip connection.[1] Especially, the skip connection facilitates the transfer of gradients from errors. I adopted that concept when building the CNN architecture in ConvLSTM.

### 1.2. LSTM & Teacher Forcing

The LSTM(Long Short Term Memory) is commonly used in RNN architecture.[2] The LSTM cell consists of several gates to coordinate the effects of data from the previous step and the current step. To accelerate the speed of learning, I used the teacher forcing technique, which feeds LSTM cells with true labels. However, there can be a huge discrepancy between training accuracy and validation accuracy if true labels are given all-time when the model is trained. So, I decided to use the teacher forcing with 50% probability in this project.

### 1.3. Ensemble Learning

Ensemble learning is the method of training multiple models to reduce bias or variance of the model. As I mentioned earlier, I could observe the tendency of over-fitting when I trained the ConvLSTM model. Therefore, I searched the ensemble method which trains multiple models in parallel. I used *VotingClassifier* supported by *torchensemble* package.

I used 10 estimators whose base model is ConvLSTM.

### 1.4. Flow of Learning

1-channel, $28 \times 28$ EMNIST dataset is given to the input of the CNN architecture. Output shape of CNN model is (*Sequence Length*, *Batch Size*, *Hidden Dim*). Then the output is fed to the LSTM cell to make outputs. Overall diagram is in Figure 1.

| layer name | output size |
|---|---|
| CONV1 $3 \times 3$ | $16 \times 28 \times 28$ |
| BN1 | $16 \times 28 \times 28$ |
| Res1_CONV1 $3 \times 3$ | $16 \times 28 \times 28$ |
| Res1_BN1 | $16 \times 28 \times 28$ |
| Res1_CONV2 $3 \times 3$ | $16 \times 28 \times 28$ |
| Res1_BN2 | $16 \times 28 \times 28$ |
| Res2_CONV1 $3 \times 3$ | $32 \times 14 \times 14$ |
| Res2_BN1 | $32 \times 14 \times 14$ |
| Res2_CONV2 $3 \times 3$ | $32 \times 14 \times 14$ |
| Res2_BN2 | $32 \times 14 \times 14$ |
| Res3_CONV1 $3 \times 3$ | $64 \times 7 \times 7$ |
| Res3_BN1 | $64 \times 7 \times 7$ |
| Res3_CONV2 $3 \times 3$ | $64 \times 7 \times 7$ |
| Res3_BN2 | $64 \times 7 \times 7$ |
| Flatten | (*Batch Size* $\times$ *Seq Length*, 3136) |
| FC | (*Batch Size* $\times$ *Seq Length*, 64) |

Table 1. Summary of CNN Model

## 2. Experimental Evaluation and Analysis

I tested my model with the given validation set and test set. I observed the tendency of training with validation accuracy and tuned several hyper-parameters based on results.

### 2.1. Model Architecture

When I first implemented ConvLSTM architecture, I could observe a big difference between training accuracy and validation accuracy (about 20%). I thought that the ResNet-18 architecture is too complex for a given task, which leads to over-fitting. So I tried to simplify my CNN model by reducing the number of convolution layers. Table 1. is the final CNN model description.

I used LSTM cells for the RNN part whose hidden dimension is 128 and I made RNN architecture deeper by setting the number of layers to 3. No additional option is not added to LSTM cells such as *bidirectional* argument.

### 2.2. Hyper-parameter Tuning

Initially, I used *Adam* optimizer with a learning rate of 0.001. As it works well, I did not try to change the optimizer or learning rate. Rather, I used a scheduler to decrease the learning rate as training proceeds. I used *ReduceLROnPlateau* scheduler for non-ensemble model and used *CosineAnnealingLR* scheduler for ensemble model.

To alleviate the over-fitting, I used *weight_decay* parameter in the optimizer. I tried using several values such as 5e-4, 1e-3. After comparing their training results, I decided to use 5e-4. Also, I used dropout for the same purpose. I inserted a dropout layer before FC layer in CNN and gave a dropout value of 0.5 to LSTM cells.

| Hyper-parameter | Value |
|---|---|
| Learning Rate | 0.001 |
| Batch Size | 16 |
| Epoch | 25 |
| Weight Decay | 5e-4 |
| Dropout in CNN | 0.4 |
| Dropout in LSTM | 0.5 |

Table 2. Summary of Hyper-parameters' Values

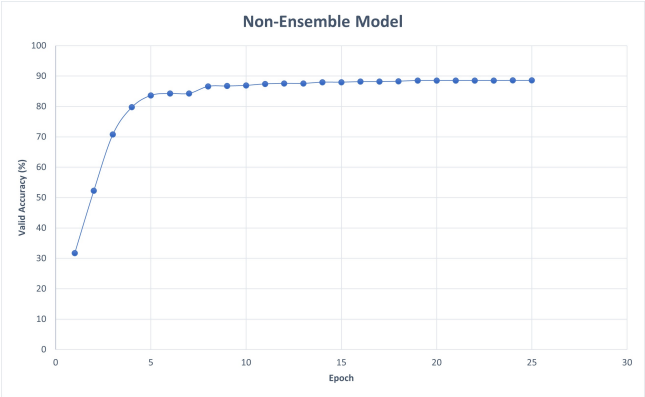


Figure 2. Validation Accuracy of Non-ensemble Model

Next, I considered batch size. As the size of the training set is 50,000, I considered the batch size among divisors of 50,000 to make each batch's size equal. The first batch size I chose was 100. As the total number of batches is relatively low, the training time was short and convergence speed was high. However, the validation accuracy didn't improve at 87 88 percent and the difference between training accuracy and validation accuracy was about 10 percent. Therefore, I decided to reduce the batch size. I experimented with the batch size with 4, 8, 16, etc. With a batch size of 4 or 8, the training time was too long and it needs more epochs to converge. I thought it was inappropriate for me to try numerous combinations of hyper-parameters. Considering all aspects, I decided to use a batch size of 16.

I considered using data augmentation as well but I didn't use it because the data augmentation technique I can apply is limited. As the training set consists of alphabet letters, I can't do flipping or tilting with huge degrees. So I thought the effect of data augmentation is not significant.

The last option I considered is the number of an epoch. The first epoch I chose was 15 but it was insufficient for the model to converge. I experimented with different epochs considering other hyper-parameters such as batch size.

Hyper-parameters I used is summarized in Table 2.

### 2.3. Result Of ConvLSTM Training

Figure 2 is the validation accuracy of non-ensemble model. I could get validation accuracy of 88.56% with 25 epochs and test accuracy of 87.54%.

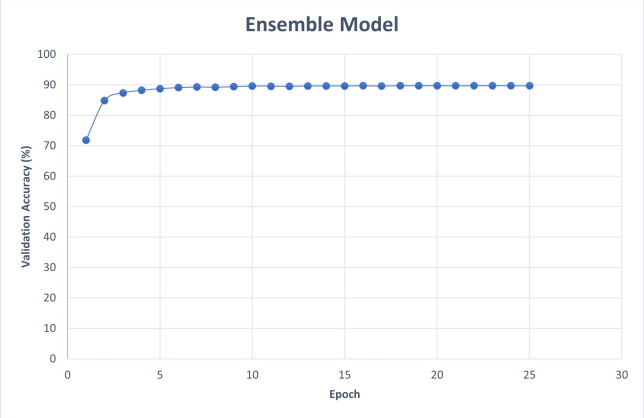


Figure 3. Validation Accuracy of Ensemble Model

### 2.4. Result Of Ensemble Learning

Figure 3 is the validation accuracy of ensemble model. I could get validation accuracy of 90.218% with 25 epochs. Also, the test accuracy of this model was 89.461%. This result shows that ensemble learning helps to deal with the over-fitting problem.

### 2.5. Analysis

First, the non-ensemble model itself was trained fairly well. Its test accuracy peaked 87%. But it can be thought relatively low comparing with its training accuracy. So, I tried using ensemble learning and it was successful. Its test accuracy peaked 89% improved by 2%. Therefore, I can conclude that ensemble learning was effective for alleviating the over-fitting problem and improving validation & test accuracy.

However, there was still a discrepancy between training accuracy and validation accuracy. For the ensemble model, training accuracy peaked 97% while its validation accuracy peaked only 90%.

## 3. Conclusion

I could observe that the choice of hyper-parameters is crucial to the improvement of the model's learning. Also, I could know that just tuning hyper-parameters can be a solution for over-fitting not using data augmentation. Also, I could observe that ensemble learning(parallel fashioned learning) is effective for curing the over-fitting problem.

## References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. Neural Comput. 9, 8 (November 15, 1997), 1735–1780.

[3] Zhou, Zhi-Hua. Ensemble Methods: Foundations and Algorithms. CRC press, 2012.

[4] Patricia.flanagan@nist.gov, "The EMNIST Dataset," NIST, 28-Mar-2019. [Online]. Available: https://www.nist.gov/itl/products-and-services/emnist-dataset. [Accessed: 22-Jun-2021].