# *Sign Language MNIST Dataset – CNN, Data Augmentation*

## 1. Data

Sign Language MNIST Dataset is image data, with labels and pixel values in single rows. According to Kaggle, the American Sign Language letter database of hand gestures represent a multi-class problem with 24 classes of letters (excluding J and Z which require motion). Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions). The training data (27,455 cases) and test data (7172 cases) are approximately half the size of the standard MNIST but otherwise similar with a header row of label, pixel1,pixel2….pixel784 which represent a single 28x28 pixel image with grayscale values between 0-255.

## 2. Create a CNN model and compile the model

For image classification, this time I used the CNN model rather than a fully connected model. I ran 3 models, the first one is the original CNN models, with Dense, Conv2D, MaxPool2D, Flatten, Dropout, and Batch Normalization. However, the result is not well, since there is a gap between loss in training and validation data shown in figure 1.

To solve the problem of overfitting, I created a model2, with a higher dropout rate and increased the dropout node. Figure 2 indicated its performance and prove that overfitting got solved since the gap between training data and validation data got smaller. In addition, I could keep training the model since the loss in validation data looks like will become smaller.

Another method to solve the overfitting in image classification is data augmentation. Compared to model 1, data augmentation did solve the problem. However, I cannot prove that it is better than the model 2. To decide a best model, we need to compare other factor, like accuracy in test data or our hardware's performance.

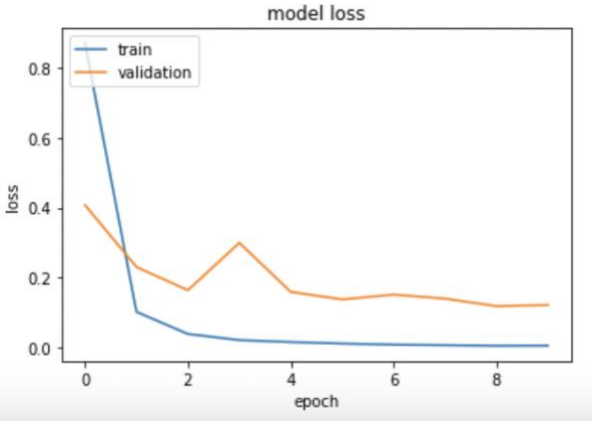


Figure 1. Loss of Model1

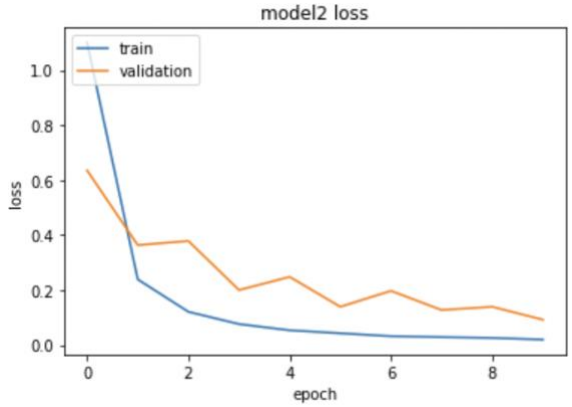


Figure 2. Loss of Model2

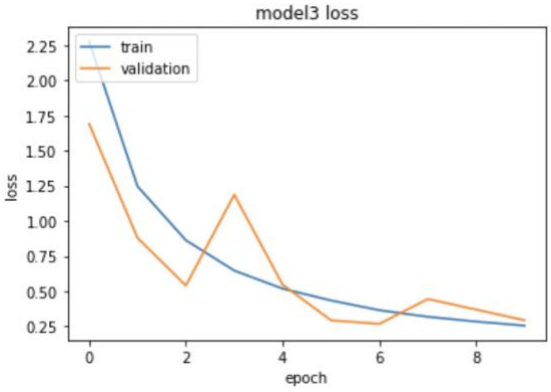


Figure 3. Loss of Model3

3. **Managerial Conclusion:**
   The project shows 2 methods to solve the overfitting problem. The first one is increasing dropout rate another is data augmentation.

| Loss / Model | Epoch Number of Best Model (Min. Validation Loss) | Train Data | Validation Data |
|---|---|---|---|
| Model1 (CNN) | 9 | 0.0056 | 0.1187 |
| Model2 (Increase Dropout) | 10 | 0.0204 | 0.0926 |
| Model3 (With Data Augmentation) | 10 | 0.3652 | 0.2679 |

Table1. The Minimum Loss in Different Models and Train and Validation Data