Machine Learning Assignment

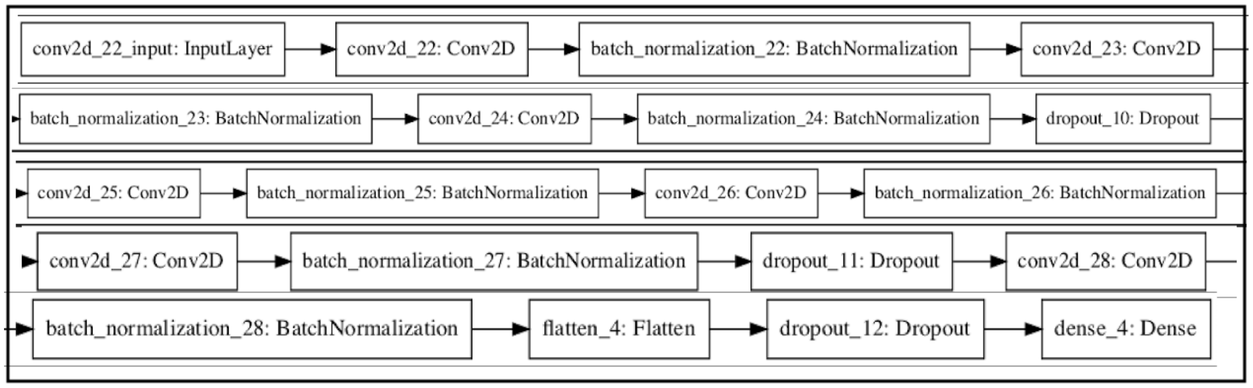
Group 49: Dennis Hokke (ANR: 589426), Aliona Codrean (ANR: 737109), Arati Sharma (ANR: 289062)

Task 1

Pre-processing: To prepare the data to be input into the model, we first checked to make sure that the data we had was consistent with what was described in the assignment. This included checking that there were the correct number of labels, the size of the dataset was correct. We also did some exploratory data analysis using visualizations to better understand the data. Furthermore, we added noise to the dataset to improve accuracy in task 2 by using s&p with the random noise function and setting it to 0.1. Furthermore, we attempted to use an ImageDataGenerator to intersect and overlap numbers and labels to try and improve the accuracy of intersecting characters and overlapping characters that we were unable to separate, however this was not very successful. We also found some labels that were incorrectly labeled in the dataset such as i was labeled as a 10 in some cases.

Model architecture: In order to do the classification, we used a 2DConvolutional Neural Network (CNN), as they have been found to be effective in multi-class classification tasks [1],[2],[3]. We tested various models with different combinations of number of layers, kernel sizes, including/excluding batch normalization and different regularization techniques. A schematic of the final model we selected is shown in Figure 1 and described below. The model consisted of 3 convolution 2D layers (size = 32), followed by 3 convolution 2D layers (size = 64), 1 convolution 2D layer (size=128), and a dense layer (size = 27) the last dense layer is 27 because the actual label values start at 1, whereas python starts at 0. After each of the convolution layers, batch normalization was applied. The third, sixth and seventh convolution 2D layers were also followed by a dropout function (probability=0.4). All of the convolution layers used the 'relu' activation function. The output from the convolution layers was flattened and then passed through the dense layer, which used the softmax activation function, to do the classification. This model performed the best among the other variations we tried (See Table 1 and Appendix A). The training and validation loss and accuracies generated by this model is shown in Appendix B

Figure 1. Model Schematic


conv2d_22_input: InputLayer
conv2d_22: Conv2D
batch_normalization_22: BatchNormalization
conv2d_23: Conv2D
batch_normalization_23: BatchNormalization
conv2d_24: Conv2D
batch_normalization_24: BatchNormalization
dropout_10: Dropout
conv2d_25: Conv2D
batch_normalization_25: BatchNormalization
conv2d_26: Conv2D
batch_normalization_26: BatchNormalization
conv2d_27: Conv2D
batch_normalization_27: BatchNormalization
dropout_11: Dropout
conv2d_28: Conv2D
batch_normalization_28: BatchNormalization
flatten_4: Flatten
dropout_12: Dropout
dense_4: Dense

Experiments: In order to train and evaluate the model, the training data was first split into train and test sets, with 80% and 20% of the data for training and testing, respectively. The train set was further split into train and validation sets, with 80% of the data used for training and 20% for validation. Before deciding on the final model parameters, we experimented with the hyperparameters, and compared the train, validation and test accuracies generated by the different configurations of the model. The results of these are shown in Table 1.

Table 1. Results of hyper-parameter tuning

Size of conv 2D layers	Batch norm	Epochs	Batch size	Train accuracy	Val accuracy	Test accuracy
1-3 = 32 4-6 = 64 7 = 128	Yes	5	64	**0.90**	**0.92**	**0.92**
1-7 = 64	Yes	5	64	0.90	0.92	0.91
1-3 = 32 4-6 = 64 7 = 128	Yes	5	128	0.90	0.91	0.91
1-3=4 4-6=8 7=16	yes	5	64	0.66	0.81	0.81

Task 2: We approached this task in three ways. In the first attempt (attempt 1), we used thresholding to remove noise from the images using the function remove_small_objects from the skimage package. For these new images, we used the label functions from skimage to label each object in the image, then followed that up with the find_objects function, which returns the coordinates of each object to use for extraction. Using these coordinates we extracted each object in the image and applied some reshaping and ran it through our model to obtain a prediction. The predictions were then stored and altered so that the output was a string that contained the true position of each letter in the alphabet.

In the second attempt (attempt 2), the images were upscaled and denoised by using median filters. We used findContours, a function from the CV3 library that uses contours to find the objects in the image. Although this performed much better in terms of properly separating the characters, it performed worse on the overall prediction of the characters. For example in attempt 1 the letter “i” was separated into two objects which results in poor accuracy, whereas in attempt 2 this was properly separated.

In the third attempt (attempt 3) the first two attempts were combined. To do so, we removed the upscaling from the second attempt as it resulted in poorer accuracy. Then after using findContours we applied the threshold and remove_small_objects functions followed by the rest of the steps in attempt 1. This combination resulted in the best accuracy in predicting the labels for the test images among the three attempts.

Contributions:

Dennis - Task 1 and task 2 attempt 1+3, and writing
Aliona - Task 2 attempt 2, and writing
Arati - Task 1 tuning, and task 2 attempt 3, and writing

References

[1] Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011, July). The German traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks* (pp. 1453-1460). IEEE.

[2] Mehmood, A., Maqsood, M., Bashir, M., & Shuyuan, Y. (2020). A Deep Siamese Convolution Neural Network for Multi-Class Classification of Alzheimer Disease. *Brain Sciences*, *10*(2), 84.

[3] Murugan, P. (2018). Implementation of deep convolutional neural network in multi-class categorical image classification. *arXiv preprint arXiv:1801.01397*.

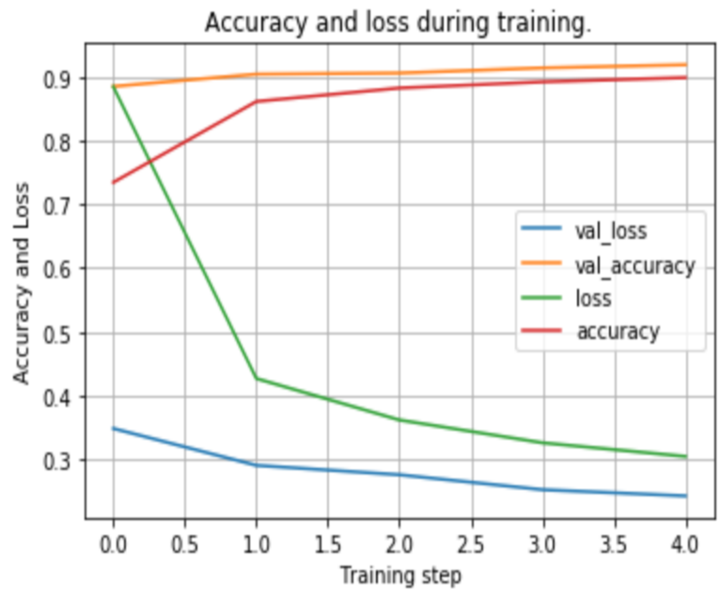
Appendix A: Classification Report

```
              precision    recall  f1-score   support

           1       0.93      0.87      0.90      1008
           2       0.97      0.95      0.96       927
           3       0.87      0.97      0.91       977
           4       0.95      0.91      0.93       947
           5       0.93      0.86      0.90       919
           6       0.97      0.95      0.96       970
           7       0.80      0.85      0.82       908
           8       0.87      0.96      0.91       933
           9       0.77      0.63      0.69       938
          10       0.96      0.92      0.94       975
          11       0.96      0.95      0.95       972
          12       0.69      0.83      0.75       958
          13       0.97      0.98      0.97       941
          14       0.94      0.93      0.93       941
          15       0.92      0.97      0.95       939
          16       0.98      0.96      0.97       969
          17       0.84      0.82      0.83       961
          18       0.93      0.94      0.93       984
          19       0.97      0.96      0.96       951
          20       0.96      0.94      0.95       999
          21       0.93      0.94      0.94      1000
          22       0.91      0.94      0.93       989
          23       0.99      0.96      0.97       976
          24       0.97      0.95      0.96       939
          25       0.96      0.93      0.94       964
          26       0.98      0.97      0.97       975

    accuracy                           0.92     24960
   macro avg       0.92      0.92      0.92     24960
weighted avg       0.92      0.92      0.92     24960
```

Appendix B: Train and validation accuracy and loss


Accuracy and loss during training.
Accuracy and Loss
Training step
val_loss
val_accuracy
loss
accuracy