Project 3: Multi-label Classification of Image Data

Group 65: Kynan Nedellec (260866794), Viet Tran (260924954), Oliver Stappas (260930067)

Abstract

In this project, our team developed a model to classify images consisting of two arbitrarily oriented and placed characters: one letter from the English alphabet and one digit from 0-9. To this effect, we trained our CNN model as well as a set of other predefined models on a set of 30,000 labelled images. Through parameter tuning and variance-reduction techniques such as bagging and dropout, we were able to achieve a training accuracy of 99.7% and a test set accuracy of 97.1% on a dataset of 6000 images.

Introduction

For this project, we were tasked with building the best performing digit-letter classifier. With this aim, we initially built and optimized our custom CNN for which we were able to get appreciable test accuracy while achieving low variance thanks to dropout. We later opted for a majority vote prediction consisting of pre-existing models such as VGG19, mnasnet0_5, etc. The voting included Resnet and Densenet which are known to be excellent for image recognition [1]. We also included an ensemble of Densenet201, ResNet18 and VGG16.

Datasets

We were provided with 2 training datasets of 30,000 images, one labeled and the other unlabeled, as well as a test set of 15,000 images. The data was single channel 56x56 images that consisted of one letter and one digit from 0-9. Data augmentation techniques were applied to the labeled dataset to increase the amount of training data and prevent the model from overfitting. In addition to normalization, random transformations such as rotations, translation and scaling were applied to the original dataset. Training labels were also converted from binary to numbers $(26 * digit\ value + character\ value)$. Further, the training set was also split 80:20 so that we could track accuracy on both sets to ensure no overfitting occured. As for the unlabeled dataset, first a prediction was done on a fifth of the dataset (with voting), then those labels along with the images were used to further train each of the models. We repeated this process for 4 other fifths of the dataset.

Results

To build our custom model, we progressively increased the depth of our CNN model, testing different widths at each layer. Since the training accuracy was higher compared to the validation accuracy, we tested dropout at every layer to reduce overfitting, observing the smallest disparity in training and validation accuracy for 20% dropout after the first linear layer. With further tuning of the learning rate and the number of epochs, we achieved 96.8% training accuracy and 93.05% validation accuracy.

After reaching what we believe to be a performance bottleneck due to the architecture of our model, we started exploring pre-built architectures. As shown in Figure 1, these architectures outperform our model, but tend to overfit. For this reason, we resolved to build a model that

could achieve high accuracy while keeping low variance. We thus opted for an ensemble model as [2] showed that the ensemble performs the best over individual models and as is shown in Figure 1, it overfits less than its individual components. Using this approach, we tried various combinations of models for different hyperparameters, computing training and validation accuracy using a simple voting mechanism where tiebreaking was given to the best performing model. We achieved training, validation, and test accuracies of 99.7%, 97.8%, 97.1% with this method.

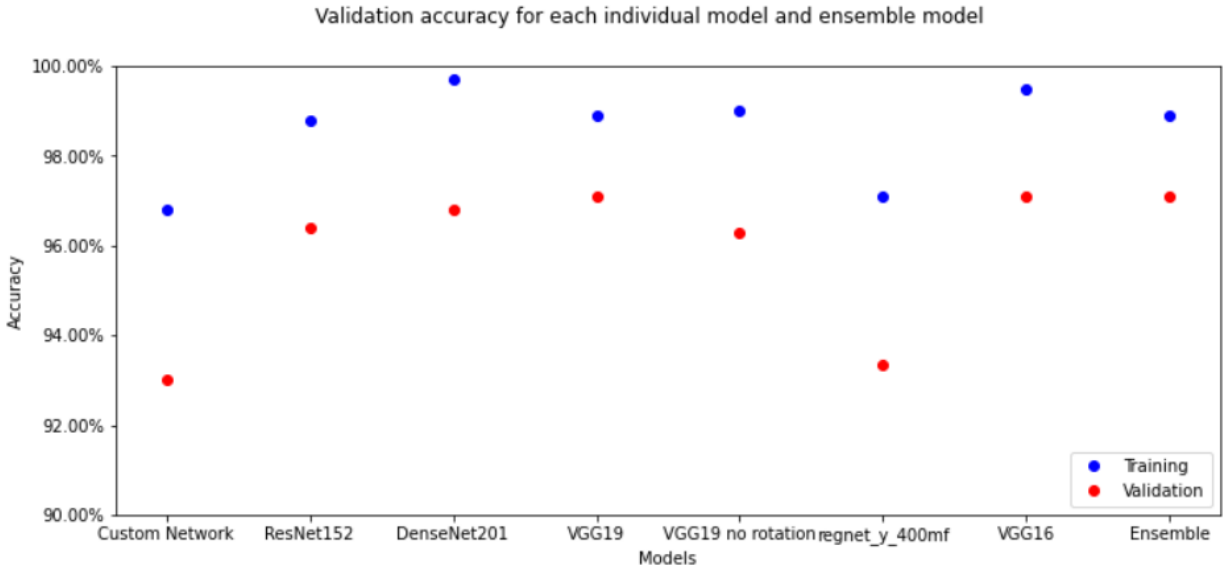

Fig 1: Validation and training accuracy for each model

During the training phase of each model, an initial learning rate was chosen alongside a number of epochs. At the beginning, the loss after every epoch was reduced by a noticeable amount although this number began to stagnate towards the later half of the training phase. (see Figure 2 in appendix) Once this occurred, smaller learning rates were used to further reduce the loss, although this process took a significant amount of time and only slightly affected the loss. Training the models further with the unlabeled did not benefit them all, some improved slightly while others experienced a slight drop in accuracy.

Discussion and Conclusion

This project first showed us the importance of data augmentation and voting in increasing performance and diminishing overfitting. Rotating the images more than 10 degrees confused the networks as the rotated image could be identified as other letters or digits. The same could be said for vertical flips as the network could mix up digits like ‘6’ or ‘9’, or horizontal flips which could confuse ‘p’ with ‘q’. Increasing the batch size when training the models also aided in improving the accuracy.

Extracting the digit and letter and putting them side-by-side significantly decreased performance. For the case of extracting the digit and letter, we believe the poor performance of the trained models to be caused by the poor performance of our digit/letter extractor. Even without any kind of denoising on the original dataset, the various models were still capable of

performing classification to a high degree showing us how powerful these cnns are. Additionally, we believe that achieving 100% accuracy on the training set is unlikely given that some characters and digits are similar to one another (for example ‘0’ ‘l’ can look similar to ‘O’ ‘1’), making it easy to confuse one for the other. Even after training each model with increasing lower learning rates, and reducing the loss, the models still tended to overfit.

This project also helped demonstrate the power of CNNs in situations where a lot of noise is present. In fact, many of the images contained patterns in the background. Because of this we had initially briefly attempted to denoise some images, but this did not improve accuracy. The fact that we still achieved high accuracy without denoising techniques or separating the characters in the images is a testament to CNNs’ ability to find patterns in images.

Possible improvements to our training methods would be to grid search for the best augmentation parameters, use the learning rate scheduler to automate the decrease of the learning rate (would save time over manually changing learning rate), utilizing the unlabeled data in a better manner to train our models or simply running our models for longer.

Reference

[1] F. Chen, N. Chen, H. Mao, and H. Hu, "Assessing four neural networks on handwritten digit recognition dataset (MNIST)," *arXiv preprint arXiv:1811.08278,* 2018.

[2] S. An, M. Lee, S. Park, H. Yang, and J. So, "An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition," *arXiv preprint arXiv:2008.10400,* 2020.

Appendix

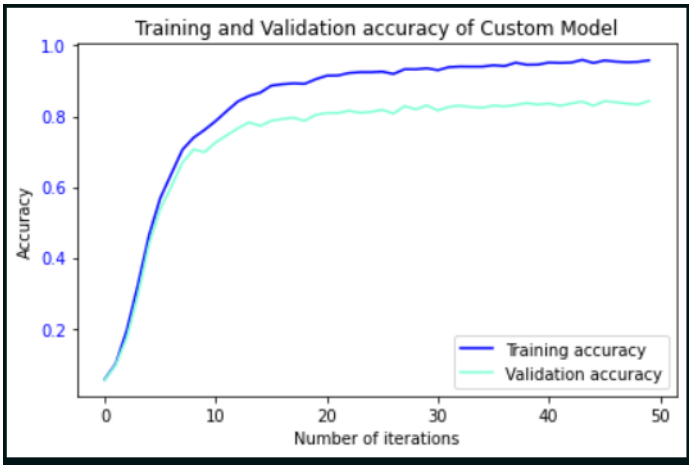

Figure 2: Training and Validation for custom model for first 50 epochs

Statement of Contributions

Kynan worked on getting the initial custom model working, digit extraction and other data augmentation techniques, writing the plotting functions, and writing part of the report.

Viet worked on finding the best models for image classification, implementation of voting, data preprocessing and writing the report

Oliver worked on training the models, finding the best combination of models for voting, data augmentations and training on unlabeled data.