COMP551 Project 3 Report

Haochen Liu (260917834)
Suofeiya Man (260835947)
Ye Yuan (260921269)

1. Introduction

Our task is to train a model that classifies the two characters in each image: one English alphabet (a-z) and one digit (0-9). During experiments, we explored a few ML models, such as clustering, baseline CNN, and VGG network, and used several optimizations and regularization techniques, such as batch normalization, dropout, MaxPooling, Flattening, and so on. Furthermore, we used the encoding and decoding technique to expand the original 36-unit label to a 260-unit label. In addition, we used the original model to give predictions to unlabelled images, then corporate these images into our validation dataset to refine the model. The results indicate that a small encoding-decoding-based VGG network with Adam optimization gives the highest accuracy of around 95% on the public test set.

2. Problem Definition and Algorithm

2.1 Task Definition

There are a few challenges:

1. The maximum pixel value exceeds 255 due to the random noise. We found that the random noise can significantly interfere with the classifier. Hence, we must remove the noise during data pre-processing.
2. The location and orientation of characters are random. Certain rotations can mix up an alphabet and a letter, for example, with a pair of input letter "L" and a rotated digit "7".
3. One alphabet label has two corresponding representations: one lower case and one upper case.
4. With a multi-label classifier, we need to output a 36*1 vector, where each element represents the class probability. The two entries with max values represent the labels of the two characters respectively.

2.2 Model Definition

Baseline CNN (Convolutional Neural Network)

The baseline model has two main aspects: the feature extraction frontend comprised of convolutional and pooling layers, and the classifier backend that makes a prediction. In between, we added a few dense layers to interpret the features and used soft-max activation functions. All layers used the ReLU activation function. We used a conservative configuration for SGD with a learning rate of 5e-4.

VGG

A simplified two-run VGGNet. The first run used labelled images as input to train the model for initializing purposes. The overall structure consists of three parts. The first part is the CONV=>RELU=>POOL blocks. We used a small kernel of size 3*3 with 32 filters and ReLU activation functions. We also applied batch normalization, max-pooling and 25% dropout to prevent overfitting. The second part includes two (CONV=>RELU)2 => POOL blocks. Here we used multiple filters, kernels, and pool sizes to progressively reduce the spatial size but increase depth. The last part consists of FC => RELU layers. From there, the model gained from the first run is used to obtain the predictions of the unlabelled images. The second run used 6000 labeled images as the validation set and combined newly predicted unlabeled images with the remaining 24000 labeled images as the training set. Through a series of experiments, the optimal model implemented random rotation, gaussian filter, one-hot encoding, and pixel normalization during pre-processing, as well as Adam optimizer for optimization.[2]

Clustering with segmentation

Based on the VGG Net above, there are two main additional aspects of this model. We applied the semantics segmentation to separate the digit and the alphabet in the same image. Then we used the K-Means Model to cluster different classes of objects. By doing so, we expect that the VGG Net can learn the digits and alphabets separately to minimize the correlation between them.

3. Experimental evaluation

3.1 methodology

We divided the labelled images into training data and validation data, assuming the underlying distribution exists. The entire training process contains two steps. In the first step, the 30000 labelled data were used to train a small VGG model. This model is then used to predict the labels of the 30000 unlabelled data. In the second step, we separate 6000 labelled data as validation set, and combine the rest 24000 labelled data with the predicted 30000 unlabelled data as the training set to train another small VGG model. After the training, the newer model is used to predict the data of test set.

Two metrics referred are accuracy and loss. After training, we used the test set to evaluate the model both during each epoch of the training run, so that we can create learning curves, and at the end of the run so that we can estimate the performance of the model.

3.2 results

Through the comparison of the models, the models ranked by the performance from highest to lowest are: small VGG >>> baseline CNN > Clustering with segmentation [1]

Model	Baseline CNN	Small VGG	Clustering
Accuracy on public Leaderboard	0.7727	0.9425	N/A
Accuracy on validation set	0.7634	0.9432	0.4646
Loss on validation set	0.0043	0.0021	0.1165

3.3 Discussion

Conclusion:

As one can see, the small VGG network is robust to solve multi-label classification problem. It also verifies that a deeper CNN model performs better than shallow ones. To increase the diversity of the training data, we predicted the unlabeled images by using the first-run model and combined them with the labeled images as a new training set. The performance improves from 92% to 94% after merging the predicted unlabeled images to tune the model.

Strengths of the method

VGG network has several outstanding strengths. Firstly, its structure is very simple and modular. The overall architecture consists of identical small kernels and pooling that helps to reduce the number of parameters. The convolutional layers inside VGG blocks are similar such that it is easy to be stacked and reused. Max-pooling effectively reduces the number of features while increasing the depth. Secondly, the two-run-based strategy steadily tunes the VGG model and speeds up convergence: it uses a relatively small and shallow network to initialize the model, then uses unlabelled images to refine the configurations.

Weakness of the method

Nevertheless, during the experiments, a few weaknesses are also non-negligible. Since the VGG network is very deep with many parameters, this requires enormous resources for computation and storage. Furthermore, the more iterations, the more time-consuming the model becomes. The trade-off between accuracy and resources required need to be evaluated.

Underlying properties of the model / data significant for the result

Besides the VGG network, the encoding-and-decoding technique also plays a key role in the improvement of performance. One-hot-encoding is relatively better than the original output label because the model only needs to learn one label rather than learning two labels at one time.

Problem and future improvement

Considering the massive computations, only several data pre-processing techniques are evaluated. However, we strongly believe that there exists more data augmentation and noise-removing techniques that are worth trying. For example, dimension reduction, and brightness corrections. Furthermore, the result of our experiment with segmentation contradicts our intuition, which is that, by segmentation, each character can be treated independently and hence gives a more precise prediction. We would like to investigate this contradiction and try to find the solutions. Even though we got the highest accuracy by using one-hot-encoding, it implicitly enhances the correlation between the alphabets and digits. For one image, the model will learn and recognize it as one object rather than one alphabet and one digit. This implicit correlation will be harmful to the performance. Using semantics segmentation is a possible solution. Unfortunately, the result contradicts our expectations.

Bibliography

Brownlee, Jason. “How to Develop a CNN for Mnist Handwritten Digit Classification.” Machine Learning Mastery, 14 Nov. 2021, https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/.

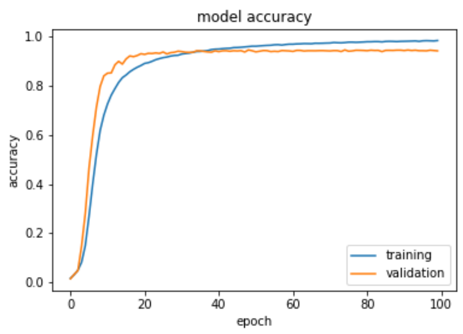
Essam, Nader, et al. “Multi-Label Classification with Keras.” PyImageSearch, 17 Apr. 2021, https://www.pyimagesearch.com/2018/05/07/multi-label-classification-with-keras/.

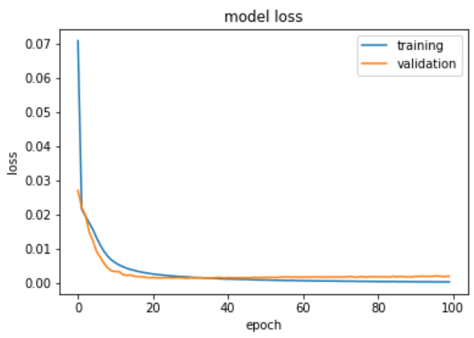
Statement of Contribution

Ye Yuan: clustering and segmentation, hyper-parameter-tuning
Haochen Liu: VGG network, baseline CNN
Suofeiya Man: hyper-parameter tuning, report

Appendix

[1] The training plots for the first and second run are shown respectively:





[2] Experiments for optimization

Stage	**Method**	**Explanation**	**Effect and evaluation**
Data pre-processing	gaussian filter	blur an image by a Gaussian function.	The visual effect is a smooth blur resembling of the input image with reduced image noise and detail.
	encoding and decoding	Use One-Hot encoding to encode the given labels. There are 260 (26 alphabets * 10 digits) classes.	Compared to the alternative that uses the original 36*1 vector as the label, this encoding makes more sense when predicting the unseen data, and it indeed performs better.
	Random rotation	A good practice of data augmentation.	Effectively increase the diversity of the dataset.
	Pixel normaliza tion	Resizing and rescaling Normalize the pixel values to be around 0.	The effect is less obvious than other methods, such as gaussian filter Rescaling squashes pixel values to the range (0,1)
	Image segmenta tion	Train the model to extract the two components.	We can separate all images into two components, one alphabet and one digit. The error rate is around 4 percent.
	Clusterin g	Use K-Means Model from Scikit-learn to cluster similar objects.	Generally unsuccessful. Similar alphabets and digits will be considered as the same class. Like "B" and "8" will be clustered to the same class.

Optimization	Adam optimizer	Theoretically faster convergence.	According to the experiment results, we chose Adam optimizer because it gives a better validation accuracy score and faster convergence.
	SGD	Higher probability to converge to the flatter minimum.	
	Model depth	Add more layers to the model, such as additional blocks of convolutional and pooling layers or more dense layers in the classifier.	In general, a deeper model improves performance on the CNN model.
	Binary cross-entropy	Treat each output label as drawn independently from the underlying Bernoulli distribution.	Allow the model to penalize each output independently.
	Learning rate	Establish an initial learning rate of 5e-4	Clearly shows the progress
	Epoch size	We experimented with epoch sizes of 50, 100, and 150, and find that: Epoch size of 50 gives the best accuracy with the given test labels. Yet epoch size of 150 gives the best accuracy on the public leaderboard.	We decide to use the epoch size of 150. larger epoch shows incremental improvements via propagation
	Drop-out rate	Established the dropout rate of 0.5, then modify it to be higher and lower later	Finalize dropout rate to 0.5 for the dense layers and 0.25 for other convolutional layers
	Batch size	Accelerate the training speed	We used the batch size of 16