COMP 551

Mini-Project 3

Yiran Fu 260904135

Barry Li 260912069

Mingzhe Feng 260886087

Abstract

In this project, we used the combo MNIST dataset to develop models for image classification. We chose the convolutional neural network as our base model and implemented a SmallVGGNet architecture. Image transformation techniques such as Gaussian blurring as well as normalization were applied to remove noise and regularize models. We used one-hot encoding to relabel our dataset. We also experimented with Semantic segmentation, but it was eventually abandoned due to lack of effective clustering techniques. After hyperparameter tuning, our best model obtained an accuracy of around 95% on the validation and public test set.

Introduction

Convolutional Neural Network is the most popular neural network model being used for image classification problems. There are many famous CNN architectures such as AlexNet, VGGNet and ResNet, etc which all can have really good results on image classification. After researching, we found that VGG is currently a very popular choice for extracting features from images and decided to build an architecture similar to it. We ultimately chose a SmallVGGNet that includes five convolution layers each with 3*3 convolutional filters and three pooling layers as our base network architecture. We used softmax as our final layer activation function with categorical cross entropy loss.

Dataset

The dataset is a combo MNIST dataset where each 56*56 image in it consists of two characters: one letter from a to z and one digit from 0 to 9. In our preprocessing, we first denoised the data by using Gaussian blurring from the scipy.ndimage package. After trying different values of its parameter sigma, we found that 0.69 is the most optimal one which can clear the noise in the background for most images while not influencing the shape of the character and number. We also designed the threshold method to turn the image into only binary colors and the rotation method to do data augmentation. To avoid overfitting and gain some insights about our performance on unseen data, we split 20% of our data as the validation set.

At the beginning of our project, we tried to train our multilabel dataset directly, but the training accuracy was only around 50 percent. We came up with two ideas to improve our performance. The first idea is by doing one-hot encoding and letting the CNN network learn two objects together with correlation. The other idea is to do semantic segmentation by coming up with our own algorithm and splitting when the object is observed. We decided to work in a parallel fashion. For the first idea, we implemented a one-hot encoding method to convert the original label of length 36 (10 for digits & 26 for characters) into that of length 260 (a combination of digits & characters, i.e. "3z"). In this way, we turned the task from a multilabel, multiclass classification to a single label, multiclass classification. For the second idea, we designed a semantic segmentation method to split the character and the number into two images. To achieve this goal, we first used the threshold method to make a clear contracted border between objects and the background. Then we found the contours of each object and bounded it into a rectangle all by using the functions provided by OpenCV. After that, we cut the bounded image and filtered the noise by setting a minimum length. However, sometimes the hand-written objects were clumped together so that they would be recognized as only one image. In this case, we used a greedy assumption assuming the image can be perfectly split by finding the bigger value between its length and width and directly cut it into two parts evenly. Through observation, this assumption was proved to be pretty valid. When more than two parts of the image are being bounded, we calculated the area of each part and selected two parts with the largest areas. Since the smaller one were much likely to be noise than real objects.

We also made use of the 30000 unlabeled images by firstly putting the labeled dataset into our CNN architecture and using this trained model to predict the unlabeled dataset. Then we merged the labeled and unlabeled datasets together and retrained another model using this combined dataset.

Results

Choice of optimizer and hyperparameters

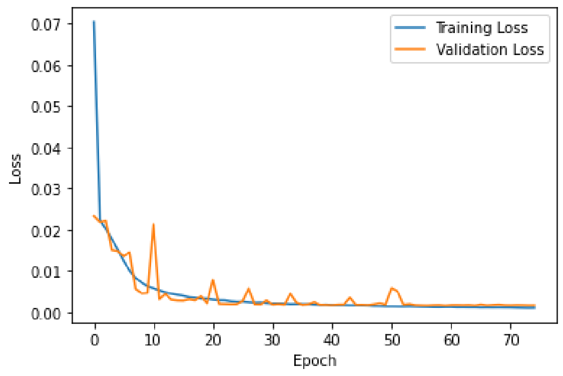
To optimize, we used the Adam optimizer which is a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments. It performed better than SGD after experiments. We chose the learning rate of 0.001 because it gave a relatively high accuracy with fair training speed. The first hyperparameter we tuned was the batch size when fitting the model. We tried 8, 16, 32 and found that 16 had the best accuracy (But the difference was really small). We also trained the model with different epochs 50 and 75 and their accuracy on the validation set was almost the same. But a larger epoch turned out to yield a little better result on the Kaggle leaderboard.

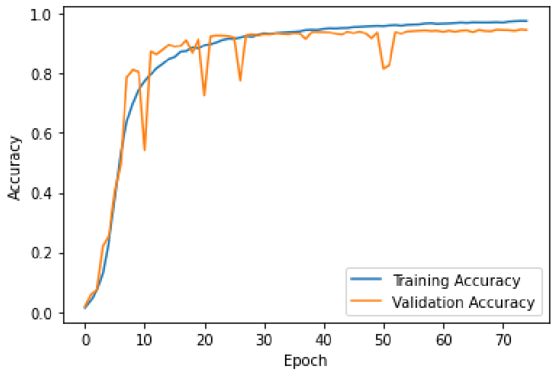
Thus, we built our final model with Adam optimizer, learning rate 0.001, batch size 16 and epoch 75.

One-hot Encoding VS Semantic Segmentation

For semantic segmentation, we forced all the images into two parts and found that around 96% of them were correctly splitted. However, a bottleneck was encountered: we realized that we were unable to split the labels correctly and efficiently (since we have no idea which label corresponds to which splitted image) and this resulted in a 40ish validation accuracy when we passed both the character and number labels to the split image, which was a lot lower than one-hot encoding. We tried a few ideas such as clustering using kmean to overcome such obstacles. However, that also went abysmal so we were forced to give up the idea.

The idea of one-hot encoding became pretty successful. During the training interactions, the loss decreased from 0.1 to 10^-4 at the end of the training, while the validation accuracy at the end of each epoch first increased significantly, and stayed around 94.5% in later epochs. After combining the unlabeled dataset and adding rotations, the validation accuracy improved from 94.5% to 96.1%.





Conclusion and Discussion

The performance of our Small VGG Net with one-hot encoding is proved to be effective with around 95% validation and test accuracy. However, we think that the one-hot encoding method has limitations since it cannot avoid the correlation between the character and the number in each image. We do believe that segmentation can have a higher ceiling if we can successfully split and assign the correct label to each image, and in the future, we want to investigate more in this field.

Statement of contributions

Minzhe Feng, set up the base prediction pipeline
Barry Li, data preprocessing and segmentation, idea gathering
Yiran Fu, helped with preprocessing and model selection, did project write-up.

Citation

Keras Small VGG Network
https://www.kaggle.com/mobasshir/keras-small-vgg-network

Multi-Label Image Classification with Neural Network | Keras

https://towardsdatascience.com/multi-label-image-classification-with-neural-network-keras-ddc1ab1afede

Basics of Image Classification Techniques in Machine Learning

https://iq.opengenus.org/basics-of-machine-learning-image-classification-techniques/

CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more…

https://medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5