COMP551 Project3

Jianing Xu, Xuanzi Wang, Yuxin Leng

November 2021

Abstract

In this project, we built the *CNN* with 3 layers and 4 layers, *VGG* and *Ressnet* models to identify the digits and letters in the images. We preprocessed the dataset to augment the training size and remove the noise in the background. We trained the models and found that *VGG16_bn* and *Resnet18* performed best on the prediction and reached an accuracy of 0.95.

1 Introduction

In this project, we tested accuracy of different model structures on a dataset containing images. Our goal is to detect the hand-written letters and digits contained in each image. By varying the model structures, model parameters, and data preprocessing parameters, we found out that deep models, such as *VGG* and *ResNet*, performed better than shallow models on this dataset. In addition, data denoising, normalization, and augmentation also helped us to fit the model by reducing variance and increasing bias. Our best performing model has a structure same as *ResNet18* and we reached an accuracy of over 0.95.

CNN is a regularized version of multilayer perceptrons, compared with other deep learning structures, *CNN* can give better results in image and speech recognition.

VGG is a classical convolutional neural network architecture which utilises small 3 x 3 filters and two 3x3 convolution kernels to replace 5*5 convolution kernels. *VGG* increases the depth of the network and improves the effect of the neural network to some extent under the condition that the same perception field is guaranteed. [1]

Moreover, *ResNet* improves accuracy by adding considerable depth. Its internal residual block uses jump connection to alleviate the gradient disappearance problem caused by increasing depth in deep neural network. It is modified on the basis of *VGG19* network, and the residual units are added through short-circuit mechanism. An important design principle of it is: when the size of feature map is reduced by half, the number of feature map is doubled, which maintains the complexity of network layer. [2]

2 Dataset Preprocess

The training dataset contains 30,000 images and each image contains a number and a letter. As there are some noises in the background of the image, we blurred the images and set a threshold so that the noise could get removed. We have tested different thresholds to find the one that performed best on our model. Figure 1 shows a comparison of a picture before and after denoising. In addition, we normalized data to have a mean of 0 and standard deviation of 1. By rotating images within a certain range, the size of the training data has been increased from 30,000 to 120,000. Data augmentation has introduced variability and helps prevent overfitting.

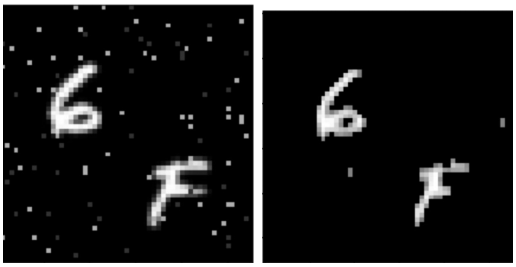

Figure 1: Denoising

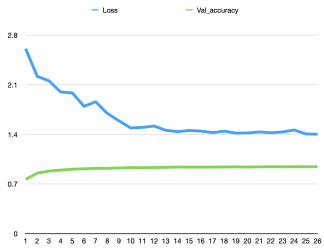

Figure 2: Resnet18 Val_acc and Loss

It is noticeable that we have also been provided with an unlabeled dataset. After researching online, we found that it is possible to perform a semi-supervised learning. After training our model on labeled dataset, we predicted pseudo labels of unlabeled dataset and computed loss of the combination of labeled and unlabeled datasets. Pseudo-labeling has increased our dataset. However, we found out that it does not significantly increase the overall accuracy. Thus, in the final model, we did not include this method.

3 Results

We firstly designed models by ourselves. First, we ran the *CNN* model with 3 and 4 layers, respectively, with each layer containing several 2D convolution, pooling, and ReLU layers. We have also added regularization to each layer to prevent overfitting. We selected *crossentropy loss* to calculate the loss of each step and *SGD* as the optimizer. Different dropout rates have been tested on the 4-layer model and we selected 0.1, which generates the highest validation accuracy without overfitting. After converging, we reached an accuracy of 0.66 for 3-layer and 0.84 for 4-layer model. However, these models converge after over 40 epochs and takes a long time to converge.
Next, we updated the model into *VGG*, which is much deeper than the previous models. we have tried *VGG11*, *VGG13*, *VGG16* and *VGG19*, also we tested these 4 *VGG* models with and without *batch_norm*. We chose the model *VGG16_bn* with accuracy 0.90. We have also tested accuracy on model *ResNet* with different depths. Figure 2 contains the result of *ResNet18*. As we can see, the model converges after 13 epochs and has reached an accuracy of over 0.94. With the training process, the accuracy is growing and the loss becomes lower.

4 Discussion and Conclusion

As the prediction accuracy has increased as we increase the depth of the model, we realized that in this case, deeper models are more robust to new data. Thus, choosing an appropriate model is significant during training. In addition, data augmentation has helped us to train a better model on larger dataset and regularization in the model has prevented the model from overfitting. In addition, GPU is also crucial during the training since it speeds up the processing of images.

Contribution

Xuanzi Wang was responsible for building 3- and 4-layer models and data denoising. Jianing Xu was responsible for data augmentation, building, and training VGG and ResNet models. Yuxin Leng was responsible for helping training models and writing the report.

References

[1] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[2] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

Appendix

4.1 Selecting learning rate in the model

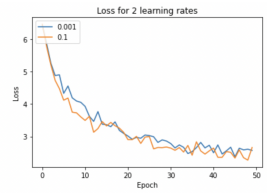

Figure 3: Loss for different learning rates

4.2 Loss comparison for different models

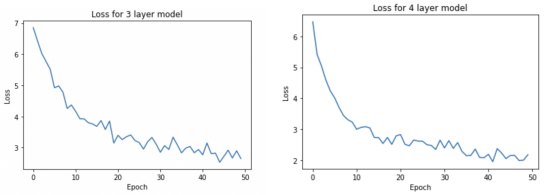

Figure 4: Loss for 3-layer and 4-layer models

4.3 Comparison of different thresholds during denoising

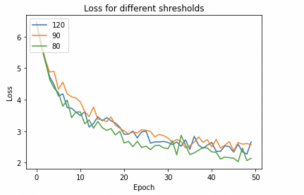

Figure 5: Loss for different thresholds

4.4 Comparison of different types of thresholds during denoising

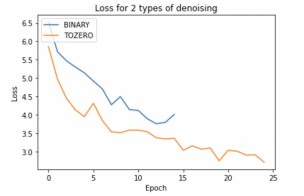

Figure 6: Loss for different types of thresholds

4.5 Loss of 4-layer model without dropout

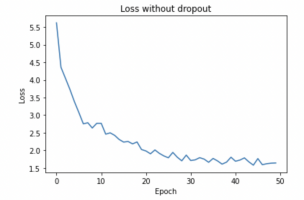

Figure 7: Loss without dropout