Neural Networks and Deep Learning COMP9444 22T3

Project: Face Mask Detection

Group: A Group

Group Member:

Jingyu Zhou, Jinzhe Li,

Qiyao Zhou, Xiang Ji,

Zihao Jiang

Abstract:

Considering the pandemic in past two years, designing corresponding mask mask face detection and recognition algorithms based on mask face data to help with face recognition has become a need in some areas. In this CV project, ResNet is used to train a binary task. To investigate the performance of VGG, Resnet34, and Resnet18 in this task, we built up data enhancement, datasets, dataloader, train structure, and tested them with different setups. The weakness of this experiment is that the data used is idealized (lack of side faces, mask types, etc.) and no test training module for inappropriate mask-wearing is added. In the future, this experiment can expand the test scope, testing the impact of epoch size on the experimental results, etc., and the use of the CascadeClassifier module to extract the face part for cameras' real-time testing is possible as well.

Introduction

Considering the pandemic in past two years, designing corresponding mask mask face detection and recognition algorithms based on mask face data to help with face recognition has become a need in some areas.

In this cross-validation project, ResNet is used to train a binary task (mask or no mask). To investigate the performance of Resnet34, VGG, and Resnet18 in this task, we built up data enhancement, datasets, dataloader, train structure (train test loop, optimizer, loss function, figure plot), and tested them with different setups (learning rate, optimizer, volume of datasets...)

Data Sources

A total of 3508 color images are used, 3008 for training (1504 masks 1504 no masks), 500 for testing (250 masks 250 no masks),the data source link can be found here: https://www.kaggle.com/datasets/prasoonkottarathil/face-mask-lite-dataset

Exploratory Analysis of Data

Transforms

Cropping, flipping, rotation, color changing, and normalization of the dataset are being used to enhance the data in order to increase generalization capability and prevent overfitting. Based on the different data augmentations used for initialization in the self-built model and the migrated learning models, the self-built model and the three migrated learning models are not comparable.

Datasets & DataLoaders

Call data_transforms and put the image data in the dataloader, the bath size is set to 256 (the benefits of larger batch size: the memory utilization is improved, the parallelization efficiency of large matrix multiplication is increased; the number of iterations required to run through an epoch (full data set) is reduced, and the processing speed for the same amount of data is further accelerated; within a certain range. In general, the larger the Batch_Size, the more accurate its descent direction will be, and the less training oscillation will occur. Based on the folder name for the unique hot encoding as Y-value and, image data as x-value, and XY-value will be loaded together into the dataloader in the final shuffle to disrupt the order.

NeuralNetwork

The most challenging aspect is to avoid overfitting, if the images of acc and loss oscillate too much repeatedly, it may mean that the model is overfitted, to prevent overfitting, we try to use different data sources, increase data volume, increase data enhancement and adjust learning rate and optimizer to test.

A Network class is provided with three models for migration learning (vgg,resnet18, resnet34). Calling the get_network method initializes the selected model and prints the model structure and the parameters to be learned and adjusted. The parameter configurations in previous convolutional and pooling layers among these three models are not updated, where the last linear output layer, the out_features are modified and set to be derivable.

SelfNet

The SelfNet class provides a CNN model that we wrote ourselves. Compared with the residual network like resnet, the vgg model is relatively less complex. However, for this binary classification task with high quality image dataset and obvious features, vgg will be a bit too big to be used. Therefore, based on the vgg model, our model only keep the first two convolutional layers and the pooling layer, and the relu activation function is still being used in between the convolutional and pooling layers to keep the gradient.

The number of input_features and out_features are changed to reduce the computational pressure. For the fully-connected linear layers, similarly, only the first two linear layers are retained, the number of input_features and out_features is reduced. The relu activation function and dropout rate in the two linear layers are retained as well.

Train and Test

We borrowed the model training method from the pytorch website, rewriting and improving have been done for it to be compatible with GPU operations. and passing in two lists to record

The loss and Accuracy values are recorded after each epoch in test_loop.

During the testing phase, we built up different train structure (train test loop, optimizer, loss function, figure plot), and tested them with different setups (learning rate, optimizer, volume of datasets, etc).

In general, these models perform well in most cases as shown in Figure 1, some unsatisfactory results were observed for the following reasons:

Overfitting occurred because using a small amount of training data and poor quality of datasets(The mask is blurry and strange in the photo, and the face part in the photo is very small). Overfitting can also occur if the learning rate is too high.

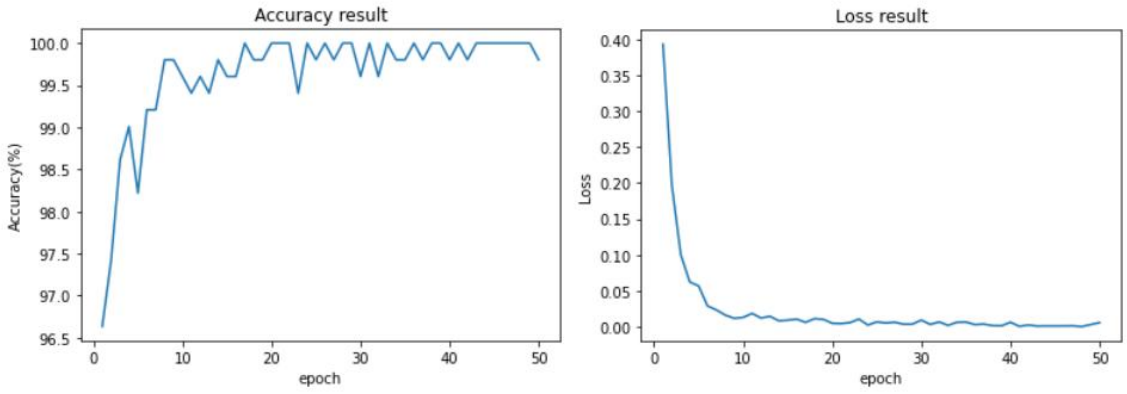

Figure 1 Test0 Self_network, epoch=50, optimizer=Adam lr0.0001, loss_function=CrossEntropyLoss, Train dataset volume=3008

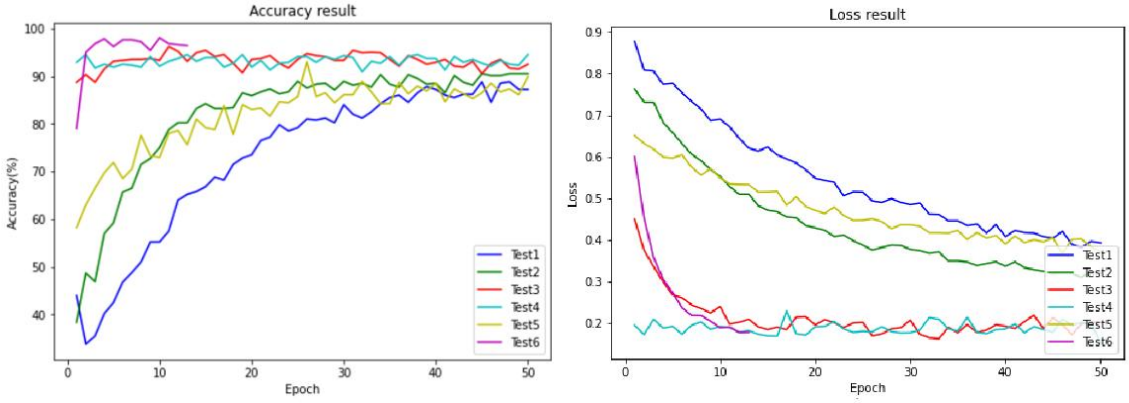

Figure2 Result Summary

Test1(Resnet18, epoch=50, optimizer=Adam lr0.0001, loss_function=CrossEntropyLoss, dataset volume=2008)
Test2(Resnet18, epoch=50, optimizer=Adam lr0.0001, loss_function=CrossEntropyLoss, dataset volume=3008)
Test3(Resnet18, epoch=50, optimizer=Adam lr0.001, loss_function=CrossEntropyLoss, dataset volume=3008)
Test4(Resnet18, epoch=50, optimizer=SGD lr0.0001 mon=0.1, loss_function=CrossEntropyLoss, dataset volume=3008)
Test5(Resnet34, epoch=50, optimizer=Adam lr0.0001, loss_function=CrossEntropyLoss, dataset volume=3008)
Test6(vgg11 epoch=50 optimizer=Adam lr0.0001 loss_function=CrossEntropyLoss, dataset_volume=3008)

Conclusion

For 3000 data, it is more stable to use the adam optimizer and set the learning rate to 0.0001 resnet18. Resnet34 fluctuates slightly, and vgg rises fastest. With resnet34, because it has a deeper network structure, it will take more training time and require less learning rate. The SGD optimizer may have larger data changes than the Adam optimizer, but because of the momentum, it may skip the local minimum value to reach a new maximum accuracy or minimum loss value.

The weakness of this experiment is that the data used is idealized (lack of side faces, mask types, etc.) and no test training module for inappropriate mask-wearing is added. In the future, this experiment can expand the test scope, such as testing more practical test data, testing the impact of epoch size on the experimental results, etc., and if you want to use the camera for real-time testing, you need to add the CascadeClassifier module to extract the face part.

References

Alippi, C., Disabato, S. and Roveri, M. (2018) "Moving convolutional neural networks to embedded systems: The Alexnet and VGG-16 case," *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* [Preprint]. Available at: https://doi.org/10.1109/ipsn.2018.00049.

He, K. *et al.* (2016) "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* [Preprint]. Available at: https://doi.org/10.1109/cvpr.2016.90.

Jais, I.K., Ismail, A.R. and Nisa, S.Q. (2019) "Adam optimization algorithm for wide and deep neural network," *Knowledge Engineering and Data Science*, 2(1), p. 41. Available at: https://doi.org/10.17977/um018v2i12019p41-46.

Kottarathil, P. (2020) "Face Mask Lite Dataset," *Kaggle Data* [Preprint]. Available at: https://www.kaggle.com/datasets/prasoonkottarathil/face-mask-lite-dataset

Kumar, A., Sarkar, S. and Pradhan, C. (2019) "Malaria disease detection using CNN technique with SGD, RMSPROP and Adam Optimizers," *Studies in Big Data*, pp. 211–230. Available at: https://doi.org/10.1007/978-3-030-33966-1_11.

pytorch (2022) *Optimizing model parameters*, *Optimizing Model Parameters - PyTorch Tutorials 1.13.0+cu117 documentation*. Available at: https://pytorch.org/tutorials/beginner/basics/optimization_tutorial.html