AIST4010 ASM1 Kaggle Report **Cheung Chi Wang 1155158048**

Model Architecture

Originally, the model is based on the architecture of ResNet [1], which used shortcuts to skip over some layers to avoid to problem of vanishing gradients. After implementation of the model, indeed, it can achieved an validation accuracy over 60 precents with proper data augmentation. However, the model cannot achieve over 65% accuracy without any pretraining.

In case of the above experiment, I implemented another model which is based on the architecture of EfficientNet [2], a group of convolutional neural networks (CNNs) that were developed to increase the prediction accuracy while minimizing the computational resources required for training and inference. This is a very efficient model for the competition when I need to train in a shorter time with a limited computational power. The first step in constructing the model is to define a base network, which consists of a series of convolutional and pooling layers. The base network is then scaled by increasing its depth, width, and resolution using a set of coefficients. The model includes a set of compound scaling coefficients that balance the trade-off between depth, width, and resolution.

As stated in the paper, there are B0 to B7 architectures which for the subsequent architectures, we increase the value of phi which is used to scale the coefficient while assuming resources doubles in every stage. However, due to limited resources of Kaggle, I faced the problem of Cuda out of memory and hence I must lower the batch size to avoid this problem. Unfortunately, this increased the training time dramatically and even affected the performance of the model by overfitting. I took reference with the EfficientNet-B0 architecture and implemented it by reducing the size of the model.

Loss function

Cross-entropy loss function is used for training the model as it is a commonly used loss function for multi-class classification problem. Cross-entropy will calculate a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem.

Hyperparameter

For optimizer, I implemented different optimizer and chose the best one among them, namely Adam, Stochastic Gradient Descent(SGD), RMSprop. Eventually, I chose to use SGD optimizer with learning rate of 0.1, momentum of 0.9. A learning rate scheduler is also implemented to better converge the model with a step size of 20 and gamma of 0.2, which means every 20 epochs, the learning rate of optimizer will be scaled by 0.2. This reduces the problem of overfitting the model. Moreover, a drop rate of 0.6 is implemented in the architecture as the trick of reducing overfit. I trained the model with 80 epochs and more of the epochs did not give a better result. The batch size of the data loader is 128, increasing it would result in Cuda out of memory.

Data augmentation

Overfitting is a huge problem in training with limited data. Therefore, data augmentation is implemented to produce more noise and data to the model [3]. I first resize the 64x64 photo to 224x224 to capture more

features. Then a Random Rotation with 8 degrees is implemented to the data, followed by ColorJitter with 0.2 as the factor for brightness, contrast, saturation, and random Horizontal flip with a probability of 0.5 is also implemented. One thing to notice is that when I added random vertical flip to the data, the accuracy was decreased and this may because of too much distortion of human faces. At last, I normalize the data by mean = [0.485, 0.456, 0.406], and s.d. = [0.229, 0.224, 0.225].

Balancing the size of dataset of each class by sampler

Originally, I implemented a weighted random sampler when I used the data loader. I calculated the weight of each class by dividing the size of data of each class by the total size of the dataset. Then the corresponding weights were implemented for the weighted random sampler [4]. Although this balance the data size of each class and avoid the overfitting problem of certain classes, under sampling the data has led to underfitting of the model, and it is not applicable to my model. Therefore, I eventually only load the data by data loader setting shuffle = True.

Reference

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for Convolutional Neural Networks," *arXiv.org*, 11-Sep-2020. [Online]. Available: https://arxiv.org/abs/1905.11946. [Accessed: 25-Feb-2023].

[3] C. Hughes, "Demystifying pytorch's Weightedrandomsampler by example," *Medium*, 30-Aug-2022. [Online]. Available: https://towardsdatascience.com/demystifying-pytorchs-weightedrandomsampler-by-example-a68aceccb452. [Accessed: 25-Feb-2023].

[4] T. Huang, "Pytorch 提供之 torchvision data augmentation 技巧," *Medium*, 24-Feb-2021. [Online]. Available: https://chih-sheng-huang821.medium.com/03-pytorch-dataaug-a712a7a7f55e. [Accessed: 25-Feb-2023].