Report: Assignment 3
COMP 8740: Neural Networks
ID: U00744746

Hosneara Ahmed

October 11, 2021

Introduction

Transfer learning is a successful technique in deep learning to let the model learn more without training it from the scratch. It is all about reusing the whole pre-trained model or re-training some layer(s) of it to improve the performance. For example, ResNet50 is a quite popular CNN model for image classification which is very deep (50 layers). It is already trained on ImageNet dataset. However, while using the same weights (pre-trained model) for tiny ImageNet dataset (500 image samples from each of the 200 classes and 10,000 validation images from ImageNet), it may not perform the same way as it did for ImageNet for different factors such as optimizers, dataset discrepancy, normalization, etc. In this assignment, we have experiemented with pre-trained ResNet50 model on tiny ImageNet dataset. Then we finetuned the model to improve the performance according to data. We have also used the best model trained on CIFER-10 dataset. Lastly, we have ensembled all these three models and trained to observe the performance improvement compared to individual three models.

Methodology

- **Evaluate model:** Pretrained model means the model is trained on a specific dataset and the weights are already learned. First, we have extracted the pretrained weights from Resnet50 that is trained on imagenet dataset. We have to change the last layer as in tiny imagenet we have only 200 classes. This is called transfer learning, which means pretrained resnet50 is being evaluate on another dataset. However, the performance can be better or worse we cannot tell beforehand. However, instead of using the weights as is, we can freeze one or more layers of pretrained resnet50 and retrain them later which is called fine-tuning.
- **Fine-tuning Resnet50 for improvement of model:**
  - **Data augmentation:** When we loaded the dataset for the evaluation of pretrained resnet50 on tiny image dataset, we adopted a few data augmentation stragies such as shear, zoom, flip, etc. Later, to fine tune, we have standardized the images so that the model does not overfit.
  - **Retraining last few layers Resnet50:** As per second task, we will see in the evaluation section that the performance of resnet50 as is, is not good enough for tiny imagenet dataset. For this reason, we have changed last few layers and added another convolution layer. After that, we have used a pooling layer and we used globalaveragepooling

instead of maxpooling as it takes the average of all the feature units. So it is supposed to give an overall representation of all the neurons in the receptive field. To get rid of overfitting problem, we have used dropout regularization after that.

- **Ensemble models:** Lastly, to further improve the model, we have used ensembling of three models - the pretrained resnet50, fine tuned resnet50, and pretrained CNN model on CIFER-10 dataset. Among all, whichever class gets the highest probability for an image, will be classified to that class. This is called votingclassifier mechanism. It is well-established that model ensembling performs better in learning from data. We will see the improvements in the results section.

Model Architecture

Model overview

Data Augmentation

For finetuning the model, we have centered all data points to zero mean, then subtracted the standard deviation from each of them. This is to standardize the images. Also, we have preprocessed the base resnet50 inputs with tensorflow's preprocess_input so that it preprocess data according to given input images.

Retrain Batch normalization layers

Batch normalization normalizes the data for each mini batch. However, when we are using resnet50 base model, then we have to retrain the batch normalization layers as the batch for the base model is different for our data. In other words, the normalized result will be different. This is because out data is much smaller than the actual imagenet dataset. So we have frozen all layers in the base model except the batch norm layers to retrain them later.

Convolution layer

We have added another conv2D layer with only 32 kernels of size (3x3). For the pooling, I have empirically chosen GlobalAveragePooloing followed by Dropout with 20% probability. Lastly one Fully Connected (FC) layer with 1024 neurons and finally a FC layer for classifying the images with softmax activation.

Model Ensemble

We have taken the three models (pretrained resnet50, finetuned resnet50, and pretrained cifer10 model from the last assignment) and ensembled them together. However, we cannot use as is any model while using pretrained model, at least we have to retrain the last layer to match with the number of classes for classification. So we did for the pretrained Resnet50 and cifer10 model too. We have used with votingclassifier for image classification in the ensemble model. voting classifier classifies according to majority voting which means what class is chosen by majority of classes will be the final class for the ensemble model.

Experiments and Results

I have run the models with batch size 128 in Google Colab Pro. The data augmentations are done only for training image, not for validation ones. Both training and validation images are rescaled within 0 to 1.

The loss function for each model is categorical_crossentropy as this is a multi-class classification problem. Optimizer is set to RMSProp with learning rate 1e-4. Each model is run for 50 epochs and loss, accuracy, validation loss, validation accuracy are observed.

Evaluation

Pre-trained Resnet50

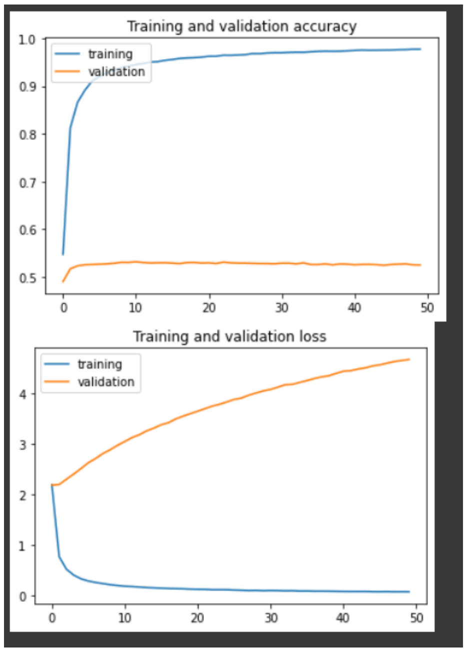

Figure 1: Loss vs. Accuracy in pretrained Resnet50 on tiny imagenet dataset

From the figure 1, we can see the training loss and accuracy is going low and high respectively which means it is learning well. However, for validation data it is not true, rather the loss is increasing which means it cannot classify correctly in the validation set. It means the model is overfitted. The reason might be, the resnet50 is trained on a bigger dataset which is imagenet but tiny imagenet is way smaller that. This means we have to fine tune the model to improve the accuracy and loss.

Fine tuning Resnet50

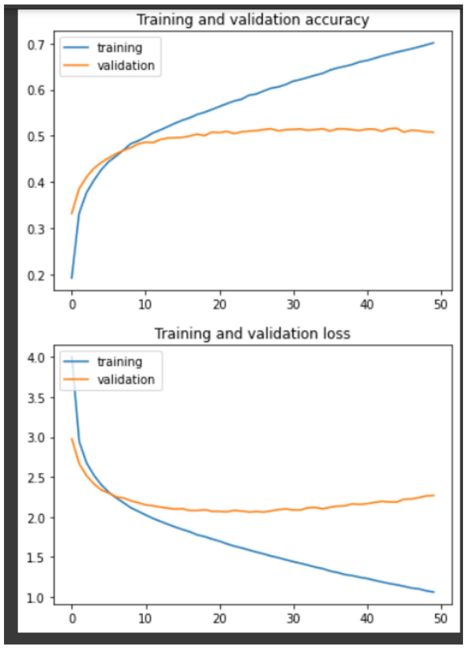

Figure 2: Loss vs. Accuracy in fine tuned model on tiny imagenet dataset

From the figure 2 we can see, after finetuning the model is performing better for training set as well as validation dataset. Even though the loss for validation is higher than the training loss, eventually after a few epochs training loss is less than the validation loss as expected. Also, the training loss is not that much high than the validation loss which indicates the model is not overfitting. But one thing to notice is that, after epoch 35, the validation loss is increasing again for a couple of epochs. This might be small spike in the loss (as validation loss decreased after epoch 43) but as we have trained the model for relatively few epochs (50), we cannot give any decision of good or bad model based on this. But overall, compared to the pretrained base model, fine tuned model is working much better so far in terms of loss and accuracy metric.

Ensembling models

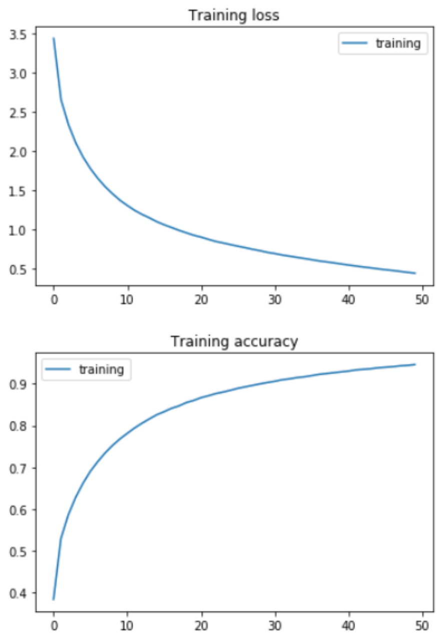

Figure 3: Loss vs. Accuracy in ensembled model on tiny imagenet dataset

After ensembling the three models, we can see the model is learning well from the training data because the loss is decreasing and accuracy is increasing gradually. However, we have to have a validation set to make sure the model is not overfitting.

Discussion and Comparison

From the experiments, we can see the fine tuned model performs better than the pretrained resnet50 model as the overfitting problem is not there anymore. After ensembling the three models, the training performance is even better.

Conclusion

We have experimented with resnet50 pretrained model along with fine tuned one to see how the fine tuned model performs compared to base model for tiny imagenet data. Ensemble model is popular for improving model performance and we have seen the same for our case too. But we have to validate the model on the validation data for the ensemble model. Also, we have to run the models for more epochs to see when the loss and accuracy get stable. But still we can see the initial improvement of the models which is a good start to choose which one to choose.