Introduction to Deep Neural Networks (Spring 2021)

Final Project (100 Pts, Due Date: June 6)

Student ID 2016312761

Name 여혁수

1. Description of your method and reason for the choice (more than 1 page, most important)

I added and used three main methods for learning. First is data augmentation. I felt that more data is needed to successful learning. So I transformed both labeled train data and unlabeled train data. I used resize function for transforming because all image data should be fitted to learning of one model. I chose the value of resize to 128, because the resized image can contain various local features stably than resizing to 32. Also it is maximum value that it does not occur any memory leak problems if the model is resnet50.

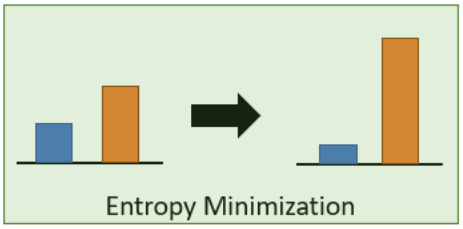
Also I used randomcrop function and horizontalflip function to make different image data from original image data. Randomcrop gets some partial range of image data, and horizontalflip flipped the image horizontally. I transformed labeled train data 3 times, but transformed unlabeled train data only one time. The number of unlabeled data is 7 times more than labeled data, so it takes too much time to learn if I do several transforms for unlabeled data.

Also I used normalization like image = (image - mean) / std in all transformations to improve performance. Normalization helps model related with cnn perform better. Normalization helps get value of data within a range and reduces the skewness which helps learn faster and better.

Second, I used pseudo labeling method. To improve performance, I should contain unlabeled image data in train data. I trained with only the labeled image data first, and then train the unlabeled and labeled image data both. I added softmax layer for output of model. The result of softmax prediction goes to sharpening, and I only chose some partial prediction for training which

has max value in probability of 10 labels more than 0.99. Sharpening is just same procedure like below.

$$\text{Sharpen}(p, T)_i := p_i^{\frac{1}{T}} \Big/ \sum_{j=1}^{L} p_j^{\frac{1}{T}}$$



e.g)

Sharpen (T=0.5)

[0.4 0.6] ➡ [0.307 0.693]

$$\text{Sharpen}([0.4\ \ 0.6], 0.5) = \left[\frac{0.4^2}{0.4^2+0.6^2}\ \ \frac{0.6^2}{0.4^2+0.6^2}\right]$$

$$= [0.307\ \ 0.693]$$

This partial data will be learned in training and the loss will be calculated with output and prediction. At that time, I do some weight balancing of loss. This is my third method, considering the progress of learning each epoch, I decreased the loss value at a certain rate for a kind of regularization. The larger the epoch, the certain rate is calculated bigger.

I chose resnet50 for my model. Because its performance is best of other models like resnet18, densenet161, or my own cnn model. Also in case of densenet, it takes too long time to learn than resnet50.

2. Train and predict procedure (e.g., hyperparameter choice/tuning)

Overall procedure is like below.

Train labeled image data -> Predict label of unlabeled image data -> Choose image data which has max value of probability more than 0.99 -> Train unlabeled image data & labeled image data

In training labeled image data, there is no difference with skeleton code. Then in predicting label of unlabeled image data, we need softmax layer to get probability of each label. And tuning the value of probability with sharpening method. If max value of probability is more than 0.99, I contained it to training data.

I tuned hyperparameter a little. Most part of hyperparameter is fixed, but I just increase epoch of learning. When I try different model and different method, I tuned many hyperparameters. But in final model, I used default hyperparameter almost.

3. Any trial you made for performance

I first use the class CNN() in model.py for model. With data augmentation, and pseudo labeling, I increased the test accuracy to 82%. And I used various transform methods. For example, brightness, sharpness, randomrotation(90). But accuracy does not increase. So I used ensemble technique that using several models and aggregate the outputs of each model. I aggregate model resnet50, densenet161, and class CNN(). By this method, I increase test accuracy to 85%. I continuously found the problem of my method, and change many times in transformations. I changed my transformation to randomcrop, horizontalflip, and normalization. And I felt that ensemble is useless and just takes long time. So I just use resnet50, and use my fine-tuned transformations. And finally I get about 92% test accuracy.

4. Final results on the public leaderboard (Screenshot of your leaderboard)

My kaggle ID: yeohyuksoo