Report

Haoyu Hu (hh25), Boda Li (bl21)
Team name: COMP540_bl21_hh25

1 Introduction

In this project, we build our models based on the ResNet. We tried ResNets that are randomly initialized and are initialized with the pre-trained model on Image Net. By using the knowledge from Iamge Net, we find the pre-trained model has a much better performance and take less time to reach high accuracy. The good performance comes from the fact that the pre-trained model can successfully extract the features from the image. Instead of training the feature extraction part, we only need to optimize the mapping between the extracted features and the final predictions during the training. We then extend this idea by replacing the fully connected layer of ResNet by a three-layer feedforward neural network (FNN), where FNN introduces a more complicated mapping between features and predictions. Our final model is a combination of several ResNet based models via ensemble methods. Besides, a label smoothing and random data pre-processing have also been introduced to avoid the over-fitting problem. Another key technique we considered is the ensemble method on the test set. We increase the dimension of the test set by introducing three data pre-processing methods. For each method, we obtain a predicted probability from our final model. Then we calculate the weighted average of these predicted probabilities and generate the final prediction.

2 Data

2.1 Exploratory Data Analysis (EDA)

Before going through the instruction and guidance for Kaggle challenges, we explore the data and get some basic understanding of them. For this Kaggle challenge, the pictures are from 251 different categories labeled from 0 to 250. The number distribution for each category in the training set is shown in histogram (Fig. 1). Most categories share similar numbers of pictures and the skewness value from our eye measurement is near zero. The category got only 34 training pictures is the marble cake. The best way to help to improve the classification for this class is by adding more pictures for this category. Because the policy of Kaggle challenges would not allow us to add figures, the only way for solving this problem is by data augmentation. The other reason for data

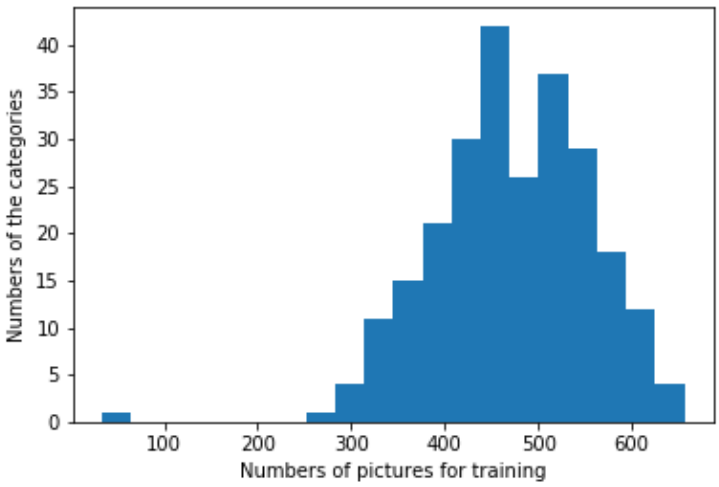

Figure 1: Histogram of the distribution of the numbers of pictures for all categories.

augmentation is that the number of training pictures is small compared with the Image Net database. Image Net asks for classifying 1000 categories but with 14 million pictures which are much larger than our training set. So only with data augmentation technique can we acquire a good accuracy on the classification of this Kaggle challenge.

2.1.1 Some unrelated pictures

By reviewing the pictures, we found there are some pictures are totally not related to the food categories. So, some treatment for the data is needed to decrease this noise for data. Some unrelated pictures are shown quite a lot in some specific food pictures like poi (Fig. 2 (a)). Here we imply a label smoothing technique described in Section 5 which helps to decrease the noise of the training data.

2.1.2 Different sizes and resolution of pictures

There are multiple sizes and resolutions inside the training data. The pictures not only share different resolutions but also different length/width ratio. So a resizing and cropping technique will be needed to help the training and prediction work.

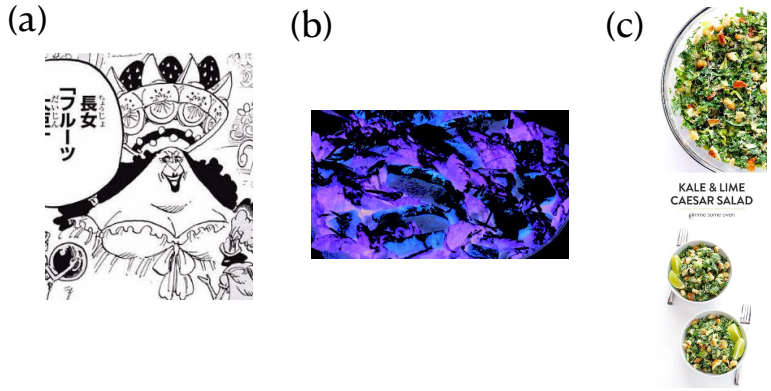

Figure 2: (a) Unrelated figure.(b) Reversed color. (c) Food not in the center.

2.1.3 Reversed color in the training set

Some of the photos in the dataset are taken in a reversed color mode (Fig. 2 (b)). The reversed color mode will be greatly different from the regular pictures. The input of our data is in the 'RGB' mode, while the reversed color will be shown in the form of $(255-R, 255-G, 255-B)$ form. We apply a normalization in the pre-processing session to help them normalize into a similar range as the normal color mode picture.

2.1.4 Food not in the center of the picture

Most food pictures are shown in the center of the picture, but some extreme long pictures are shown the food separately on the upper half and lower half which might cause a problem when we want to train it as shown in Fig. 2(c). This requires us to use a random crop to treat the training sample to acquire the object for feeding into the CNN. This will be described in the data pre-processing section.

2.2 Data Pre-processing and Data Augmentation

Before performing the training algorithm on the data set, some data pre-processing is necessary. As discussed in the EDA section that some images contain multiple objects and the objects may not at the center of the image. In order to extract the correct object from the image, we randomly crop the images and re-size all the images to the same size (224,224). We also consider some random pre-processing, namely random 90-degree rotation and random horizontal flipping, in order to augment the data set. These procedures are not only pre-processing data, but play the role of data augmentation. By enlarging the training set,

these procedures can avoid overfitting issue. Finally, we normalize the data by subtracting the mean and dividing by the standard variance of the Image Net to help the network work better. In summary, we perform the following data pre-processing on the train set: i)random 90-degree rotation; ii) random cropping and re-sizing; iii) random horizontal flipping; iv) normalization. These processes are in sequence. For the validation set, we initially apply the same pre-processing technique on the validation set. However, the randomness on flipping, cropping is bringing extra bias on the validation set which decreases the performance. So later on, a random data pre-processing on the validation set is not necessary. We first re-size the image to a size larger then $(224, 224)$ and then center crop it to $(224, 224)$. We try different re-size scale, and find first resize the figure to $(300, 300)$ have better performance by picking the best parameter by validation set (highest validation accuracy). We found that these could bring 5 % to 10 % accuracy improvement on the validation set by switching from adopting the same pre-processing method to use resizing and center cropping. We apply the same data pre-processing method to the test set.

2.3 Ensemble Method for Data Pre-processing

This technique is developed during this Kaggle Challenge and we did not see any paper that talking about this trick. This trick is useful for improving accuracy in this problem. We have this idea from switching pre-processing techniques on validation sets. In the beginning, we used random rotation, random crop, and resizing on the validation set. This method both help to capture some food object, not in the center but also perform data augmentation which is not useful for validation and test set. So, we switched to a deterministic process that only performs resize and center crop. But,this method also loses some information on the edges and the classification on the food not in the center. Along this line of thinking, inspired by the bagging method, we perform multiple data pre-processing on the validation/test set, instead of using single data pre-processing. We basically consider three data loader (data pre-processing) denoted by $f_{1,2,3}$:

- f_1: Re-size the figure and normalization.
- f_2: Center Crop, re-size the figure and normalization.
- f_3: Random data pre-processing (Same data pre-processing used on the train set.)

. For each data point x, we calculate $f_n(x)_n$ and generate predicted probability $\{p_n^i\}_n$ for each class i and each data pre-processing n. As shown in Fig 3. The final probability p^i is a weighted average of each p_n^i:

$$p^i = \sum u_n p_n^i \tag{1}$$

, with u_n is the weight . To determine the u_n, we search for the maximum value of validation accuracy on by tuning $\{u_n\}$ and then decide to use $u_1 : u_2 :$

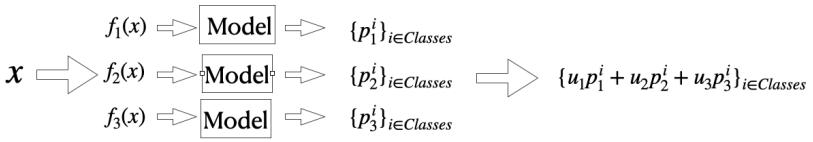

Figure 3: Ensemble method for test data set.

$u_3 = 0.425 : 0.575 : 0.25$. We compare this ensemble method with simple center cropping and find the validation accuracy has a 3% increase. The idea behind this ensemble method is to increase the dimensionality of test/validation data, such that the model can gain more information of the input data and make a better prediction.

3 Model selection and training strategy

To achieve the best performance, we train multiple models and then combine them via the ensemble method to generate the final results. We basically consider two different types of models, one is the basic ResNet model as described in Ref [1], the other one is the combination of ResNet and naive feed-forward network.

3.1 ResNet

For the ResNet, we try ResNet-101, ResNet-152 [1]. We compare the performances between random initialization and initialization with a pre-trained model on Image Net. The best accuracy we can get in the not pre-trained data is only 40 % for top 1 accuracy on the validation set. It turns out the pre-trained model has a much better performance which already trained with millions of images, we decided to focus on the pre-trained model.

Two training strategies/designs have been used: i) training the whole model; ii) first training the top fully-connected layer and then training the whole network [5]. The idea behind the second method is using the bottom layer as feature extraction. Since the pre-trained model has a good performance on Image Net, the convolutional layers can successfully extract the features of the images. The last fully-connected layer only combines these features and generate predictions. Instead of updating all the layers during the training which may disturb well-behaved convolutional layers and cost longer time, we only update the last layer for the first two or three epochs and then turn on the update of the bottom layers to improve the final performance.

3.2 Chaining Model

Following the idea of feature extraction, we then consider the combination of ResNet and feed-forward neural network. For the ResNet part, we use ResNet-

101, ResNet-152 [1], ResNeXt-101 [7] and WideResNet-50 [8]. We then replace the fully-connected layer of the ResNet by a three-layer feed-forward neural network with 2000 hidden units of each hidden layer and leaky ReLU as the activation function. Unlike the simply fully-connected layer, FNN can represent more non-linear relations between the features and predictions. Similarly to the feature extraction case, we first train the feed-forward network by fixed the ResNet part and then update all the layers together.

3.3 Ensemble Method

To generate the final output, we use the ensemble method. We pick four or five models that have the best performance (highest validation accuracy) to form an ensemble. Each model gives a predicted probability p_i^α where α is the model index and i is the class index. Then the final predicted probability p_i is a weighted average:

$$p_i = \sum_\alpha p_i^\alpha w^\alpha \tag{2}$$

, where w^α is the weight of each model and is decided according to the validation accuracy.

4 Hyper-parameters and Optimizer

For the learning rate, we use the step decay learning rate [2], with

$$\eta_e = \eta_0 (0.1)^{e/3} \tag{3}$$

, where η_e denotes the learning rate of epoch e, and η_0 denotes the learning rate of epoch 0. We set $\eta_0 = 0.01$ and multiplied the learning rate by 0.1 every 3 epochs.
In the case of feature extraction and chaining model, the fully connected layers are also trained with a step decay learning rate with $\eta_0 = 0.01$. However, before turning on the updates of all the layers, we first do cross-validation to pick the best initial learning rate. As we showed in Fig. 5, we calculate the validation accuracy after 2000 iterations (batch size = 16) at different learning rates and decide the η_0. In practice, we first calculate the validation accuracy at $1e^{-1}, 1e^{-2}, 1e^{-3}, 1e^{-4}, 1e^{-5}$ and then reduce the grid size to find the best learning rate. However, due to the limited computational resources, we didn't fine-tune the learning rate.
In addition, we use the stochastic gradient descent method with Nesterov momentum as the optimizer, and let the momentum factor be 0.9.

5 Preventing Overfitting

Due to the limited computational power, we found it's impossible for us to fine-tune the regularization strength since the best regularization depends on the

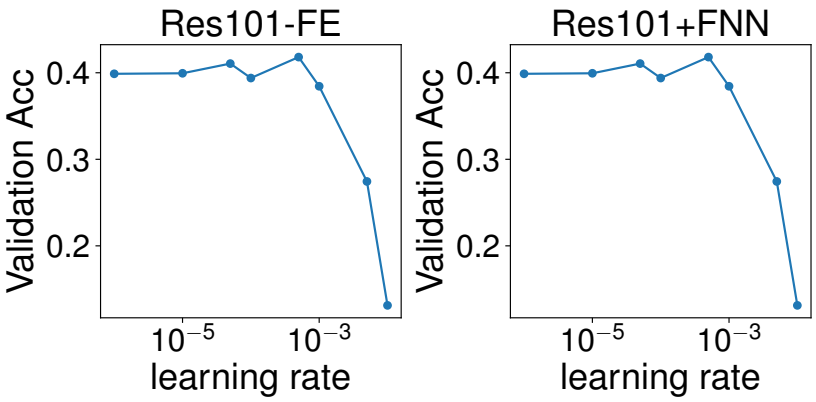

Figure 4: (Color online) Cross validation to decide η_0 for ResNet-101 with feature extraction (FE) and the combination of ResNet-101 and FNN.

number of layers. We simply set the regularization strength to zero and use other techniques, such as data augmentation, label smoothing, and ensemble method, to decrease variance. We also considered dropout but ResNet is not equipped with dropout which makes us need to build a dropout enabled ResNet which is too time-consuming.

As we mentioned before, we introduce several random data pre-processing on the train set, which effectively increases the amount of training data and avoids overfitting [4]. Besides, the label smoothing that was proposed in Ref [6] has also been applied. Label smoothing changes the true probability to

$$q_i = \begin{cases} 1-\epsilon & \text{if i == label[i]} \\ \epsilon/(\text{num of classes} - 1) & \text{otherwise} \end{cases} \tag{4}$$

and the loss is generated by the negative cross entropy $J = -\sum_i q_i \log(p_i)$ with p_i the predicted probability. In practice, we set $\epsilon = 0.1$. The label smoothing encourage the output to be finite for all the labels and prevent the network to be over-confidential. This also helps to solve some of the unrelated figure problems. Finally, we also consider the ensemble method. By combining the predictions from multiple models, the variance and generalization error are reduced.

In practice, our model doesn't exhibit a very severe over-fitting problem. Our main challenge is still about reducing the bias instead of reducing the variance. From the learning curve as shown in Fig. 5, there is still some room for us to reduce the training loss and the validation accuracy is still growing. In summary, the methods we discussed in this section are enough for us to prevent over-fitting.

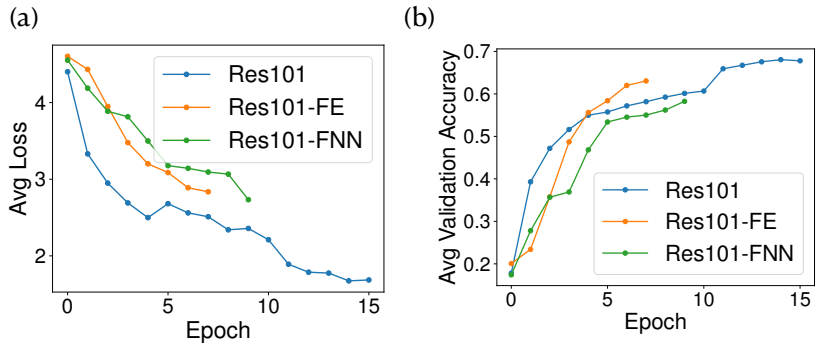

Figure 5: (Color online) (a) average loss for each epoch. (b) validation accuracy of each epoch.

6 Feature Extraction (Transfer learning)

In this section, we discuss the effect of feature extraction (FE) or transfer learning [3]. As we discussed before, we utilize the knowledge gained while training the model on the Image Net to solve the classification problem we met. The convolutional part of the ResNet plays the role of extracting the features, and the fully connected layer transforms the features to the predictions. Along this line of thinking, it's no need to disturb the feature extraction part at the first several epochs while the learning rate is quite large. Then we freeze these layers and only train the remaining part. However, a single connection layer may not be able to capture the complicated relations between the features and classes, so we can replace the fully connected layer a feedforward neural network.
In Fig. 5, we show the evolution of the average loss of each epoch. Res101 denotes the ResNet101, Res101-FE denotes ResNet101 training with feature extraction, and Res101-FNN denotes the combination of ResNet101 and FNN. For ResNet101-FE and Res101-FNN, we turn on the update of the all the layers after 4 epochs.
From the validation accuracy, we find using the feature extraction, the ResNet101 only requires 6 epochs to reach a 65% validation accuracy, which is much faster comparing with directly train the ResNet101 that requires more than 10 epochs. For the ResNet101-FNN, we expect are better performance, however, after 10 epochs training, the performance is still worse then Res101 and Res101-FE. It may because adding the FNN makes the model more complicated and less easy to converge. We believe a better result should be achieved if we have more time to train this model.

Similar features are shown in the ResNet 152. Since the domain knowledge between ImageNet challenge and Kaggle iFood 2019 is different, we still tried to use the progress from the ImageNet pre-trained ResNet model. We think that adding more fully connected with the ReLU activation unit between each layer

Model	Validation Accuracy	Train Loss	Epochs
ResNet-101	0.68	1.69	18
ResNet-101-FE	0.63	2.83	8
ResNet-101-FNN	0.58	2.73	9
ResNext-101-FNN	0.65	2.73	12
WideResNet-50-FNN	0.67	2.54	24
Ensemble model (Final model)	0.72	Depend	Depend

Table 1: Performance of different models.Validation Accuracy is the Top 1 accuracy

can help to better generalize these domain knowledge changes which will let the ResNet better generalize on our testing set.

7 Performance and final model

In Tab 1, we list the performance of the models we tried. We choose five models with highest validation probabilities: ResNet-101, ResNet-152, ResNet-152-FNN, ResNext-101-FNN and WideResNet-50-FNN (FNN represent three fully connected layers with ReLU activation in the middle), and using the ensemble methods (as described in Section 3.3), with the weights: $w_1 : w_2 : w_3 : w_4 : w_5 = 4 : 4 : 3 : 2 : 5.5$. respectively, to construct the final models. The relative weight was dtermined by both their relative accuracy and the grid search on whole validation set. In addition, an ensemble method on the test/validation set as shown in Section 2.3 has also been used to improve the performance. The final top-one validation accuracy on this ensemble model is , which is about 5% better then the single model.
In Fig. 6, we show the confusion matrix, the number of samples, validation accuracy, F-measure, recall, and precision. We notice that the model has a bad performance for class 116, 162, 238. However, this results from the small amount of data in these classes as shown in Fig. 6(b). Ignoring these classes, the overall validation accuracy is around 70%, and the F-measure, recall precision all have pretty good performance with the value around 0.7 on average. However, it's also worth mentioning that, even after ignoring the classes with little data points, the performance still has a large variance, with the validation accuracy varying from 0.5 to 0.99. This indicates, for certain classes, the performance is still quite poor.

8 Kaggle Timeline

In Tab 2, we show the time-line of our kaggle submission. We use the 28-th and 29-th submissions with scores 0.1204 and 0.12240 respectively as our final submission. The final model we used has been discussed in the previous section.

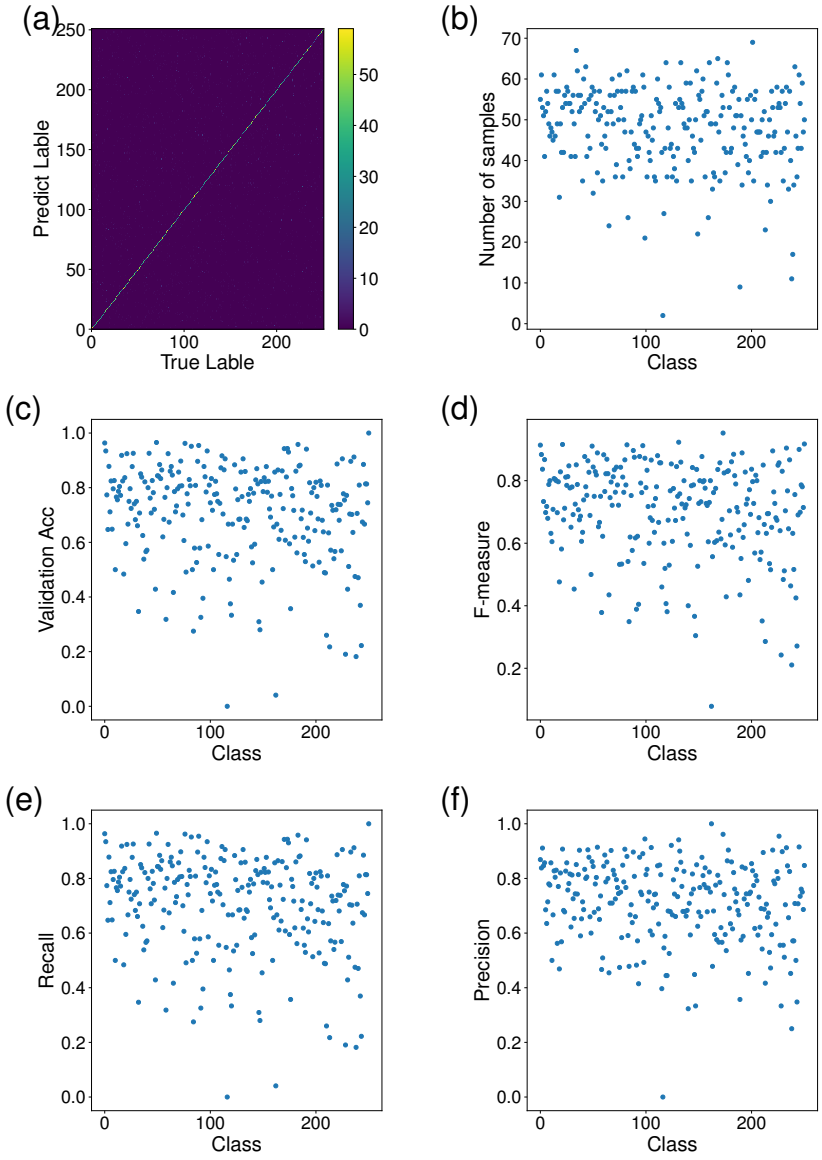

Figure 6: (Color online) (a) Confusion matrix. (b) Number of samples of each class. (c) Validation accuracy. (d) F-measure. (e) Recall. (f) Precision.

#	Model	Score	description
1	ResNet-18	0.75132	3 epochs. w/o pre-train.
2	ResNet-18	0.62950	5 epochs. w/o pre-train.
3	ResNet-18	0.58534	8 epochs. w/o pre-train.
4	ResNet-18	0.60848	9 epochs. w/o pre-train.
5	ResNet-18	0.51556	10 epochs. w/o pre-train.
6	ResNet-18	0.50910	11 epochs. w/o pre-train.
7	ResNet-152	0.65723	10 epochs. w/o pre-train.
8	ResNet-18	0.50863	12 epochs. w/o pre-train.
9	ResNext-10132x4d	0.31222	30 epochs pre-train with reg 0.0001 top3.
10	ResNet-152	0.25831	32 epochs. w/o pre-train.
11	ResNet-101	0.15329	18 epochs. Pre-train.
12	ResNet-152	0.15881	30 epochs. Pre-train.
13	ResNet-101 + FE	0.19135	8 epochs. Pre-train.
14	Ensemble	0.14002	Model: ResNet-101(pre-train), ResNet-152. Weight:1:1
15	Ensemble	0.13978	Model: ResNet-101(pre-train), ResNet-101-FE, ResNet-152. Weight:3:2:3
16	ResNet-152+FNN	0.16950	18 epochs pre-train
17	ResNet-152+FNN	0.16833	24 epochs pre-train
18	ResNet-152-FNN	0.16316	32 epochs pre-train
19	ResNet-152-FNN	0.16527	34 epochs pre-train
20	ResNext-101-FNN	0.17115	7 epochs
21,22	Ensemble	0.15176, 0.14953	Use all the trained models.
23	ResNext-101-FNN	0.16950	12 epochs
24	Ensemble	0.13778	Models: ResNet-101, ResNet-152,ResNet-152-FNN, ResNext-101-FNN(All pre-trained). Weights:3:3:2:2
25	Ensemble	0.13215	Models: WideRes50, ResNet-101, ResNet-152, ResNet-152-FNN, ResNext-101-FNN. Weights:3:3:3:2:2
26	WideResnet50	0.14742	24 epochs pre-train
27	Ensembled dataloader wideresnet50	0.13414	weight matrix in 2.2.1
28-30,32,33	Ensembled dataloader ensemble model	0.122-0.124	All models weight matrix for ensemble model varies
31	WideResnet50	0.14800	28 epochs pre-train

Table 2: Timeline of the Kaggle submission.

9 Challenges and discussion

The main challenge we met is limited computational resources. We don't have enough resources to fine-tune the hyper-parameters and train the models for more epochs. For several models (such as ResNet101-FNN in Fig. 5), the validation accuracy is still decreasing, but we don't have enough time to train it for more epochs. Basically, instead of training a single model for a lot of epochs to get very high accuracy, we prefer to train multiple models with less number of epochs and then ensemble the models to increase the performance.

During this project, we find the pre-trained model is quite useful. Without using a pre-trained model, the highest accuracy is around 0.6. With the pre-trained model, we can easily get this accuracy within three or four epochs. Since we don't have enough resources, this helps us a lot to increase the performance. Another useful technique we used is the ensemble on the test/validation set as discussed in section 2.3. By the ensemble method, the model can gain more information from the test/validation data, and increase our final accuracy by $\sim 2\%$.

If we have more time, we decide to try other models, such as the GoogleNet and compare the performance between different models. Because the differences in the different model are larger which make the correlation between each model decreases. This will benefit the accuracy of our ensemble model. Also, the effect of data pre-processing hasn't been fully studied and the hyper-parameters haven't been fine-tuned.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[2] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks, 2018.

[3] Minyoung Huh, Pulkit Agrawal, and Alexei A. Efros. What makes imagenet good for transfer learning?, 2016.

[4] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning, 2017.

[5] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition, 2014.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

[7] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks, 2016.

[8] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016.