Fine Grained Food Image Classification Based on Transfer Learning and Ensemble

Yunda Jia yj32, Yu Wu yw92

Department of Electrical and Computer Engineering, Rice University,
Department of Electrical and Computer Engineering, Rice University

April 25, 2020

Recent studies in image classification have demonstrated a variety of techniques for improving the performance of Convolutional Neural Networks(CNNs). In this report, we present how we train deep learning models on iFood 2019 at FGVC dataset. We implemented transfer learning and assembled the results of ResNet 152, ResNeXt 101, DenseNet 201, and Wide-ResNet 101 models. Our proposed results show that we earn an 11.603% error rate on test data from Kaggle competition. Our approach achieved the 4^{th} position in this competition.

1 Introduction

1.1 Project Goal

Automatic food identification can assist towards food intake monitoring to maintain a healthy diet. Food classification is a challenging problem due to the large number of food categories, high visual similarity between different food categories, as well as the lack of datasets that are large enough for training deep models. Therefore, the purpose of our project is to find proper

deep models to identify food types and predict the fine-grained food-category label given an image.

1.2 Project Statement

For iFood dataset , there are 251 fine-grained (prepared) food categories with 118,475 training images, 11994 validation images and 28377 test images collected from the web. We follow the data science pipeline from pre-processing the data to modeling and validation.

2 Preliminaries

2.1 Data Explatory

In this competition, the dataset has 251 fine-grained (prepared) food categories with 118,475 training images collected from the web and the validation set of 11,994 images and the test set of 28,377 images with human verified labels. First we look at the distribution of the dataset, by finding the number of image ids and the number of images, we notice that although many classes do not have the same image number, most classes are in the range of reasonable numbers of images, except for label 162 [figure1]. For the quality and properties of images, we find that the images of original dataset are not resized to the same size and have a very wide range of the sizes. Based on the situations we found, we do the data preprocessing and data augmentation.

2.2 Data Augmentation

Data augmentation is a powerful tool in cases of increasing the size of training dataset and resizing images. Broadly, data augmentation in images involves changing the orientation of images, selecting a part of an image, randomly or otherwise. We discuss a few transforms here,including random rotate the image from 0 degree to 30 degrees, then random crop the image into 224 * 224 pixels, random flip the image horizontally, random flip the image vertically, and normalize

the image to shift the mean each pixel position to zero and variance as one.

2.3 Image Pre-processing

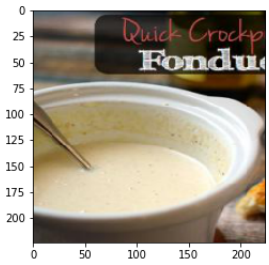
After Image augmentation, there are different combinations of transform ways. In order to show how image pre-processing works, here is original image and images after basic transforms.

Figure 1: Original Image

There are some basic transformations:

- Random Resized Crop: Training images are cropped into 224 * 224 pixels randomly shown in **Figure. 2.a**
- Random rotation: Training images are random rotated from 0 degree to 30 degrees shown in **Figure. 2.b**
- Random Horizontal Flip: Training images are flipped horizontally with probability $p = 0.5$ shown in **Figure. 2.c**
- Random Vertical Flip: Training images are flipped vertically with probability $p = 0.5$ shown in **Figure. 2.d**

- Random Affine: Training images are randomly affined with 30 degree shown in **Figure. 2.e**
- Normalize: Training images RGB channels are normalized with mean 0.485, 0.456, 0.406 and standard deviation 0.229, 0.224, 0.225 shown in **Figure. 2.f**



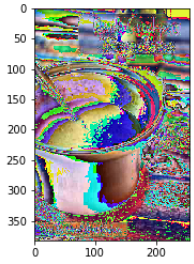
(a) Random Resized Crop

(b) Random Rotation

(c) Random Horizontal Flip

(d) Random Vertical Flip

(e) Random Affine



(f) Normalize

Figure 2: Transformations

After the combination of different transforms, the input image looks like **Figure. 3**

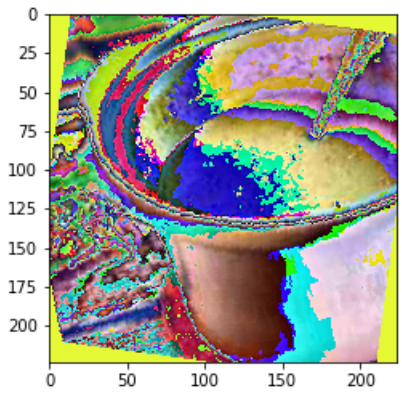


Figure 3: Input image

2.4 Training Procedure

We use Google Colaboratory platform to train our model with GPU Tesla P100-PCIE-16GB. We save checkpoints into Google drive in case of refresh memory of google colab.

In the training phase, an image is transformed after the different combination of transforms shown above and the input image looks like **Figure. 3**. As for hyperparameters, we tried different combinations. We use 64 and 32 batch size which is related to GPU memory. Our loss function is cross entropy loss, optimizer is stochastic gradient descent where we selected base parameters and multiply them with 0.1 and remained the same fully connected layer parameters. SGD momentum is 0.9. We have tried initial learning rate 0.01, 0.001 and 0.0001 which depends on the training procedure is at beginning or after long training process. And learning rate will change after 5 epoches with gamma 0.1.

2.5 Evaluation Metrics

The selection of metrics used to measure the performance of the model is important because it indicates the direction in which the model is developed. We use Top-1 accuracy for training

process and use Top-1 and Top-3 accuracy for validation process.

3 Model Structure

3.1 Model Selection

Since the introduction of AlexNet[1], many studies have mainly focused on designing new network architechtures for image classification to increase accuracy. For example, new architectures such as Inception, ResNet[2], DenseNet[3], ResnextNet[4] have been proposed. Inception introducd new modules into the network with convolution layers of different kernel sizes. ResNet utilized the concept of skip connection and DenseNet added dense feature connections to boost the performance of the model.[5] In our project, we have tried official ResNet 50, ResNet 101, ResNet 152 net, ResNext 101 net, DenseNet as our training models. Based on transfer learning, we freeze the pre-trained weights of model but change the output features of classifier or fully connected layers to be 251.

For ResNet, the plain baselines are inspired by VGG nets but added residual networks. The total number of weighted layers of ResNet 34 is shown in **Figure. 4** [2]

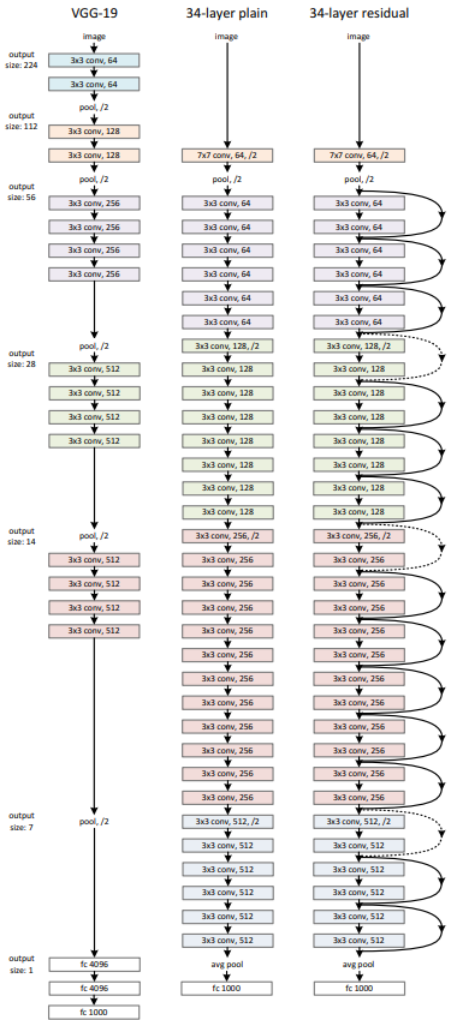


Figure 4: Plain ResNet Structure

Based on the plain network, there are some shortcut connections which turn the network into its counterpart residual version. There are some deeper ResNets which uses the same logic behind ResNet 34. The structure of different ResNets is shown in **Figure. 5**

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5: ResNet Model Structure

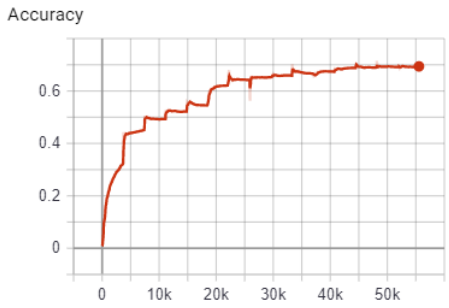
For ResnextNet which means the next dimension, on top of the ResNet. This next dimension is called the ”cardinality” dimension. Compared with ResNet, ResNeXt got 15% improvement of Top-5 error rate. The structure of ResNeXt is shown in **Figure. 6** [4]

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	3×3 max pool, stride 2 $\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		$\mathbf{25.5}\times10^6$	$\mathbf{25.0}\times10^6$
FLOPs		$\mathbf{4.1}\times10^9$	$\mathbf{4.2}\times10^9$

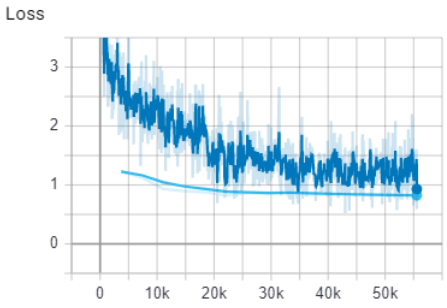
Figure 6: ResNeXt Structure

4 Experiment Results

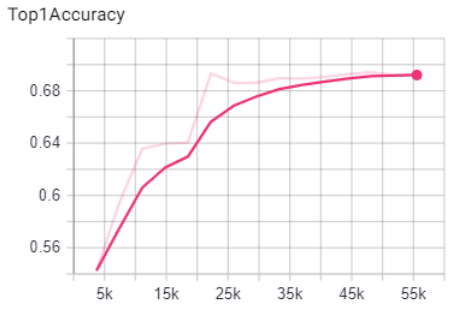
We tried different hyperparameters on different models. During the training process, since we use Google Colab platform, it refresh all memory each 12 hours, so we save our model checkpoints, train, valid loss and accuracy curve into Google drive. Since the epoches are too large that sometimes after one night training, we change the hyperparameters selection because of no progress on accuracy. Here are some curves of training and validation, take ResNeXt 101 training process as an example in **Figure. 7**.



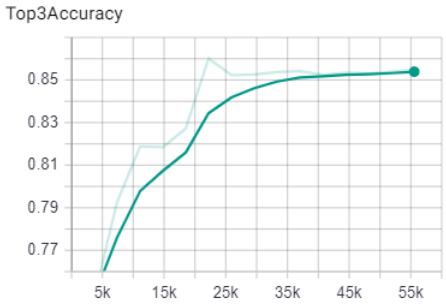
(a) Training top-1 accuracy



(b) Training loss



(c) Validation Top-1 Accuracy



(d) Validation Top-3 Accuracy

Figure 7: Training process of ResNeXt model

We choose the best result of each original model and the error rates of each model is shown in **Table. 1**

Model name	top-3 err. (test)
ResNet 101	14.612
ResNet 152	13.995
ResNeXt 101	13.990
WideResNet 101	14.883
DesNet 161	17.819
DesNet 169	19.170
DesNet 201	18.689

Table 1: Error rates(%) of best model performance

Then we do ensemble phase. With different results we have, we calculate the frequency of labels each image, and take the Top-3 frequent labels as the final results. Since we have a lot of results of models. We choose the pre-result which error rate is less than 20%. Then combine them together. We got the final results which performs the best is 11.603%.

Here is our timeline of kaggle submmision **Figure. 8**.

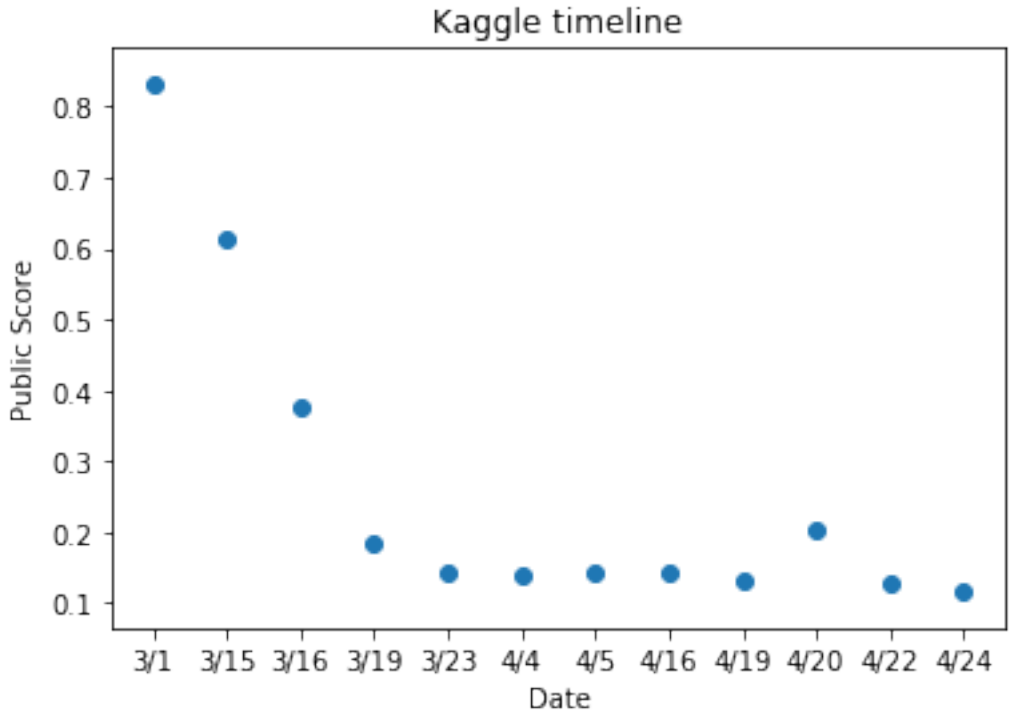


Figure 8: Submission results

5 Conclusion

In this report, we show the processes and results of training on different Convolutional Neural Networks. We assemble the results of ResNet 152, ResNeXt 101 and DenseNet 201. Finally, we reached an 11.603% error rate on test dataset in iFood 2019 FGVC competition and placed the 4^{th} position.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[4] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[5] Jungkyu Lee, Taeryun Won, and Kiho Hong. Compounding the performance improvements of assembled techniques in a convolutional neural network. *arXiv preprint arXiv:2001.06268*, 2020.