Fine-tuned ImageNet model for Chinese Recipe recognition on mobile device

Zou, Zhao Ning
HKUST
znzou@connect.ust.hk

Abstract

Modern health applications usually require the user to record their daily calorie intake through manual input for analysis purposes. Research has shown that manual input sometimes underestimates the actual food take. With the rapid development of automatic image recognition using machine learning, people started to utilize this technology to retrieve recipe labels corresponding to given dish pictures. The current approaches mostly focus on the recognition of food categories based on the global dish appearance. Such approaches could obtain impressive results on western food and food with standard cooking procedures. Moreover, some works utilize additional information, such as ingredient labels, restaurant GPS locations, to improve recognition accuracy further. Currently, few works have been done on Asia or Chinese food recognition. Regarding the training techniques, transfer learning has become more popular nowadays, and many studies have proved its advantages and generalization ability over many tasks. However, there are still little works done on how to train a transfer learning model efficiently. The relationships between the hyperparameters settings, model architecture, and datasets are still not yet fully explored. This paper first analyzed the effect of different hyperparameters and optimizers on transfer learning performance and then presented a lightweight transfer learning model for Chinese food recognition to run on a mobile phone.

1. Introduction

While millions of cooking recipes are posted on the Internet or review applications, find the right recipe name and the corresponding nutrient information given a recipe image remains a challenge yet to be fully explored. The major problem behind this challenge is the recognition of food categories. In health-related applications, users often need to record their food intake for dietary habit monitoring through manual input. In addition to time-consuming, this input process is error-prone. As investigated in [1], self-reporting data tends to underestimate the actual food intake, making the application's health recommendation deviate from reality. With the increasing phone's image post-processing ability, the recognition model can obtain a raw picture closer to the actual food appearance, making the model easier to extract its features. These concerns motivate the use of mobile devices in automatic food recognition [2].

In the literature, food recognition is regarded as a general pipeline that facilitates the estimate of calories and nutrient facts [3] [4]. These works are effective for recognizing western dishes and the recipe with standardized cooking method (e.g., fast food or restaurant food) that often have similar visual appearance and the identical ingredient content. However, most Chinese dishes, especially homemade dishes, have no standardized cooking method, food presentation, or strict ingredient composition. Directing mapping the Chines recipe picture to the name is much more challenging than that for standardized food. What makes the recognition more challenging is that some dishes in Chinese food share slightly different appearance but have totally different names and tastes. For example, Beef curry (咖喱牛肉) and Braised Beef with Potatoes (土豆燉牛肉) share the same brown and yellow color.

In the meantime, transfer learning has become a cornerstone of computer vision. It is believed that high quality pre-trained ImageNet [5] models provide better penultimate layer features that can perform well under a wide range of tasks. Moreover, this makes the model itself having better transferability [6]. Indeed, deep features extracted from DCNN [7], trained on ImageNet, and fine-tuned on food images, obtained exhibit impressive recognition performance [2] [8]. The combination of multi-model features also leads to better food recognition performance, from [9]. The above works inspired us to use transfer learning to solve the Chinese food recognition task.

This paper studied the current approaches for recipe recognition and proposed a small fine-tuned ImageNet [5] model for recognizing Chinese recipes on mobile devices. The model can make a balance between model size and accuracy.

2. Related work

Transfer learning from an ImageNet model has now become the de-facto standard of solving computer vision problems. It is believed that the weights learned on the source dataset with a large number of images provide better initialization for the target dataset than random initialization [10]. According to [6], transfer learning using a good pre-trained ImageNet model helps reduce the time and errors during training and perform better in many different tasks. After transfer learning, fine-tuning techniques are often used further to increase the model's accuracy on the target dataset.

Regarding food recognition, the main challenge comes from visual variations in shape, color, and texture layout. In recent years, variants of recognition approaches have been investigated for different food-related applications. The Deep features extracted from DCNN, which is trained on ImageNet and fine-tuned on food images, often exhibit impressive recognition performance [2]. In [3] [4], associating food categories to their respective recipes is regarded as a general method to facilitate the estimation of calories and nutrition facts. In [9], a combination of multi-model features sometimes also leads to better recognition performance. One of the best performances on UEC Food-100 dataset is achieved by fusion of DCNN features with RootHOG and color moment [11].

Other than recognizing the global appearance of the food, some works have done to retrieve the ingredient formation of the food to assist the prediction [12]. Ingredient information is useful when recognizing unseen food categories, but the model size is still large. Others include utilizing GPS and restaurant menu information in the recognition process. However, restaurant information is sometimes difficult to collect.

3. Datasets

In this project, we used VIREO Food-172 Dataset [11] as our food image dataset. The food categories were compiled from "Go Cooking" and "Meishi", which are two popular websites for Chinese dishes. The dataset contains more than 110K Chinese food images with a total of 172 categories, and each category contains more than 1300 images. The resolution of each image is 256 × 256 × 3. The dataset covers eight major groups of food, and the group meat contains the most significant number of categories. Other groups include vegetables, soup, egg, Seafood etc.. Figure 1 shows some examples of food categories in VIERO Food-172.

Steamed pork with rice powder
Boiled sliced pork in hot chili sauce
Grilled fish
Sautéed shredded pork with skin of tofu
Sautéed diced chicken with chili and pepper
Deep fried lotus root
Sautéed shredded pork in sweet bean sauce
Boiled sliced chicken
Steamed fish
Sweet and sour garlic
Chicken feet with pickled peppers
Roast chicken
Chicken feet with pickled peppers
Deep fried chicken drumsticks
Steamed chicken with chili Sauce
Roast chicken wings
Sauté spicy chicken
Sautéed vermicelli with spicy minced pork
Stewed chicken with mushroom
Stewed chicken with three cups sauce
Shredded chicken
Fish filets in hot chili oil
Sautéed sliced pork, eggs & black fungus
Yu-shiang shredded pork
Shredded preserved egg
Braised beef with potatoes
Deep fried shrimp
Poached egg
Spicy shrimp
Braised beef with brown sauce
Stirred mushrooms & vegetables
Beef curry
Fried lamb with cumin
Rice with Stewed pork
Beef kebabs
Hot and sour potato silk
Griddle cooked tea tree mushrooms
Sautéed preserved pork with garlic sprout
Scrambled egg with okra
Fried beans with eggplant

Figure 1 Examples of food categories in VIREO Food-172

4. Methods

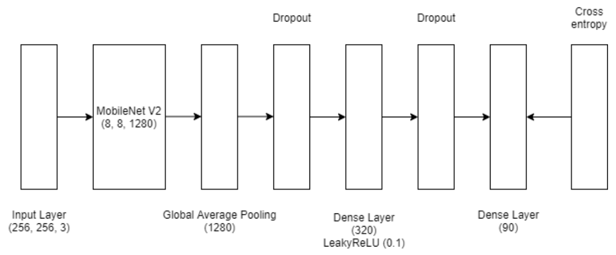
5.1 Architecture Design

The experiments were conducted on VIREO Food-172 dataset. In each food category, 60% of the images are randomly picked for training, while 10% are for validation and the remaining 10% are for testing. For performance evaluation, the average top-1 and top-5 accuracies are adapted for the food recognition, which are standard measures for the single-label task.

Regarding the model's architecture, we formulate food recognition as a deep learning problem and modify the architecture of pre-trained DCNN for our purpose. We selected MobileNetV2 [12] as our base model for image feature extraction. MobileNetV2 is a lightweight model designed for running on mobile devices; therefore, it only has 2.2 million parameters in the convolution layers. After selecting the base model, we need to decide the architecture of the fully connected layers and prediction layer. We have derived three different architectures for the model. The first design (Arch-1) considers adding one hidden layer with 320 neurons and LeakyReLU. Dropout is also used to reduce the overfitting level. The second design (Arch-2) has two hidden layers of 320 neurons and 200 neurons, respectively. The third design (Arch-3) is similar to the first design but with 400 neurons in the hidden layer. At last, all models adapted the sparse categorical cross-entropy loss function to estimate the performance of them.

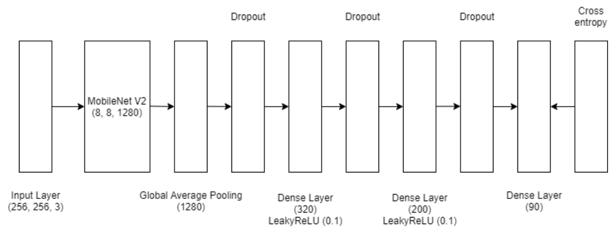
$$J(w) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Since memory space and computation power on mobile devices are limited, the primary consideration here is to obtain a model with fewer parameters. So we compared the increase in parameters with the increase in performance and selected the model with the best performance-cost ratio.



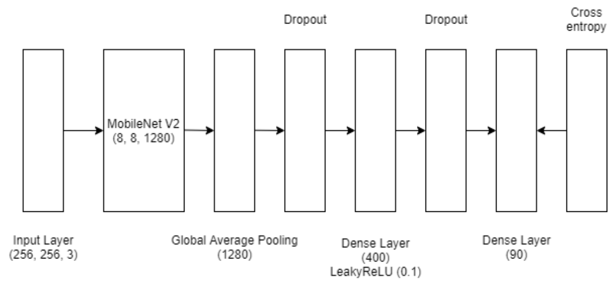
Dropout
Dropout
Cross entropy
MobileNet V2 (8, 8, 1280)
Input Layer (256, 256, 3)
Global Average Pooling (1280)
Dense Layer (320) LeakyReLU (0.1)
Dense Layer (90)

Figure 2 Arch-1 with 2.69M parameters



Dropout
Dropout
Dropout
Cross entropy
MobileNet V2 (8, 8, 1280)
Input Layer (256, 256, 3)
Global Average Pooling (1280)
Dense Layer (320) LeakyReLU (0.1)
Dense Layer (200) LeakyReLU (0.1)
Dense Layer (90)

Figure 3 Arch-2 with 2.75M parameters



Dropout
Dropout
Cross entropy
MobileNet V2 (8, 8, 1280)
Input Layer (256, 256, 3)
Global Average Pooling (1280)
Dense Layer (400) LeakyReLU (0.1)
Dense Layer (90)

Figure 4 Arch-3 with 2.81M parameters

5.2 Optimizer and Hyperparameter Selections

There are many papers discussed the hyperparameter selection for training neural networks from scratch, mostly on batch size, learning rate, and weight decay. However, few works are investigating the selection and relationship between hyperparameters and datasets when doing transfer learning. So, the common practice is to use the default hyperparameters for training large models while using a smaller initial learning rate to prevent destroying the original model. In [10], the paper pointed out the similarity between the original dataset of the model and the target dataset contributes to the selection of the learning rate, momentum, and weight decay in SGDM. According to this paper, we should select a larger learning rate and momentum to obtain better convergence since our food dataset is far apart from the ImageNet dataset.

Apart from using SGDM, Adam [13] is another famous optimizer that is often seen in research. Adam performs local optimization with a metric constructed from the iteration history; thus, it can change during the training process adaptively. In [14], The author discovered that in transfer learning, the models trained using these adaptive methods, including Adam and RMSprop, generalize worse than stochastic gradient descent, even though these solutions have better training performance. Therefore, our project examined the performance of SGDM and Adam and the effect of different learning rates.

Apart from optimization, overfitting is another problem that we need to tackle in our project. Transfer learning often suffers from overfitting problem because the target dataset's size is small compared to the original dataset, and the pre-trained model has high adaptability to any data intake. Therefore, we selected a higher dropout rate, say 0.4, for our model.

We also need to set the number of layers to unfreeze during the fine-tuning process. The increased number of unfreezing layers may cause overfitting. Therefore, We tested a range of models with a different number of unfreezing layers to obtain the best model among them.

5. Experiments

We split the experiment into three parts. First, we compared the performance of SGDM and Adam on transfer learning and fine-tuning. We hope to obtain similar observations as in the previous papers. Then we used the best optimizer and hyperparameters we found in the first part to train three models with architecture stated previously to identify 90 food categories. Finally, we analyzed the observations from the previous parts and constructed our final transfer learning model to identify all food categories in the dataset.

6.1 Transfer Learning Comparision

In this section, we only trained the fully connected layers of the model. We selected Arch-1 as our testing model and predict on 40 food categories. The hyperparameters used here are learning rate (0.05, 0.01, 0.005, 0.001) and momentum (0.9). We set other default hyperparameters to be batch size 256, LackyReLU alpha 0.1, dropout rate 0.4. Here are the training results:

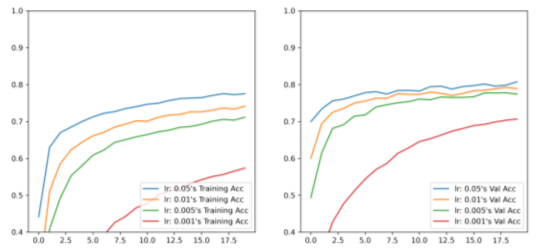


Figure 5 SGDM 0.4 Dropout Results Train/Val

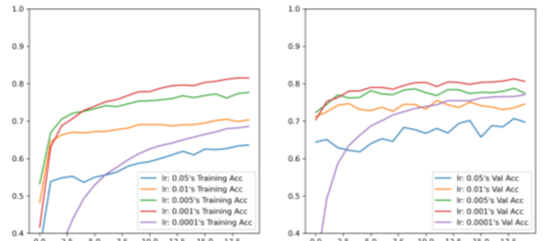


Figure 6 Adam 0.4 Dropout Results Train/Val

SGDM LR	0.05	0.01	0.005	0.001	
Train Acc	0.775	0.742	0.712	0.574	
Val Acc	**0.808**	**0.789**	0.774	0.707	
Adam LR	0.05	0.01	0.005	0.001	0.0001
Train Acc	0.636	0.704	0.777	0.815	0.686
Val Acc	0.697	0.746	**0.774**	**0.806**	0.771

Regarding the SGDM results, we observed that SGDM could work well under a larger learning rate. The convergence speed is directly proportional to the learning rate. A larger learning rate, say 0.05, did not cause any divergent behavior. We believed this is mainly due to the model's simple architecture at the top part. Since we are only training the fully connected layers on top of our model, the loss map might be less complicated than training the whole model. We also observed better generalization ability from SGDM. The difference between the training accuracy and validation accuracy are mostly 2-3% apart.

Regarding the Adam results, we observed that Adam had the best performance under a small learning rate of 0.001. With a large learning rate, say 0.05 or 0.01, Adam performs much worse than the other models. Given a larger learning rate and Adam's ability to stabilize the step size when the calculated gradient is small, the real step size of Adam might be even larger than expected. So, this may cause divergent behaviors, and the model cannot converge to the correct minimum point.

Lastly, based on the result of SGDM and Adam, we came out with two observations. We found that a larger learning rate on SGDM will not cause divergent behaviors but will make the model converge in fewer iterations. This is a similar finding in [10]. Also, the SGDM models generalize slightly better than the Adam models, similar to the results from [14].

6.2 Fine-Tuning Comparison

Based on the model performance in the previous section and early trials on fine-tuning. We selected two models obtained from the previous section to conduct fine-tuning: SGDM with a learning rate of 0.01 and Adam with a learning rate of 0.001. In this section, we further trained each of them on a range of number of unfreezing layers and evaluated their performance. The fine-tune starting layers on the base model MobileNetV2 were set to be 82, 91, 99, 108, 117, and 126. The smaller fine-tune starting layers mean more layers on top are unfrozen for fine-tuning. We also reduced the learning rate to 1/10 of the initial learning rate to not destroy the original ImageNet model. Here are the training results:

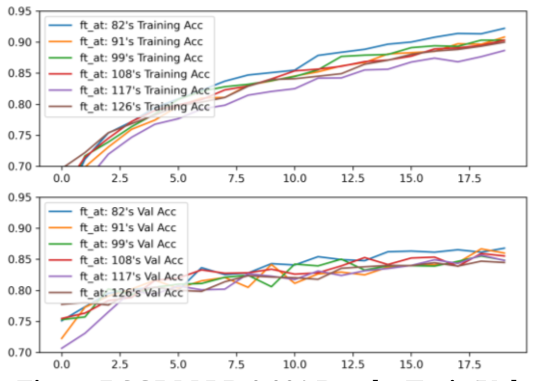


Figure 7 SGDM LR 0.001 Results Train/Val

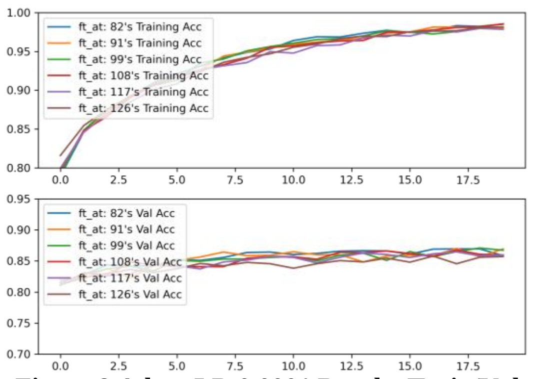


Figure 8 Adam LR 0.0001 Results Train/Val

SGDM LR 0.001						
	82	91	99	108	117	126
Train Acc	0.922	0.908	0.903	0.903	0.886	0.900
Val Acc	**0.868**	0.860	0.848	0.855	0.848	0.845
Adam LR 0.0001						
	82	91	99	108	117	126
Train Acc	0.985	0.980	0.985	0.986	0.978	0.982
Val Acc	0.859	**0.869**	0.867	0.860	0.860	0.857

We found that models using SGDM were not able to converge within the pre-defined number of iterations, so we have trained them for more iterations. Since the model ranking did not change, we did not present their accuracy graph. From the SGDM results, we found that the number of unfreezing layers is directly proportional to the overfitting level, as expected. Moreover, the model with layers unfreezes from layer 82 achieved the best performance. We also observed that the accuracy difference across the different number of unfreezing layers is only 1-2%, which means the parameters update might have mainly happened in the last part of the base model. So unfreezing more layers did not make the model changed much.

Models using Adam as optimizer converge quicker than SGDM, but the overfitting problem is also more significant. Unfreeze at 91 layers got the best result in Adam, but the accuracy difference across the different number of unfreezing layers is only about 1%. This might prove Adam's ability to converge under any model architecture, so selecting the number of unfreezing layers is not as careful as in SGDM.

Based on the comparisons between the two optimizers in transfer learning and fine-tuning, we found that the adaptive method such as Adam converges better on the training set than SGDM, but it also has a worse generalization performance. This is equivalent to the findings in [14]. These results suggest the practitioners consider to use more SGDM optimizer when conducting transfer learning.

6.3 Model Architecture Comparison

Based on the observations from the previous parts, we selected SGDM with learning rate 0.01, momentum 0.9 as our optimizer setup for the architecture comparison. Moreover, we chose to fine-tune start from the 82nd layer of the base model. We trained these models on 90 food categories. The transfer learning part cost 30 iterations, and the fine-tuning part cost another 50 iterations. Here are the results:

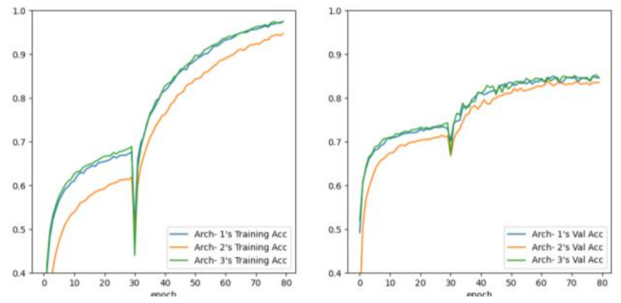


Figure 9 Architecture Comparision

	Arch-1	Arch-2	Arch-3
Train Acc	0.9748	0.9468	0.9744
Val Acc	**0.8460**	0.8353	0.8456
Test Acc	**0.8414**	0.8304	0.8407

From the accuracy graph and table above, we could observe that Arch-1 and Arch-2 performed better than Arch-2 in both the training and validation part. Arch-2's two hidden layers structure didn't help the model to gain a performance advantage compared to the one hidden layer structure in Arch-1. Meanwhile, Arch-1 has fewer parameters in the full-connected layers than Arch-2. The second hidden layer in Arch-2 might not utilize or combine the knowledge learned from the first hidden layer and present the most valuable features to the prediction layer. So the predictions from Arch-2 were even worse than the predictions which direct utilized the information from one hidden layer. Comparing the result of Arch-1 and Arch-3, which both of them used one hidden layer, the model with more neurons in the hidden layer (Arch-3) also did not gain a noticeable performance increase. This might mean that the additional neurons did not help the model to exploit more features from the convolution layer. However, we only experimented on 90 food categories; these additional neurons may be helpful if the total number of food categories is larger than 90, say 150 or 172. More features will be taken into consideration when the dataset is more diverse.

6. Final model

In this section, we have trained the final model to recognize all food categories from the VIREO Food-172 Dataset. Based on the findings in the previous sections, we selected Arch-3 as the base structure and modified the prediction layer to match the categories in the whole dataset. The transfer learning part cost 30 iterations, and the fine-tuning part cost 50 iterations. Here is the accuracy and loss graph.

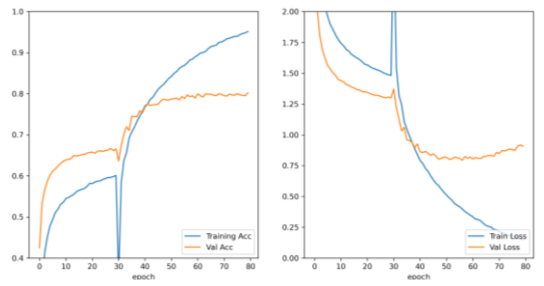
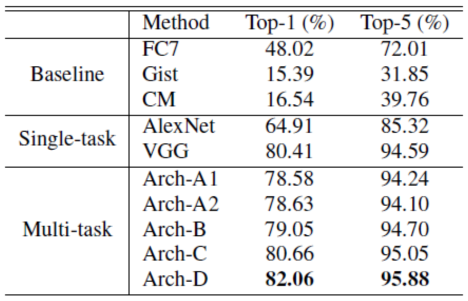


Figure 10 Final Model

Finally, we obtained a Chinese food recognition model with around 80% top-1 accuracy and 94.6% top-5 accuracy. The overfitting problem is still obvious, with around 10% accuracy difference between training and validation. But in the meantime, this model only contains 2.84 million parameters. We are surprised that transfer learning on this efficient and light-weighted MobileNetV2 model could obtain results as good as more extensive networks like VGG16. Here is the comparison with the models obtained from [12], from the author of the dataset.

Our model		
Train Acc	Test Acc (Top1)	Test Acc (Top5)
0.9513	0.7971	0.9462



	Method	Top-1 (%)	Top-5 (%)
Baseline	FC7	48.02	72.01
	Gist	15.39	31.85
	CM	16.54	39.76
Single-task	AlexNet	64.91	85.32
	VGG	80.41	94.59
Multi-task	Arch-A1	78.58	94.24
	Arch-A2	78.63	94.10
	Arch-B	79.05	94.70
	Arch-C	80.66	95.05
	Arch-D	**82.06**	**95.88**

7. Conclusions

We have presented three main pieces of our work: the optimizer and learning rate comparison of SGDM and Adam, an architecture comparison, and a light-weighted Chinese food recognition model capable of running on mobile devices.

Similar to some previous findings, SGDM with a larger learning rate could obtain faster convergence during training without causing divergent behaviors. SGDM also has a better generalization ability, which could keep the overfitting gap small. This property is especially useful in transfer learning because the target dataset for learning is usually small. So this could help prevent the model from learning too much about the noise in the training set.

Regarding another widely used optimizer named Adam, we found that Adam is strongly adaptive to many different architectures. The performance on the training set was usually better than SGDM. In the fine-tuning part, although testing models had a different number of unfreezing layers, their training performances were still quite the same to each other. This proved Adam's ability to converge any many situations

While the results are encouraging, the current work is worth further investigation in two directions. First, we can further explore the reasons behind SGDM and Adam's different performance on transfer learning. We currently only presented this observation but could not give a reliable explanation for this. The second improvement is about the food recognition model, which we could train the model on more food images. We could also compare the performance using different base models, such as ResNet [16] or DenseNet [17]. Finally, we would like to thank again to Dr. Chen Jing Jing for providing the dataset for this project.

References

[1] A. H. Goris, “Undereating and underrecording of habitual food intake in obese,” in *The American journal of clinical nutrion*, 2000.

[2] N. J. V. R. A. Meyers, “Im2calories: towards an automated mobile vision food diary,” in *IEEE International Conference*, 2015.

[3] J. Y. W. Wu, “Fast food recognition from videos of eating for calorie estimation,” in *IEEE International Conference*, 2009.

[4] K. Y. Y. Kawano, “Real-time mobile food recognition system,” in *Computer Vision and Pattern Recognition Workshop*, 2013.

[5] L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *CVPR09*, 2009.

[6] S. Kornblith, "Do Better ImageNet Models Transfer Better," 2019.

[7] I. A. Krizhevsky, “Imagenet classification with deep convolutional neural networks,” 2012.

[8] K. Y. a. Y. Kawano, “Food image recognition using deep convolutional network with pre-training and fine-tuning,” in *IEEE International Conference*, 2015.

[9] D. K. X. Wang, “Recipe recognition with large multimodal food dataset,” in *In Multimedia & Expo Workshops*, 2015.

[10] H. Li, “Rethinking the Hyperparameters for Fine-tuning,” 2019.

[11] C.-w. N. Jing-jing Chen, “Deep-based Ingredient Recognition for Cooking Recipe Retrival,” in *ACM Multimedia*, 2016.

[12] M. Sandler, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” 2018.

[13] D. P. Kingma, “Adam: A Method for Stochastic Optimization,” 2015.

[14] A. C. Wilson, “The Marginal Value of Adaptive Gradient Methods in Machine Learning,” 2017.

[15] S. L. Smith, “A bayesian perspective on generalization and stochastic gradient,” in *ICLR*, 2018.

[16] K. He, “Deep Residual Learning for Image Recognition,” 2015.

[17] G. Huang, “Densely Connected Convolutional Networks,” 2016.

[18] J. B. Diederik P. Kingma, “Adam: A Method for Stochastic Optimization,” 2014.

Appendix: Part of the recognition results (172 categories)

The label on top is the true label, the labels in the table are the prediction results.

52

1	52	0.90315
2	38	0.09685
3	53	0.0
4	61	0.0
5	60	0.0

18

1	18	0.99549
2	41	0.00256
3	103	0.00036
4	5	0.0003
5	149	0.0002

115

1	115	0.98946
2	153	0.00791
3	53	0.00228
4	134	0.00034
5	57	1e-05

112

1	112	0.99989
2	113	8e-05
3	134	2e-05
4	160	1e-05
5	52	0.0

120

1	61	0.3785
2	133	0.18187
3	108	0.1764
4	125	0.08091
5	111	0.03877

19

1	48	0.42183
2	127	0.34797
3	12	0.11904
4	19	0.07025
5	42	0.01562

-115

1	141	0.99998
2	5	1e-05
3	171	0.0
4	53	0.0
5	60	0.0

19

1	19	1.0
2	171	0.0
3	64	0.0
4	62	0.0
5	61	0.0

74

1	81	0.95757
2	74	0.01605
3	124	0.0106
4	160	0.00646
5	8	0.00276

33

1	33	0.99732
2	20	0.00252
3	122	8e-05
4	94	4e-05
5	32	2e-05

85

1	85	0.69486
2	99	0.13944
3	129	0.07839
4	109	0.02509
5	91	0.02496

110

1	110	0.99831
2	60	0.00084
3	125	0.00035
4	133	0.00011
5	0	6e-05

-114

1	71	0.66137
2	142	0.33532
3	170	0.00169
4	145	0.00133
5	137	0.00021

114

1	134	0.477
2	156	0.3317
3	114	0.12603
4	115	0.02532
5	109	0.01374

34

1	34	0.97415
2	0	0.02555
3	117	0.00017
4	118	4e-05
5	123	4e-05

9

1	9	0.99995
2	137	3e-05
3	107	1e-05
4	135	1e-05
5	52	0.0

70

1	90	0.94797
2	27	0.01486
3	70	0.0147
4	15	0.0121
5	98	0.00696

106

1	106	0.99999
2	117	1e-05
3	53	0.0
4	61	0.0
5	60	0.0

21

1	21	0.53739
2	25	0.44937
3	120	0.00419
4	125	0.00213
5	108	0.00197

123

1	123	0.4528
2	98	0.26803
3	161	0.09982
4	0	0.08268
5	129	0.05388

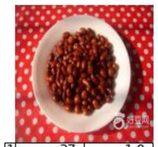
103

1	103	0.71408
2	138	0.16896
3	43	0.07909
4	160	0.03093
5	124	0.00326

-113

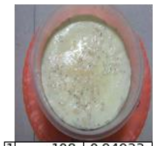
1	19	0.93445
2	69	0.06319
3	161	0.00118
4	143	0.00056
5	123	0.00018

37



1	37	1.0
2	171	0.0
3	53	0.0
4	61	0.0
5	60	0.0

108



1	108	0.94933
2	126	0.02288
3	162	0.01455
4	81	0.00923
5	138	0.00126

97

1	97	0.86024
2	149	0.12565
3	140	0.01025
4	167	0.00266
5	164	0.00075

111

1	111	1.0
2	171	0.0
3	53	0.0
4	60	0.0
5	59	0.0

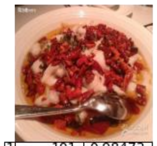
-124

1	67	0.95668
2	101	0.02608
3	14	0.01144
4	132	0.00569
5	103	5e-05

43

1	14	0.70512
2	67	0.15312
3	48	0.09003
4	101	0.02249
5	40	0.01654

101



1	101	0.98473
2	103	0.0152
3	104	6e-05
4	67	1e-05
5	171	0.0

-95

1	161	1.0
2	171	0.0
3	53	0.0
4	61	0.0
5	60	0.0

127

1	127	0.68664
2	123	0.29906
3	40	0.00254
4	25	0.0024
5	43	0.00237

40

1	40	0.99999
2	14	1e-05
3	171	0.0
4	54	0.0
5	61	0.0