Identifying States of Cooking Objects Using VGG Network

Tianze Chen

***Abstract*—Object recognition has been a long time problem since computer vision started to develop. In robotic learning, recognizing the state of an object for the robot is very important. In this paper, we are going to discuss the architecture of the VGG deep learning network to help with the state recognition problem since state recognition in nature is an image classification problem. The main work of this paper is to midify the structure of VGG to investigate if the accuracy of state classification could be improved together with some other modifications on the convolutional layers as well as the fully connected layers.**

keywords: VGG, Convolutional Layers, Learning Rate, Optimizers

I. Introduction

THE goal of artificial intelligence is to make robots imitate or even replace humans to do the complicated work. In all of the potential fields, cooking is espically important since it is related to our everyday life. In order to make a robot learn cooking, there are several important steps: grasping, manipulation, object recognition, state recognition as well as image and video understanding. For example when cutting a carrot, the robot should grasp the knife from a shelf and hold the knife in the correct direction. Then the robot may search videos on the internet to find out how to do the cutting through watching the videos. Furthermore, strength and the cutting angle are important when doing the cutting, which relates to manipulation. Eventually the robot should recognize the object (e.g. carrot) and identify the state (e.g. sliced carrot) to guarantee the whole procedure.

In robotics, objects at different states would require different grasping strategeries. To achieve different states, different manipulations would be required. Object recognition played an important role in the work of[1], [2], [3], [4], [5], [6]. Methods on telling the states of the object and learning the relative motion from videos and images are shown in[7], [8], [9], [10].

Of all the aspects, this paper focuses on a small part of the state recognition problem, which is classification. Classification mainly helps to tell what sate an object is in. For my network the VGG19[11] was used as a baseline to try to solve a simple 7 class recognition problem. The model which was generated achieved an accuracy of 80% on average on the testing dataset. In the following sections the methods used to reach the final model and the experiments done on different settings related to the model will be stated.

Course Name: Neural Networks/Deep Learning. The work is supervised by Professor Yu Sun and TA Ahmad Babaeian Jelodar .

II. Data and Augmentation

In this project, the data includes seven different classes: creamy_paste, diced, grated, juiced, julienne, sliced and whole with 18 cooking objects(tomato, onion, bread, pepper, cheese, etc). In our work, dataset version 1.0 of the state recognition challenge from[12][1] was used.

The annotation on the dataset was first done to classify an image to one of the seven classes. After randomly checking other people's annotation, a dataset of the size 5117 with labels was created. In the data 80% was randomly seperated as training and the rest was regarded as validation data. Since 5117 samples are too small for such a complicated structure, in order to make the better use of the dataset, several augmentation methods were added: rescaling, rotation, width and height shiftting, shear and zoom, horizontal and vertical flipping. The corresponding factors are listed in Table I.

TABLE I: Overview of data augmentation

Name	Augmentation Factors
Rescale	1/255
Rotation	30
Width Shiftting	0.5
Height Shiftting	0.5
Shear	0.2
Zoom	0.2
Horizontal Flipping	True
Vertical Flipping	True

III. Methodology

When designing network layers, the following aspects were investigated: the size of the output of the convolutional layers, the size of the filters of the convolutional layers, the dropout factor, the optimizer and the learning rate. These are the main factors that can influence the behavior of the network. The explainations of the detail of each aspect will be shown in the following subsections. The architecture of the model is shown in Fig. 1.

The total number of convolutional layers was 19 because in the paper of VGG, the writers mentioned that it could give the best accuracy when the number of convolutional layers was between 16 and 19. The weight of the model was loded from the pretrained VGG19 on Imagenet.

A. The convolutional Layers

The original convolutional layer has the filter size of 3×3 and the last four convolutional layers have 512 filters. The last

[1]The dataset is available at here..

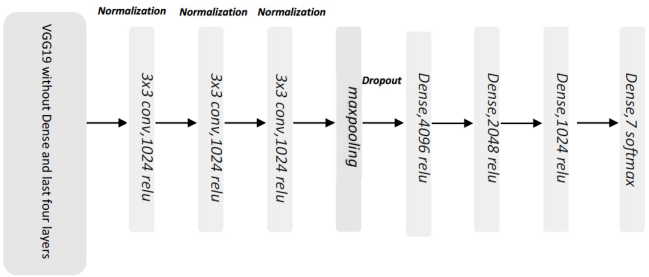


Fig. 1: The Architecture of the Modified Model

four convouluntional layers of the original VGG network were frozen and different filter sizes with different number of filters were tried.

B. The Dropout Factor

Since the total structure was too complicated for the data, dropout was added to decrease the chance of having overfitting. The dropout rate was 0.5.

C. Learning Rate

Learning rate can influence the learning efficiency. Choosing the proper learning rate can help to achieve a good result. After discovering the region for different optimizers a decay was added to the learning rate. The reason of doing so was to decrease the learning step when the training approached its optimal solution. The corresponding learning rate regions can be found in Table II.

D. Optimizers

The most commonly used optimizers for CNN are SGD and Adam. Also there are some other optimizers that are available such as Adagrad, Adadelta, Adamax and Nadam. A test of different optimizers was set to evaluate the result and then the best optimizer was chosen. To make things easier, only the factors of learning rate and decay were used even though for different optimizers there are other available factors that can influce the training result. The factors are listed in Table II.

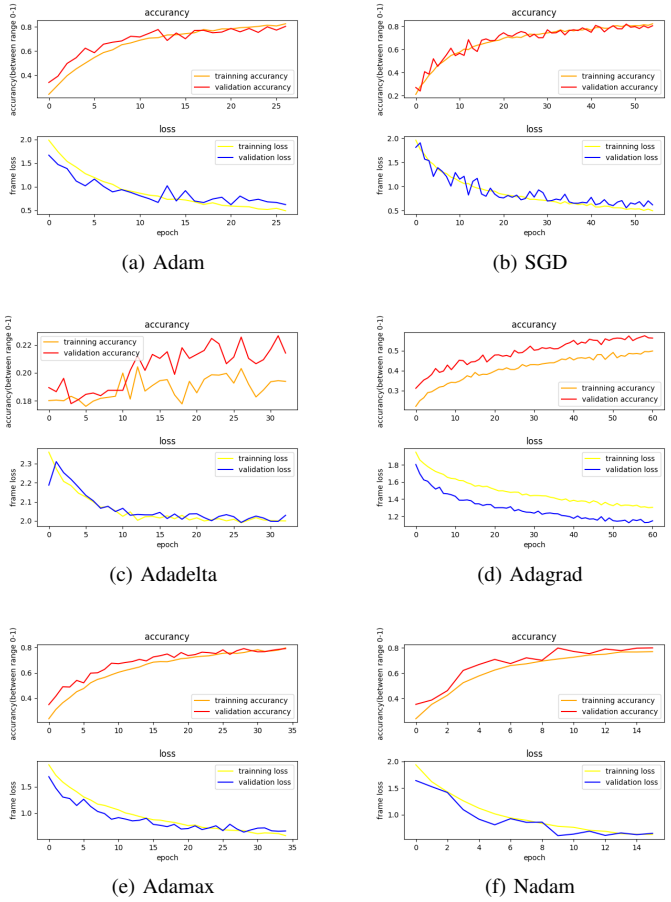
TABLE II: Overview of data augmentation

Name	Learning Rate	Learning Rate Decay
SGD	10^{-3}	9^{-5}
Adam	10^{-5}	1.8^{-7}
Adadelta	10^{-5}	1.8^{-7}
Adagrad	10^{-5}	1.8^{-7}
Adamax	10^{-5}	1.8^{-7}
Nadam	10^{-5}	1.8^{-7}

IV. Evaluation and Results

A. Results of Different Optimizers

The training results of the optimizers together with the corresponding factors listed in Table II will be presented in Fig. 2. As we can see, different optimizers gave different



(a) Adam (b) SGD

(c) Adadelta (d) Adagrad

(e) Adamax (f) Nadam

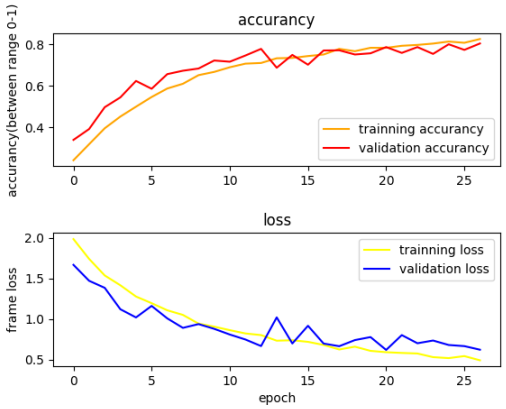
Fig. 2: Training Result of Different Optimizers

results. The SGD model in (b) performed well but the total time, here was epoch, was much longer compared with the Adam optimizer in (a). For (c), the Adadelta optimizer, the training and validation accuracy were pretty low and it took the model 30 epochs to stop at the accuracy of 22%. The main reason is the learning rate was not in the proper region and the architecture also needed some more adjustments, which was a weakness compared with (a) and (b) since more epochs would be needed. In (d), the Adagrad optimizer, the validation accuracy and losses are always higher than the training values, which means the current model was not complicated enough for the optimizer. Since the structure was already deep enough for the dataset, there was no need to make the architecture more complicated. For (e), the Adamax optimizer, the result looked similiar to (a), but it took many more epochs to reach its goal under the same condition. And (f) had the same problem as (d) on the complexity of the model.Thus Adam optimizer was chosen.

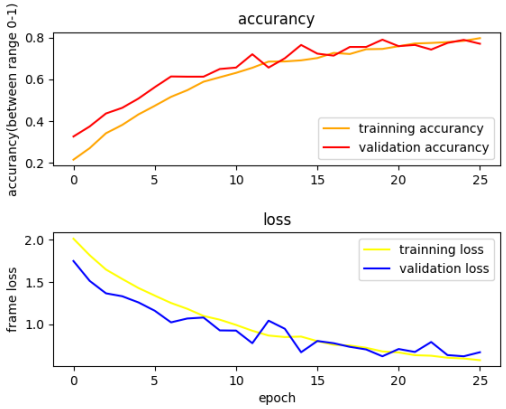
B. Size of the Convolutional Layers

Filter size of 3×3 and 7×7 were tried, together with the filter number of 512 and 1024 to figure out what the result would be. The result will be presented in Fig. 3. And the corresponding factors of the convolutional layers are listed in

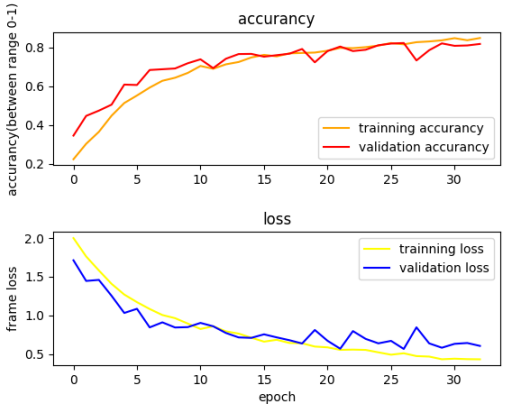
Table III.



(a)



(b)



(c)

Fig. 3: Training Result of Different Convolutional Layers

TABLE III: Overview of data augmentation

Label in Figure 2	Filter Size	Filter Number
(a)	3×3	1024
(b)	7×7	1024
(c)	3×3	512

From the result we can find out that the difference between training reslut was not that significant. The reason mainly because only the last three convolutional layers of VGG19 were changed instead of the whole layers' structure. Based on the result, the convolutional layer with the filter size of 3×3 and filter number of 1024 were chosen.

C. Model Finalization and Testing

The last four layers in the block 5 of VGG19 were frozen during the training and then three more self-designed convolutional layers which has 1024 filters with the size of 3×3 were added. After that a dropout layer with the ratio of 0.5 was added before the fully connected layers. Two different approachs were compared: the first one is to train the model with earlystopping, the second is to train the model 100 epochs to overfit the model. The training results are shown in Fig. 4 and Fig. 5.

From Fig. 5 we can see that the training start to overfit around the 40th epoch, and the validation accuracy continued to be around 80%, which is the same as it shows in Fig. 4. But when these two models were tested, a higher accuracy of 82% was achieved on Fig. 5's model which is overfitted compared with the 78% on Fig. 2's model which used decay in training. This maybe because the in the decay method, the epochs set to stop the training was large and the model stopped training too early. From the confusion matrix we can see that the states of sliced and whole have the most misprediction with each other. The main reasons are as following:

- The sliced and whole state of some objects such as tomato and egg can be very similar. Sometimes we may not be able to tell the difference.
- Even though the sliced and whole dataset have the most samples of all the seven classes, the number of the samples are still too small for the model to learn the features of the two classes.

More information can be gained from Fig. 6.

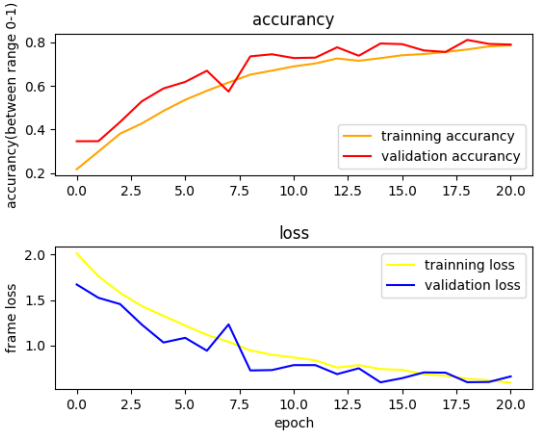


Fig. 4: Training with earlystopping

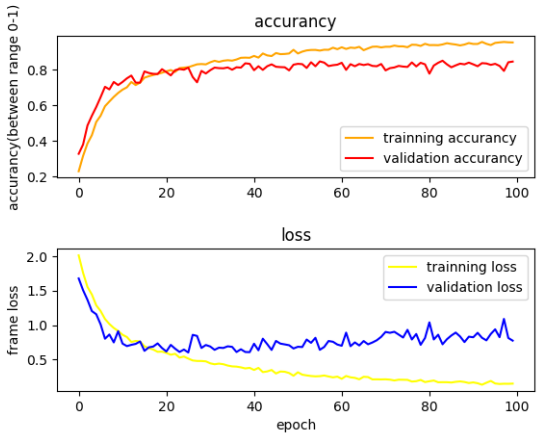


Fig. 5: Training 100 epochs

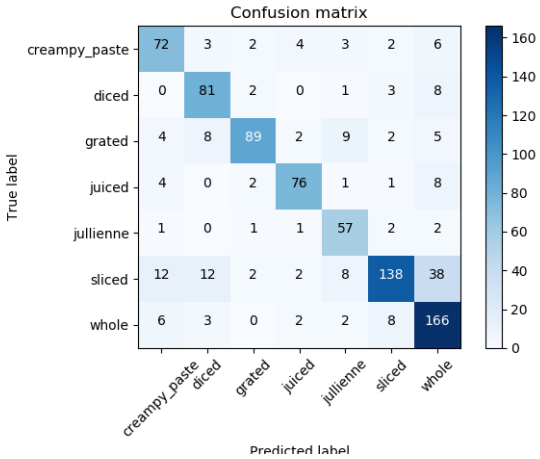


Fig. 6: Confusion Matrix

V. DISCUSSION

In this project a modified version of VGG19 was used to do the training and validation on the classification problem. Some issues that may be related to the result of the model were found.

The first issue is that it is easy for the model to get overfitted during training, such as overfitting in only 20 epochs. That is because the dataset's size is not large enough. The model can 'memorize' some of the features in the samples.

The second issue is that differences in some states are hard to tell in the images, such as creampy_paste and grated. That may cause prediction errors which we can see in the confusion matrix in Fig. 6.

The third issue is the complexity of VGG19. It is a huge and deep network, even though the last four layers of block 5 in the original VGG19 network were frozen, the newly added layers could not have effect on the whole model.

REFERENCES

[1] Y. Lin and Y. Sun, "Grasp planning based on strategy extracted from demonstration," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 4458–4463.

[2] ——, "Robot grasp planning based on demonstrated grasp strategies," *The International Journal of Robotics Research*, vol. 34, no. 1, pp. 26–42, 2015.

[3] ——, "Task-oriented grasp planning based on disturbance distribution," in *Robotics Research*. Springer, 2016, pp. 577–592.

[4] Y. Sun, Y. Lin, and Y. Huang, "Robotic grasping for instrument manipulations," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2016 13th International Conference on*. IEEE, 2016, pp. 302–304.

[5] Y. Lin and Y. Sun, "Grasp planning to maximize task coverage," *The International Journal of Robotics Research*, vol. 34, no. 9, pp. 1195–1210, 2015.

[6] ——, "Task-based grasp quality measures for grasp synthesis," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 485–490.

[7] Y. Lin, S. Ren, M. Clevenger, and Y. Sun, "Learning grasping force from demonstration," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1526–1531.

[8] Y. Sun, S. Ren, and Y. Lin, "Object–object interaction affordance learning," *Robotics and Autonomous Systems*, vol. 62, no. 4, pp. 487–496, 2014.

[9] Y. Sun and Y. Lin, "Modeling paired objects and their interaction," in *New Development in Robot Vision*. Springer, 2015, pp. 73–87.

[10] D. Paulius, Y. Huang, R. Milton, W. D. Buchanan, J. Sam, and Y. Sun, "Functional object-oriented network for manipulation learning," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2655–2662.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] A. B. Jelodar, M. S. Salekin, and Y. Sun, "Identifying object states in cooking-related images," *arXiv preprint arXiv:1805.06956*, 2018.