Cooking Object's State Classification using Fine-Tuned VGG Model

Mathankumar Venkateshwaran

Department of Computer Science and Engineering, University of South Florida

Abstract*— Building a Robotic cook still face tremendous challenges in basic physically-interactive manipulations and task-oriented grasping. To do different manipulation on objects they need to have knowledge of those states. This paper focuses on the analysis of cooking object state's classification based on the dataset of cooking object annotated as 11 different cooking states (like diced, sliced, etc.,). The data set is then divided into training, validation and testing. A pretrained VGG19 model is used as a base model and is trained with training data along with validation dataset for the model validation to prevent from overfitting. The model is then fine-tuned for our problem of classification of cooking object's state. An untouched test data is used to find the models performance. This model can further be tuned and used for other object state classification.

Keywords*— Cooking object state classification, Convolutional Neural Network, Pre-Trained Model, Fine-Tuning, Optimizer.

I. INTRODUCTION

Cooking process goes through manipulation of different objects from one state and other until the final recipe is prepared. Inorder to build a robotic cook we need to make it recognize and classify those states distinctly so that it can comprehend that its manipulation of that object to that particular state is successfully achieved or to be sure of either how to proceed with the object for a specific recipe. This drives the work on the classification of cooking object's state. There are lots of work going on in image classification, which can be used as a base for the state classification. Especially, Image Net Large Scale Visual Recognition Challenge (ILSVRC) in 2014 has brought very good models for the image classification like VGG, Inception, RESNET, Alexnet, etc., which are then used for the various purpose, adapted, updated[13].

In this paper, we are discussing the reports of the experiments and result obtained through the Fine-tuned VGG19 model. The fine-tuning involve the inclusion of additional layers to help the classification even further along with the model. The given datasets are used to train and fine-tune the model to classify the cooking objects state. Different techniques have been experimented on the dataset like data augmentation, preprocessing, batch normalization, dropout, optimizers to better the classification accuracy.

II. METHODOLOGY

A. Dataset

Dataset used for this experiment consists of cooking objects belonging to 11 different classes each represents different state of object in cooking manipulation. They are namely creamy paste, diced, floured, sliced, mixed, peeled, grated, julienne, whole, juiced and other. The dataset is divided into training and validation, where the training dataset consisted of 6348 images and validation set had 1377 images.

B. Data Processing

Building a deep neural network involves building a sequence of layers that process the data to find a feature and patter in them to establish or form a rule from which further related data can be comprehended. Inorder for the layers make more sense of given data it has to be preprocessed to certain input format, for which different techniques were employed.

Preprocessing of the input is the initial technique to process to make the data bring out its behavior and remove its inconsistency throughout the dataset.

Data augmentation is technique that helps us when the dataset is small for our training it will help create additional data from existing dataset by flipping them horizontally or vertically, shifting the range, rotation, scaling etc., which would help training the model better.

III. EXPERIMENT

For the experiment, we have used VGG19 pretrained model as a base model and applied different techniques on top of the model to fine-tune and adapt to our problem of cooking object state classification. VGG uses Convolutional neural network(CNN) for the extraction of the basic block of feature information from the image which play a vital role in identifying each image distinctly[5][21], different kind of CNN layer with the use of different kind of kernel and other attributes, can be adjusted to extract different kind of features from the image which are then can be used for the different problems[19].

The VGG19 model we are using has lots of such CNN layer which got rained on Imagenet data on so the layers are trained to optimum weights for the image classification, which would help as the knowledge can be used for our problem of state classification by adapting and adding additional layers to it, this type of machine learning is called transfer learning.

.

A. *Model Architecture I*

Base model layer of VGG19 was kept along with weights for most layer of it since the bottom layer in a model always do the feature extraction and it is same for most of the model of the same or related problems. For this model we have removed the topmost layer of base model and added a CNN layer to further more adapt the feature extraction for our problem followed by which we have added global averaging to reduce dimension for the following two dense layers and finally converge it to classify 11 classes, which is clearly shown in the Fig1.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv4 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv4 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv4 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
New_conv (Conv2D)	(None, 4, 4, 512)	2359808
batch_normalization_1 (Batch	(None, 4, 4, 512)	2048
activation_1 (Activation)	(None, 4, 4, 512)	0
max_pooling2d_1 (MaxPooling2	(None, 2, 2, 512)	0
global_average_pooling2d_1 (	(None, 512)	0
dense_1 (Dense)	(None, 2048)	1050624
batch_normalization_2 (Batch	(None, 2048)	8192
activation_2 (Activation)	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_2 (Dense)	(None, 128)	262272
batch_normalization_3 (Batch	(None, 128)	512
activation_3 (Activation)	(None, 128)	0
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 11)	1419

Total params: 23,709,259
Trainable params: 3,679,499
Non-trainable params: 20,029,760

Fig1. Model Architecture I

Batch normalization is used after CNN[16] and both of dense layers we have added to normalize feature values which help us in training speed[10]. Similarly, the activation function of ReLu is also added after each of the above-mentioned layers. Inorder to prevent the model from overfitting we have added the layer of Dropout after each additional CNN and the dense layer added.

Model is then compiled with optimizer SGD using categorical cross-entropy as loss function along with momentum with 0.001 learning rate to train on the augmented training dataset while being monitored with early stopping callback to prevent overfitting and also checkpoint callback is used to save the best model for validation accuracy[1][3].

All these were done while freezing the base model completely so that it won't train for this first iteration of the epoch with a maximum limit to 200. For the next iteration, only the first seventeen layer of the base model is frozen so that it can be now trained along with the newly added layer to fine-tuned for our problem.

In this iteration, the model is compiled with optimizer of Adadelta and allowed to train furthermore on the data while validation data is used for validating the model with callbacks. The use of SGD and adadelta is that because SGD is used to find the global minima and then the adadelta can adapt the learning rate based on the problem and can arrive at local minima. Epoch used for the second iteration is 50. And based on the checkpoint best model of validation accuracy is saved and used for the test the model performance using the separate test data.

B. *Model Architecture II*

In this model, we did the same two iterations same epoch with some changes to additional layers to fine-tune base

model from the previous architecture. Apart from the removal of the top layer from base model as earlier, we have added flatten layer to reduce the dimension of the features extracted from the previous layer and the result is forwarded to the two dense layers which are followed by dropout to prevent overfitting. And finally, one dense layer to classify the 11 states as can be seen in Fig2.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv4 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv4 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv4 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
flatten_1 (Flatten)	(None, 8192)	0
dense_1 (Dense)	(None, 4096)	33558528
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
dense_3 (Dense)	(None, 11)	45067

Total params: 70,409,291
Trainable params: 50,384,907
Non-trainable params: 20,024,384

Fig2. Model Architecture II

Now the Model is compiled with optimizer RMSprop a with 0.001 learning rate to train on the augmented training dataset while being monitored with same callback to prevent overfitting and save model. In this model also, freezing the base model was same as the earlier model to fine-tune for our problem[4][19].

For the second iteration, the model is compiled with SGD using categorical cross-entropy as loss function along with momentum with 0.001 learning and allowed to train furthermore on the data while validation data is used for validating the model with callbacks. Epoch used for the second iteration is 50. And based on the checkpoint best model of validation accuracy is saved and used for the test the model performance using the separate test data. Since there isn't any big process or convolution layer batch normalization layer was not used.

IV. Result

Performance of the two models are assessed on the test data. Compared to the two model Model II had better accuracy than the earlier one. The model I had an accuracy of 50.18% while the Model II had an accuracy of 64.42% on the test dataset. The loss and accuracy of both models on train, validation and test are tabulated below in Table1.

Performance Metrics	***Model I***	***Model II***
Train Accuracy	58.67%	84.92%
Train Loss	1.3779	0.4779
Validation Accuracy	50.18%	65.40%
Validation Loss	1.6129	1.2277
Test Accuracy	49.98%	64.42%
Test Loss	1.5830	1.2060

Table1. Performance result

From the above table, we can see that Model II performed better than Model I in case of accuracy, i.e., classifying a better number of state of the cooking object. This is better visualized in the graph plots below in Fig3&4.

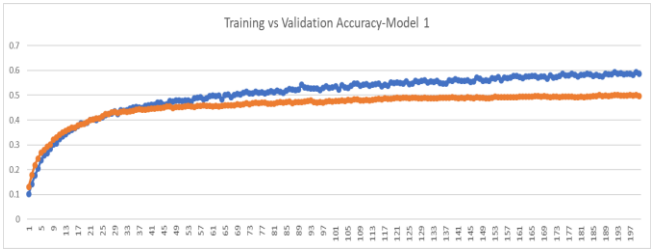

Fig3. Training vs Validation Accuracy Model I

And we can see the Model I have a better resistant to overfitting than that of the Model II, this might be the reason of usage of both dropout and the batch normalization and the kernel regularization of L2 we have used in that model.This difference can be clearly viewed in the figures Fig5&6 below.

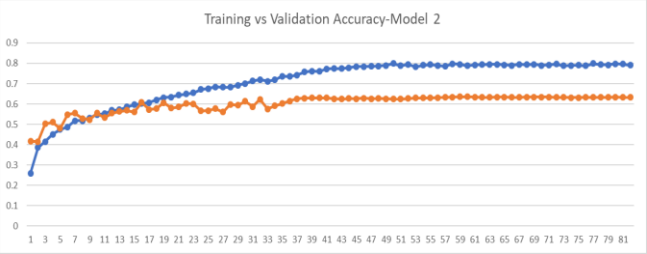

Fig4. Training vs Validation Accuracy Model II

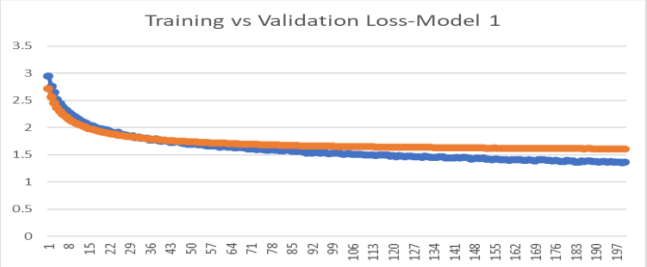

Fig5. Training vs Validation Loss Model I

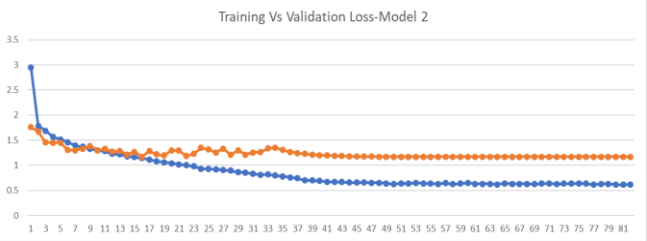

Fig6. Training vs Validation Loss Model II

V. Conclusion

In this paper, we have presented the result of two models for the cooking object's state classification build from the fine-tuned VGG19. We had an accuracy of 64.42% for the classification of the 11 states of the cooking objects annotated to a specific problem. On the process of building a model for cooking object state classification, we have understood how different parameter and techniques of the deep learning model affects a model performance in a different way. In future work, we can analyze more techniques of deep learning better with the larger dataset and can build a better for cooking object state classification.

VI. References

[1] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), pp.1929-1958

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

[3] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), pp.1929-1958.

[4] Keras Tutorial : Fine-tuning using pre-trained models FEBRUARY 6, 2018 BY VIKAS GUPTA

[5] Classifying Cooking Object's State using a Tuned VGG Convolutional Neural Network by Rahul Paul, arxiv.org/ftp/arxiv/papers/1805/1805.09391.pdf

[6] Transfer learning & The art of using Pre-trained Models in Deep Learning by DISHASHREE GUPTA, JUNE 1, 2017

[7] Multi class Fish Classification on Images using Transfer Learning and Keras by Tahsin Mayeesha

[8] Transfer learning from pre-trained models by Pedro Marcelino

[9] Image classification with a pre-trained deep neural network Publié le mardi 21 Juin 2016 dans Sémantique Données non-structurées, Machine Learning

[10] Batch normalization in Neural Networks by Firdaouss Doukkali

[11] State Classification with CNN Astha Sharma arxiv.org/ftp/arxiv/papers/1806/1806.03973.pdf

[12] Objects as Attributes for Scene Classification Li-Jia Li*, Hao Su*, Yongwhan Lim, Li Fei-Fei

[13] ImageNet: VGGNet, ResNet, Inception, and Xception with Keras by Adrian Rosebrock on March 20, 2017

[14] G. E. Dahl, T. N. Sainath and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, 2013, pp. 8609-8613.

[15] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," 2013 IEEE International Conference on Image Processing, Melbourne, VIC, 2013, pp. 4034-4038.

[16] Jin, Jonghoon & Dundar, Aysegul & Culurciello, Eugenio. (2014). Flattened Convolutional Neural Networks for Feedforward Acceleration

[17] Zhun Sun, Mete Ozay, Takayuki Okatani, {sun, mozay, okatani}@vision.is.tohoku.ac.jp, Design of Kernels in Convolutional Neural Networks for Image Classification, arXiv:1511.09231v3 [cs.CV] 29 Nov 2016.

[18] A bunch of tips and tricks for training deep neural networks by Long Ang

[19] A Comprehensive guide to Fine-tuning Deep Learning Models in Keras (Part II) OCTOBER 8, 2016 by Felix Yu

[20] Handling overfitting in deep learning models by Bert Carremans

[21] VGG19 Fine-tuning model September 3, 2017 marubon-ds.blogspot.com/2017/09/vgg-fine-tuning-model.html