# State Classification of Cooking Objects using pre-trained VGG-19 Network

***Abstract*— Object and state recognition are both parts of the image classification problem, in which the object recognition is the process of identifying an object in an image whereas the state recognition is the process of identifying the progress by recognizing the state of object in that particular image. Object and state recognition inside the kitchen is very important if the cooking is being carried out by a robot. In this article the approach which is used to classify the object among one of the total 11 states is transfer learning. A dataset with a large number of images was maintained which consisted of various objects which were divided into 11 different states. A new network has been created by adding few more layers to the pre trained network called VGG-19 which is used as the base model. This paper discusses about how to improve the accuracy for the state recognition by using transfer learning.**

## I. Introduction

The technology in the today's world has made a very impressive and unbelievable improvement. With the improvements, and the innovation in the technology the life of the humans is getting easier day by day. One such innovation which eased the human life is the introduction of the robots which assists people in their mundane works. Robots are well trained and are being used in the field of medicine, manufacturing, construction etc. [1][2][3] But not much research has been done to use robots for cooking.

As everyone is aware of the universal fact that food is the basic need of human, it is beneficial to train the robots in order to replace the humans in the kitchen. Cooking is a very complex process which has various stages and divisions. Cooking involves identifying the correct ingredients, cutting the objects, identifying various states of an object, mixing the ingredients and using the heat to prepare the final product. Teaching a robot how to cook, is equivalent to teaching a person who is unaware of the cooking, albeit it's even more complicated as compared to humans because humans have billions of neurons in their brain and making a similar neural network which can work exactly like a human brain is still something which requires a lot of improvement in the existing technology. To train a robot to perform cooking, first it needs to be trained to identify/detecting the objects [4], grasp the objects and other items like knife to cut these objects by analyzing angle and the technique accurately, and later it needs to identify the state of the object correctly to perform the rest of the actions. Various efforts have been done to teach robots to correctly identify and grasp the objects as grasping different objects and cutting those objects needs different strategies [5][6]. But identifying the state of the objects correctly is something which still needs to be investigated more deeply.

This paper discusses about the approach of identifying the state of the object while cooking by focusing on only a small part of the state recognition that is classification. We have used total of 11 classes like sliced, peeled, whole, creamy_paste, floured, julienne etc. Few of the sample images from the dataset have been displayed above in the Fig. 1. There are few states which are ambiguous like sliced and grated and it is even sometimes difficult for a person who is not much skilled in the cooking to spot the difference between those two. A deep neural network has been used which is VGG-19 [7] as the base model and transfer learning process is used to train the model. Few layers are added on top of the pre-trained network and various parameters like learning rate, optimizers, and regularizers have been changed and tested to increase the accuracy of the network.

Figure 1. Example of different states of cooking objects( from left to right: creamy_paste, floured, juiced, grated, juilenne, mixed, whole, peeled respectively)

## II. Data Collection And Preprocessing

### A. Data Collection and Annotation

The first step in preparation of the data was to annotate the objects in each frame of the provided video whenever the object changes its state. The data annotation is a process in which square boxes are drawn around the object, selecting its state whenever that object appears in the frame and it changes its state. The annotation on a bell pepper is shown in Fig. 2. After the data annotation, data was collected which consists of thousands of images of objects like onion, bell pepper, tomatoes, meat, bread etc. which were in any of the 11 possible states like whole, big-cut, sliced, creamy_paste etc. Secondly, this data was shuffled and randomly divided into three datasets, i.e. train, valid and test.

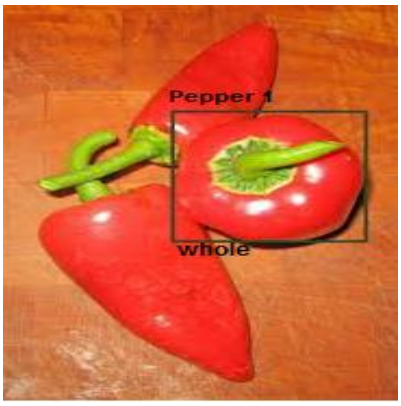



Figure 2. Data Annotation on a bell pepper

Figure 3. Example of data augmentation: original, rotated, zoomed-in and horizontally-flipped image

The train dataset consists of 6348 images and the valid dataset consists of 1377 images.

### B. *Data Augmentation and Preprocessing*

The data is the most crucial part of a deep neural network which is used for image classification. Providing enough data and correct data is very important to train the network correctly. Data preprocessing is the process of converting the raw data into a more refined and consistent form. Data preprocessing helps us to overcome the problem of inconsistent, insufficient and incomplete data [8].

There are various ways of data preprocessing like data normalization, data resizing, data augmentation etc. The one which is discussed here is the data augmentation. The dataset size which is mentioned above is not a very large and sufficient. Therefore, to increase the size of the dataset the data augmentation was done. Augmenting the data involves image re-scaling, zoom-in or zoom-out, flipping the image, rotating the image etc. The data augmentation increases the number of images by producing new images which prevents over-fitting. Fig. 3 shows the example of augmented data after rotation, zoom-in, horizontal image along with the original image.

The parameters which are set in the final network discussed in this paper are shown in the below table.

TABLE I. DATA AUGMENTATOIN PARAMETERS AND VALUES

| S. No. | ***Parameter Name*** | ***Parameter Value*** |
|---|---|---|
| 1. | shear_range | 0.2 |
| 2. | zoom_range | 0.2 |
| 3. | horizontal-flip | True |
| 4. | rotation_range | 30 |

## III. METHODOLOGY

This paper presents an enhanced model which is based on a pre-trained model named as VGG-19. It is a convolutional layer network which is a feed forward network. When the output of one layer is forwarded to the next layer as input, that network is known as feed-forward. We have followed the same approach and added few layers after the layers of the base model and the output of the base model is used as input to the self-defined layers. The transfer learning or knowledge transfer approach is used as it is very efficient framework when pre-trained network has been trained on another dataset and we have different dataset compared to the prior one [9] [10]. The following sections explain about the base model, proposed model and the training process.

### A. *Base Model*

The pre-trained model that has been used here is VGG-19. VGGNet has been created by the University of Oxford in 2014, which is the first runner-up in the ILSVRC-2014, but it is the best used model. It has two variation VGG-16 and VGG-19.

VGG-19 is a deep convolutional layer network that has 19 layers which consists of convolutional layers, Max-Pooling layers, fully connected layers and a last layer that uses softmax function for the classification [7]. Also, all the filters that have been used are of 3x3size. VGG-19 uses the data from the Imagenet [11]. The structure and the weights of this pre-trained network are made available online for free [7]. This model is selected because of its simplicity and depth. The architecture of the base model has been shown in Fig. 4 [12].

### B. *Proposed Model*

The model that has been proposed in this project has been created by adding five more layers on the top of layers of the base model. These five layers consist of 1 pooling layer, 1 batch-normalization layer and 3 fully-connected layers. Fig. 5 represents the architecture of the proposed model and Fig.6 represents the summary of the model.

- Pooling Layers- The first layer that is added is a GlobalAveragePooling2D layer. This average pooling provides a linear transformation. This pooling layer has been added to reduce the variance and computation complexity. The average pooling is chosen instead of max-pooling to retain more amounts of data and to preserve localization. The average pooling encourages the network to identify the complex content of the object [13].

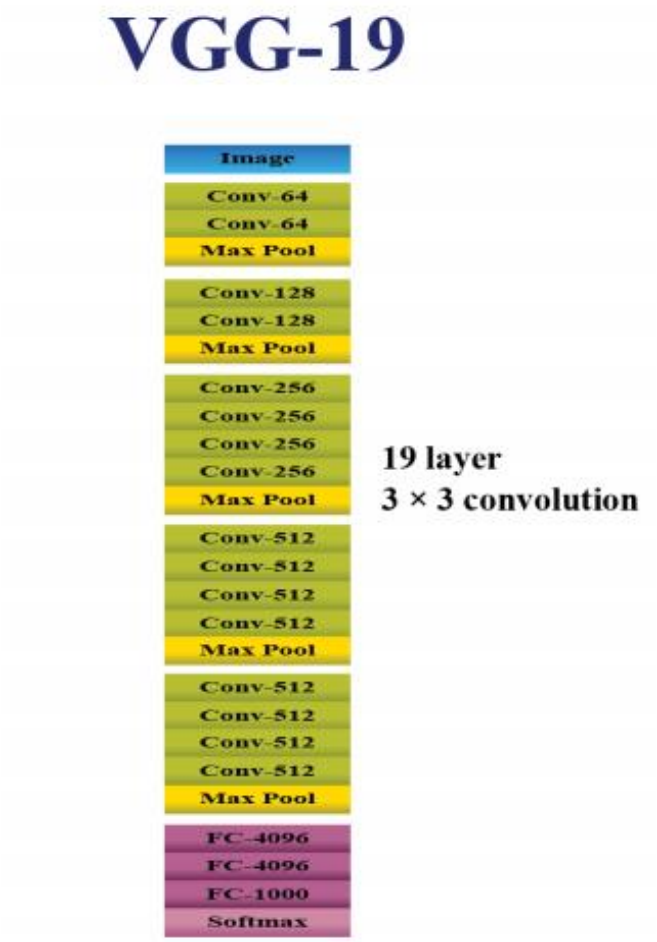



Figure 4. Architecture of Base Model VGG-19

- Batch Normalization Layer- The second layer that has been added is the batch-normalization layer which is added just after average pooling and before the dense layers to normalize the data. This layer normalizes the inputs in a way that they shift towards zero or mean of the inputs and their mean output activation becomes zero and standard deviation becomes one. This layer increases the performance and stability of the network and makes the learning faster.
- Dense Layers- The next three layers that are added are all fully connected layers with 2048, 1024 and 512 neurons respectively. These dense layers are known as fully connected layers because every input node is connected to every output node. All three layers have ReLU as the activation function. ReLU is used because it provides more enhanced performance and speed.
- Regularization- To prevent over-fitting two types of regularization has been used in this model, one is L2-regularization and the other is Dropout. Every dense layer has used dropout with the factor of 0.4, 0.3, and 0.2 respectively.

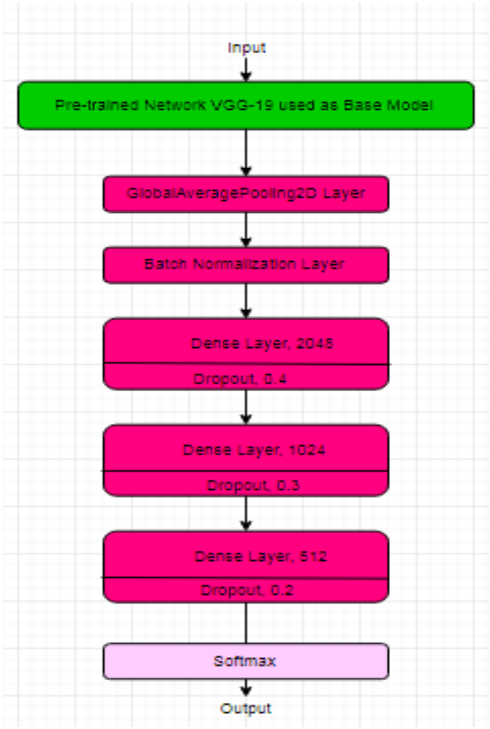



Figure 5. Architecture of Proposed Model built on VGG-19

Some randomly selected neurons are dropped-out [14]. L2 regularization prevents over fitting by adding square of the weights which is known as a regularization term to the coefficients which are capable to overfit.

- Output Layer- The last layer is the activation layer which is using the softmax function which generates the output between 0 and 1. The softmax is used as this state recognition is a classification problem.

### *C. Training*

The training is done by keeping the newly added and the top two layers of the best model were unfreeze and set as trainable and the rest of the layers of the best model are set to non-trainable, i.e. their weights will not get updated while training. The model is compiled using the Adam optimizer. The training was done using SGD and Adam optimizer and Adam was the choice as it produces the best results. Various experiments were done with the learning rate and finally, it was set to 0.0005. The decay of 0.0001 is given to the Adam optimizer that means that after the accuracy will reach to its optimal value, the learning rate will be decreased by a value which is equivalent to the decay. The image size was set to 256 x256. The best model was saved which was later used for training and also, the early stop technique was used to make the training fast and reduce over-fitting with a patience of 10. Early stop makes the model to stop learning if it has reached its maximum accuracy and after that point neither the accuracy is getting higher nor the loss is getting lower, it saves training time. The final accuracy which is achieved

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_1 (InputLayer) | (None, None, None, 3) | 0 |
| block1_conv1 (Conv2D) | (None, None, None, 64) | 1792 |
| block1_conv2 (Conv2D) | (None, None, None, 64) | 36928 |
| block1_pool (MaxPooling2D) | (None, None, None, 64) | 0 |
| block2_conv1 (Conv2D) | (None, None, None, 128) | 73856 |
| block2_conv2 (Conv2D) | (None, None, None, 128) | 147584 |
| block2_pool (MaxPooling2D) | (None, None, None, 128) | 0 |
| block3_conv1 (Conv2D) | (None, None, None, 256) | 295168 |
| block3_conv2 (Conv2D) | (None, None, None, 256) | 590080 |
| block3_conv3 (Conv2D) | (None, None, None, 256) | 590080 |
| block3_conv4 (Conv2D) | (None, None, None, 256) | 590080 |
| block3_pool (MaxPooling2D) | (None, None, None, 256) | 0 |
| block4_conv1 (Conv2D) | (None, None, None, 512) | 1180160 |
| block4_conv2 (Conv2D) | (None, None, None, 512) | 2359808 |
| block4_conv3 (Conv2D) | (None, None, None, 512) | 2359808 |
| block4_conv4 (Conv2D) | (None, None, None, 512) | 2359808 |
| block4_pool (MaxPooling2D) | (None, None, None, 512) | 0 |
| block5_conv1 (Conv2D) | (None, None, None, 512) | 2359808 |
| block5_conv2 (Conv2D) | (None, None, None, 512) | 2359808 |
| block5_conv3 (Conv2D) | (None, None, None, 512) | 2359808 |
| block5_conv4 (Conv2D) | (None, None, None, 512) | 2359808 |
| block5_pool (MaxPooling2D) | (None, None, None, 512) | 0 |
| global_average_pooling2d_1 ( | (None, 512) | 0 |
| batch_normalization_1 (Batch | (None, 512) | 2048 |
| dense_1 (Dense) | (None, 2048) | 1050624 |
| dropout_1 (Dropout) | (None, 2048) | 0 |
| dense_2 (Dense) | (None, 1024) | 2098176 |
| dropout_2 (Dropout) | (None, 1024) | 0 |
| dense_3 (Dense) | (None, 512) | 524800 |
| dropout_3 (Dropout) | (None, 512) | 0 |
| dense_4 (Dense) | (None, 11) | 5643 |

Total params: 23,705,675
Trainable params: 3,680,267
Non-trainable params: 20,025,408

Figure 6. Summary of Proposed Model built-on VGG-19

is 69.67% on the valid dataset and on the unseen test dataset the accuracy is 63.5%.

## IV. Experiments And Results

Numerous experiments were done by using myriad of comparisons of various parameters to achieve a higher accuracy and to train the model correctly. This section shows comparison between two models to show by manipulating

TABLE II. Comparison of Accuracies And Loss Of two models

| ***Model No.*** | ***Accuracy*** | | ***Loss*** | | ***Test Accuracy*** |
|---|---|---|---|---|---|
| | ***Training*** | ***Validation*** | ***Training*** | ***Validation*** | |
| 1 | 71.59% | 56.20% | 1.16 | 1.31 | 53.8% |
| 2 | 75.22% | 69.67% | 0.82 | 1.12 | 63.5% |

What all factors, the accuracy got increased and the over-fitting got reduced. The accuracies and loss of those models can be seen in Table II.

First model (Model 1) was submitted for the round 1 and the number of epochs was set to 30, model was compiled twice using two different learning rates i.e. 0.005 and 0.0001 respectively. All the layers of the base model were kept frozen and three new dense layers were added on top of base model with 70, 50 and 30 neurons in each layer with dropout regularizers of factor 0.3, 0.2 and 0.2 respectively. Batch normalization was also added to make the training fast. The test accuracy which was achieved by this model was 53.6%

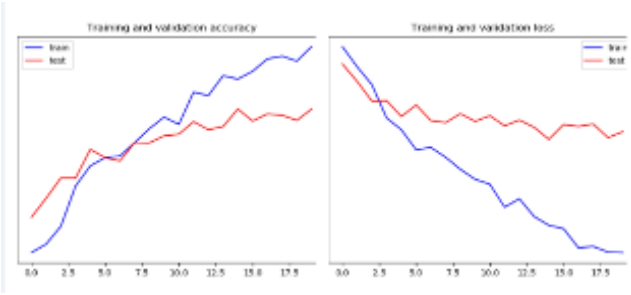
The second improved model (Model 2) which was submitted for round 2 was trained for 150 epochs and early stop technique was used to stop the training of the network once it reaches its optimal value and no change appears after a patience of 10. In this model two top layers of the base model were unfreeze and trained with the newly added layers while the model is compiled only once with a lower learning rate of 0.0005 and a decay of 0.0001. The number of neurons in each layer was increased by a large factor as compared to the above model. The three fully connected layers have 2048, 1024 and 512 neurons with the dropout factors of 0.4, 0.3 and 0.2 respectively. Also, L2 regularization was added to the dense layer with 2048 neurons. The test accuracy was increased by certain amount and it reached 63.5%.

Now, by closely looking at the above models, it can be easily noticed that by increasing the number of epochs with early stop, raising the number of neurons in dense layers, unfreezing the base model layers, setting the learning rate in the correct region, adding batch normalization and regularization helps the model to train faster and better, in addition it increases the accuracy. Fig. 7 represents the graphs of the accuracy and loss of both the models discussed.

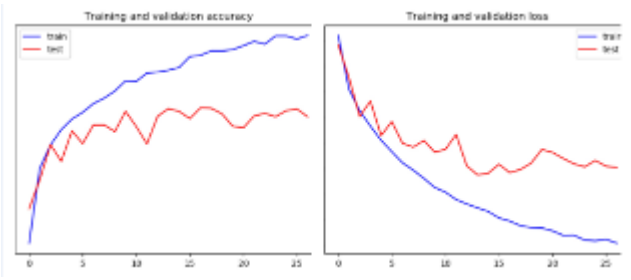
## V. Discussion

This paper discussed about a solution to the problem of state recognition in the cooking process that can be used to train the robots to cook. A model was designed by using the transfer learning approach by using VGG-19 as the base model and an accuracy of 63.5% was achieved. Few things which we learnt were that the size of the dataset plays a vital role in training and since we had a smaller dataset, in future

to




a)




b)

Figure 7. Training and Validation Accuracy and Loss of the two models: a) Model 1 and b) Model 2

to improvise the training, training should be done by using a larger dataset. Also, unfreezing more number of layers adds up to the higher accuracy. Training the model to a higher number of epochs also helps in better learning. It was also noticed that there is ambiguous data present and it is difficult to classify the object's state. For instance, the grated and creamy_paste seems to be confusing at times [15]. Some efforts can be done to solve this issue.

## REFERENCES


[1] International Federation of Robotics Frankfurt, Germany, 2018, March. "Robots and the workplace of the future".

[2] Frédéric Kaplan, 2004, October. "Everyday Robotics: robots as everyday objects", Joint sOc-EUSAI conference.

[3] "Robotics: Facts", Url: http://idahoptv.org/sciencetrek/topics/robots/facts.cfm

[4] Christian Szegedy, Alexander Toshey, Dumitru Erhan, 2013. Deep Neural Networks for Object Detection", Advances in Neural Information Processing Systemss 26 (NIPS 2013).

[5] Pavel Dzitac, Abdul Md Mazid, 2013, Nov. "Robotic Object Grasping in context of human grasping and manipulation", IEEE Conference on Robotics, Automation and Mechatronics (RAM).

[6] Changhyun Choi, Wilko Schwarting, Joseph Delpreto, Daniela Rus, 2018, July. "Learning Object Grasping for Soft Robot Hands", IEEE Robotics and Automation Letters.

[7] Karen Simonyan, Andrew Zisserman, 2015, April. "Very Deep Convolutional Networks for Large Scale Image Recognition", ICLR-2015.

[8] Rehab Duwari, Mahmoud El-Orfali, 2014 May. "A study of the effects of preprocessing strategies on sentiment analysis for Arabic text", Journal of Information Science, Volume: 40 issue:4, page(s): 501-513.

[9] Sinno Jialin Pan, Qiang Yang Fellow, "A survey on Transfer Learning", IEEE Transactions Knowledge and Data Engineering.

[10] Yusuf Aytar, 2014. "Transfer Learning for Object Category Detection", Robotics Research Group Department of Engineering Science, University of Oxford.

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei. "Imagenet: A Large-Scale Hierarchical Image Database", Department of Compute Science, Princeton University, USA.

[12] Uttam Mojumdar, Taqi Tahamid Sarker, Gulnahar Mahbub Monika, Nurul Amin Ratul, 2016, Dec. "Vehicle Model Identifiction using Neural Network Approaches", Department of Computer Science and Engineering, BRAC University.

[13] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba. "Learning Deep Features for Discrimainative Localization", Computer Science and Artificial Intelligence Laboratory, MIT.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, 2014-June. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Journal of Machine Learning Research 15 (2014) 1929-1958.

[15] Ahmad Babaeian Jelodar, Md Sirajus Salekin, Yu Sun, 2018-Oct. "Identifying Object States in Cooking Related Images".

[16] Astha Sharma, "State Classification with CNN", University of South Florida. URL: https://arxiv.org/ftp/arxiv/papers/1806/1806.03973.pdf

[17] Md Sirajus Salekin, Ahmad Babaeian Jelodar, Rafsanjay Kushol, 2019-Jan. "Cooking State Recognition from Images Using Inception Architecture". URL: https://arxiv.org/pdf/1805.09967.pdf

[18] Rahul Paul, "Classifying Cooking Object's State using a Tuned VGG Convolutional Neural Network". URL: https://arxiv.org/ftp/arxiv/papers/1805/1805.09391.pdf

[19] Tianze Chen, 2018. "Identifying States of Cooking Objects Using VGG Network". URL: http://rpal.cse.usf.edu/reports/course_reports/2018_DL_Chen1.pdf