Convolution Neural Network for Cooking State Recognition using VGG19

Hemanth Potlabathini

***Abstract*— In this paper we are aiming at solving the state recognition challenge that we face while trying to teach a robot manipulation tasks, so that it could do it by itself all the physical work required for the task to be completed. For example, a robot assigned with a cooking task must be able to identify the objects it needs for cooking a recipe. So, recognizing the state in which the ingredients are is one of the most important things, the robot must be capable of doing this. We have used convolutional neural networks for achieving this. There is a total of 11 states in which most of the objects would be. So, there will be 11 classes and given an image, the convolutional neural network must be able to classify the object in the image to its suitable class. We have used a VGG19 architecture for this problem and have made modified the VGG19 model by trying different layers and have experimented with different types of optimizers and have performed data augmentation, regularization to improve the model.**

Keywords – Convolutional Neural Networks, VGG19, state recognition, object, grasping.

I. INTRODUCTION

As we all know, robots have been getting into every possible industry now a days. They would make our life much more convenient. Think about a robot which could cook food for us, that would save a lot of hassle. A cooking robot, if perfected can be very effective and very useful. But it certainly lacks the intelligence human beings have. For elderly people and people who are very busy and don't find time for cooking, these robots could be of great help to them. But getting the robots to do things perfectly as humans is very tough thing to achieve. So, in this paper we discuss about one of the most important things, a cooking robot should be capable of doing, and that's state recognition. The robots before cooking must be able to identify the state of the objects [4] [5], it would use for cooking. An object or ingredient may be in different states namely, diced, julienne, whole etc. The way of cooking an object is different based on the state it is in currently, so identifying the state is one of the problems, these cooking robots will face. Over the past couple of years deep learning has made a lot of improvement and these deep learning approaches are now a days getting good results for solving such problems.

Robotic grasping is necessary for cooking robot as it needs to hold the object for cooking. A grasp is described as the factor at which the robot can hold the object and lift it, without slipping it. And these robots require large amount of data for learning. We will be using images with objects belonging to different classes for training the network. But for the cooking robot to know the sequence of states a object goes through while cooking, using cooking videos for training would be a much better way to train the robot. In [2] the author has used more than 200 cooking videos were used for training the robot. And Functional Object-Oriented Network was used for developing the cooking robot. Lot of manipulation techniques and grasping techniques need to be taught to the robot. But this paper only deals with the state recognition task, which is an important thing in developing a cooking robot. In [3] Resnet based deep model is proposed for solving the sate recognition challenge.

II. METHODOLOGY

A. State challenge

For a cooking robot as we discussed earlier object manipulation is the important thing and the state of the objects changes frequently throughout the cooking process. Suppose you have an onion, then initially it will be in the whole state, and then based on the recipe we are cooking the desired state of onion may change, we may chop the onion based on requirements. So, the state changes from whole to chopped. Similarly, the ingredients may be placed separately or mixed in same bowl, then the objects are initially in unmixed state and as soon as they put in same bowl and mixed, then their state changes from unmixed to mixed. So, the challenge for the robot would be to identify the current state of the object and then decide the correct procedure to cook it in further steps. This is generally referred to as Sate challenge.

B. Convolutional Neural Network

As convolution neural networks have made a huge progress in image and video recognition over the past couple of years, we have decided to employ convolutional neural networks for tackling the state challenge. The convolutional neural networks are like regular neural networks with learnable weights and bias but the main thing with convolutional neural networks is that it assumes that the inputs it is going to receive are images and it has certain properties encode with it, this makes things easy as parameters required will drastically reduce. All the operations we could perform on a regular neural network can also be performed on the convolutional neural networks.

C. Dataset

Convolutional neural networks need a large amount of data for training. The dataset [1] we have used consists of 11 different classes. Each class corresponds to a state. There is a total of 11 classes representing 11 states. The states are sliced, julienne, floured, grated, whole, creamy paste, diced, juiced, peeled and other is the last state. If the image cannot be classified correctly to any of the 10 states, then it is states as other state. There are two directories in the dataset namely train and valid. The test folder consists of 6348 images, where as the valid directory consists of 1377 images. Fig 2.1 represents the dataset and it shows the ratio of train and test images of each class.

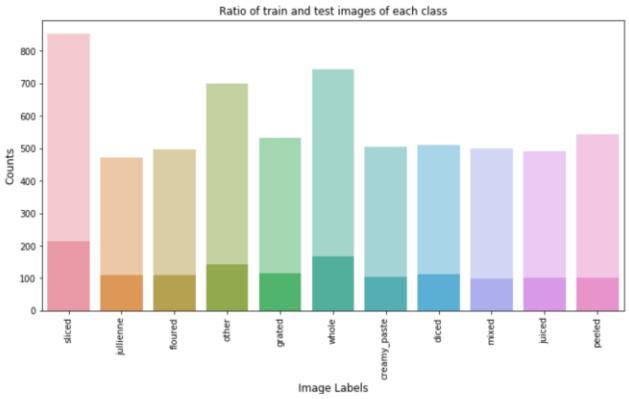

Fig 2.1 Ratio of Train and Test images in each class

The fig 2.2 shows the sample images of creamy paste and diced classes in the dataset.

Fig 2.2 Dataset Sample

D. Model

We have used VGG19 [6] architecture pre-trained on ImageNet has been used for our convolutional neural network. It is necessary to do so because, it contains millions of parameters. The training is done on the train folder in the dataset and the valid folder is used to constantly keep track whether it is learning correctly or not. We have added two fully connected layers on to the top of the base model. One layer consists of 80 neurons and the other consists of 50 neurons. Fully connected neurons are those in which each neuron is connected to every other neuron in its next layer. The last layer consists of 11 classes each representing a state. The Fig 2.3 shows the structure of the model.

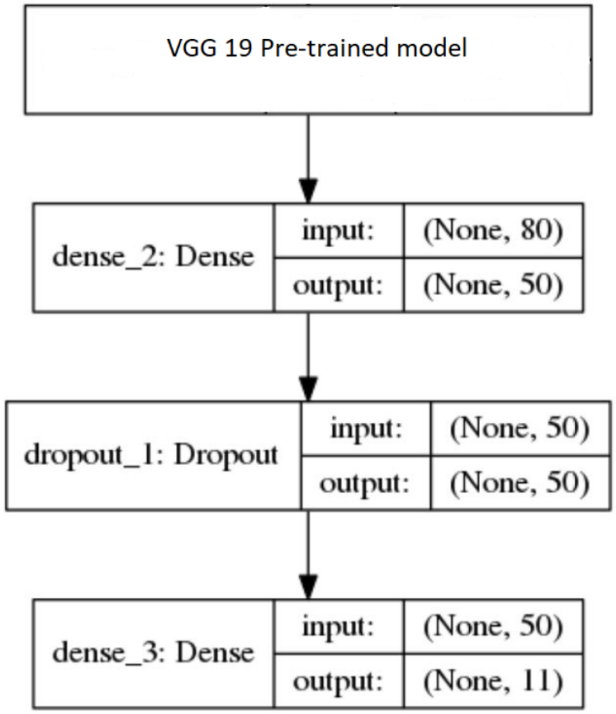

Fig 2.3 Model structure

In the first stage we will freeze the layers of the pre-trained network and train the layers we have added and in the second stage we will unfreeze some of the layers of the pre-trained network and train them along with the fully connected layers added at the end. The data we from the dataset we have provided is still not enough for the network. So here we can try Data Augmentation so that the number of images can be increased by a significant number. This is also preprocessing for the network.

The ImageDataGenerator[7] class from the Image Preprocessing is used for achieving data augmentation. The ImageDataGenerator function consists of parameters such as rotation_range, width_shift_range, height_shift_range and Zoom_range etc. These parameters will change the images currently present in the dataset and generate a new image from it. It doesn't entirely generate a new picture, but it will change the dimensions of the image so that it will be considered as a new image by the network. For example, flipping an image horizontally or vertically will create a new image. Fig 2.4 shows the sample pictures generated using train_generator.

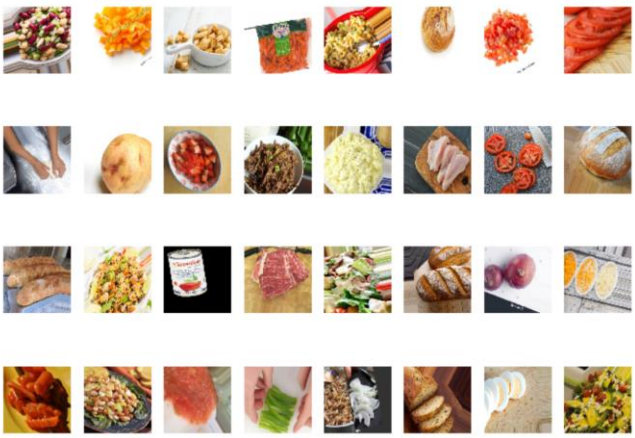

Fig 2.4 Sample images created by train_generator

We have used relu activation function for each of the added layer. Softmax function is used for the last layer. Activation function will convert its inputs to the form of probabilities. The image will belong to the class with the highest probability. We can use optimizers for calculating the learning rate for parameters individually and update the weights of the parameters iteratively. We can change learning rates and number of epoch and see if the accuracy is improving or not. In the train_generator and test_generator functions we define the batch size, which would decide the number of images to be sent for training for each epoch. And then we will save the model, to use it for testing on different dataset. So, these are the general things we do for building a convolutional neural network.

We have done regularization after the second fully connected layer. Dropout of 20 percent is used here for regularization. Dropout will randomly ignore some of the neurons and their weights won't get updated, this will make the left-over neurons to take over the work that needs to be done by the neurons removed, so the network begins to learn more efficiently as a result of dropout. This will also decrease the overfitting of the network.

The optimizer used here was Adam with lr of 0.001 and decay of 0.0. Adam was used at the both stages of training here. Initially Rmsprop and SGD were also used for training, but the accuracy is much better when Adam optimizer is used. The learning rate of 0.01 is used for the first stage of training and learning rate of 0.001 is used for the second stage of training. If the initial learning rate is very large, then the accuracy has dropped very low. Learning rate of 0.001 has yielded much better results for the first stage. Then the second stage learning rate must be smaller compared to the first stage. For the second stage unfreezing the layers of VGG19 has most of the times resulted in decrease in the accuracy. So, the second stage also trains the network from 20^{th} layer. Increasing the number of layers also decreased the accuracy of the network. Doing Batch Normalization helped in increasing the accuracy of the network. Batch normalization will make the large data into mini batches, this will help the network train much faster as it normalizes the data and makes things easy for initializing.

III. Experiments and Results

Many parameters need to be considered for training the network and each parameter may have significant impact on the accuracy of the network. The number of layers were frequently changed from 2 to 3 and tested. The learning rate for first stage is generally 0.01 or 0.001 and based on the first stage learning rate, second stage learning rate is determined, and it must be less than the first stage. Different optimizers such as Adam, RMSprop and SGD were tested. Apart from these the number of layers to be trained in the second stage of training is also varied. Epochs for each stage also have been varied and tested. The best accuracy for the test data is 58%.

TABLE I: Classification of validation accuracy and Accuracy at stage 2 for different changes in the parameters

Layers	**Epochs**		**Optimizer**	**Learning rate**		**Val_acc**	**Acc stg 2**
	I	***II***		***LR1***	***LR2***		
2	12	12	RMSprop	0.01	0.001	63	57
2	20	20	RMSprop	0.01	0.001	64	55
2	25	25	Adam	0.01	0.001	15	15
2	20	20	SGD	0.001	0.0001	66	54
2	25	25	SGD	0.01	0.0001	60	51
3	35	35	Adam	0.001	0.0001	66	54
3	30	30	Adam	0.01	0.0001	69	54
2	30	30	Adam	0.001	0.0001	65	57
2	25	25	Adam	0.001	0.0001	68	58

TABLE II: Classification of loss for different changes in the parameters

Layers	**Epochs**		**Optimizer**	**Learning rate**		**Val_loss**
	I	***II***		***LR1***	***LR2***	
2	12	12	RMSprop	0.01	0.001	12
2	20	20	RMSprop	0.01	0.001	13
2	25	25	Adam	0.01	0.001	23
2	20	20	SGD	0.001	0.0001	14
2	25	25	SGD	0.01	0.0001	17
3	35	35	Adam	0.001	0.0001	16
3	30	30	Adam	0.01	0.0001	14
2	30	30	Adam	0.001	0.0001	12
2	25	25	Adam	0.001	0.0001	12

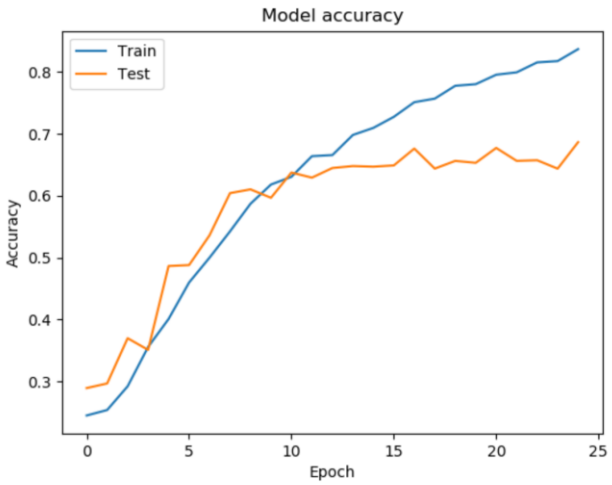

Fig 3.1 Training Accuracy vs Test Accuracy

IV. Conclusion

In this paper we have discussed a solution for state recognition using convolutional neural network using VGG19 architecture pre-trained on ImageNet weights. The best accuracy of the model is 58 percent on test data. The accuracy of this model needs to be improved. If a object is in two different states in a same picture, then it may be classified wrongly, such things need to be dealt with, in the future. And images may be wrongly classified for some of the similar states, if you consider melted and juice states, then network may classify melted objects as juice and juice objects as melted. Such ambiguity must be avoided. Trying to recognize the states of the objects in a video would be a great thing to do in the future. With day by day improvements in the deep learning field, we can surely build a network which can do the task more precisely in the future.

References

[1] http://rpal.cse.usf.edu/datasets_cooking_state_recognition.html.
[2] Sun, Yu. "AI Meets Physical World--Exploring Robot Cooking." *arXiv preprint arXiv:1804.07974 (2018)*
[3] https://arxiv.org/abs/1805.06956
[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In CVPR, pages 770–778. IEEE Computer Society, 2016.
[5] Ahmad Babeian, David A. Paulius, and Yu Sun. 2018. Long Activity Video Understanding using Functional Object Oriented Network. IEEE Transactions on Multimedia (2018),
[6] K. Simonyan and A. Zisserman. Very deep convolutional networks For large-scale image recognition. CoRR, abs/1409.1556, 2014.
[7] https://keras.io/preprocessing/image/