[[1]](#footnote-1)

Classification of cooking object states using CNNs

Sachit Anand Bhootham

*Abstract*— In order to build a robot which can replace humans in terms of cooking, it should have a proper understanding about the environment of cooking in order to execute any cooking tasks. There are several objects involved in cooking activities and each of these objects have different states. It is necessary that a kitchen robot understands all the object states in order to make cooking like the cooking performed by humans. There are several papers published where object recognition has been studied and models have been implemented but there is very minimal research performed on the object state recognition. In this paper we build a model based on deep learning methodology to differentiate the cooking or object states from several images so that our robot can identify various cooking states involved in cooking. In the developed model we have used various layers, activation functions and optimizers on which our model is analyzed. The validations results produced by our model show that the solution for cooking or object state recognition problems has a great potential.

# INTRODUCTION

Robots have automated several tasks which humans manually perform daily. Several industries such as automobile manufacturing, food packaging, construction etc, have deployed robots which play a crucial role in increasing the production, minimize human dependency and maximize cost cutting. The areas where robots are currently being used are very structured and hence, they have become very convenient to be used for the tasks which are repetitive. Currently scientists are focusing more on how robots can be used in our day-to-day tasks. Kitchen robots have become one of the major research topics where humans are replaced by robots for cooking. In order to create a robot which can cook, it must understand the cooking environment and various cooking states. Using previous research which was performed for object identification can be used to identify various objects involved in cooking, but it is important to have object state identification (for example mashed, whole, sliced, juiced etc.) for the robot to better understand cooking. For example, let us consider an egg, there are a lot of states involved. An egg can be a yolk, it can be boiled, mashed, sliced, diced etc.

In order to cook something, it involves various instructions which need to be performed one after the other. And different recipes have different instructions. For example, to make an omelet, first a robot needs to take a whole egg, crack it and pour the yolk into a bowl. Later take onions, other vegetables necessary and dice them and mix it along with yolk. Finally, the mixture needs to be poured into a hot pan. The robots here need to grasp the object by the instruction along with a specified motion to perform the task. There are various grasping strategies for robots [1], [2] already proposed. In [1], creators proposed a planning for task-oriented grasping from the very significant portion of the statistical nonparametric distribution model. In [2], [3], the creators proposed a grasp planning strategy based on the real time human demonstration such as thumb direction, placement and grasp type. In [4], the creator proposed a Robotic grasping technique for the instrument manipulations which involves physical interactions which are both voluntary and in-voluntary. In [5], the author proposed strategies on robots grasping from motion models under uncertainties. And similarly, there are several other techniques and strategies being researched in the robotic kitchen space.

There are various object states such as cut, sliced, mashed, mixed, diced, juiced etc. It is important for the robot to identify all these states in order to make cooking efficient. Hence our main goal here is to build a model which takes in a large set of different object state images and classifies them accordingly. There are various Convolutional Neural Network (CNN) models implemented for object classification and detection and have performed well. In this paper a Convolutional Neural Network has been implemented using preprocessed image test data. A total of 7213 images have been considered as the test set and a total of 1543 images have been considered as the validation set. After building the model using ‘Relu’ activation function, Batch Normalization, MaxPooling, Dropout and using ‘Adam’ optimized a training accuracy of 70.5% has been obtained and validation accuracy of 62.4% on 1543 test images has been obtained.

# Data and Preprocessing

For data and preprocessing [10], I have used image data generator, where the following parameters were set. The height shift range is set to 40 percent, validation split is set to 20 percent, horizontal flip is set to true, shear range is set to 40 percent, zoom range is set to 20 percent, width shift range is set to 30 percent, vertical flip is set to 20 percent, rescale is set to 1 over 255. Now the entire dataset is divided into train and test sets. For training set we set the target size to 128, 128, batch size to 32 and class mode is set to categorical. The batch size on the training set affects the accuracy and several batch sizes have been tested, out of which for the current model batch size 32 is suitable. A total of 7213 images belonging to 11 classes have been used for training. For the test set the target size is set to 128,128. The batch size is set to 32 and the class mode is set to categorical. The batch size of the test set does not affect the validation accuracy. A total of 1543 images belonging to 11 classes have been used for testing.

# Methodology

In our model we have added a total of four convolutional layers each of size 128, 128, 256 and 256, respectively. We have used L2 kernel regularizer for penalizing weights. We have added Batch Normalization [6], Dropout [7], MaxPooling2D [8] etc. Batch Normalization is a deep learning technique used for training, which standardizes the layer inputs for each mini batch. Dropouts are used to drop a percentage of weights to generalize or regularize the learning process and avoid overfitting [14]. In our case we have used a single dropout at the end of all the years which drops 10 percent of weights for each cycle. These 10 percent weights are randomly chosen and dropped. We have added MaxPooling2D to reduce the total number of pixels in an output image of the previous layer and thereby reducing the dimensionality. We have tried various activation functions for our model, for example we used ‘relu’, ‘elu’, ‘softmax’, ‘tanH’ etc. An activation function [9], is used to get the output of the nodes. It maps the outputs of the node values in between -1 to 1 or 0 to 1 etc. It purely depends on the function we use. In our case we have used ‘relu’ activation and ‘softmax’ to predict the classes.

For optimizing our model, we have tried various optimizers such as Adam, RMSprop, SGD etc. An optimizer is an algorithm which is used to alter the attributes of a CNN such as learning rate and weights in order to minimize the losses. In our case Adam optimizer has outperformed to other optimizers when it comes to the validation accuracy and minimizing the losses. Finally, a checkpoint has been set to save the model with best validation accuracy

# Evaluation and Results

The model shown in figure 1 has been implemented using python programming, Tensorflow [11] and Keras [12]. The dataset used by the model has a total of 11 labels such as creamy paste (595), diced (698), floured (511), grated (605), juiced (723), julienne (499), mixed (503), other (807), peeled (558), sliced (948), whole (766) etc. A total of 7213 images which are used for training the model and a total of 1543 images are used for validating the model. Initially an epoch size of 50 was considered and later the size of increased to 60, 75, 90 and 120. Increasing the number of epochs may result in an overfit. In order to analyze our model, after initial few epochs have run the loss value was observed. If the loss value is gradually decreasing the epochs were continued to run until the end. Or else the execution was stopped and after making necessary changes the model was re-executed. This process was continued until I got maximum the accuracy.

**Analyzing Optimizers**: To optimize the model a total of three different optimizers were used, they are Adam, RMSprop and SGD. The below diagrams show us the validation and training accuracy comparisons and training

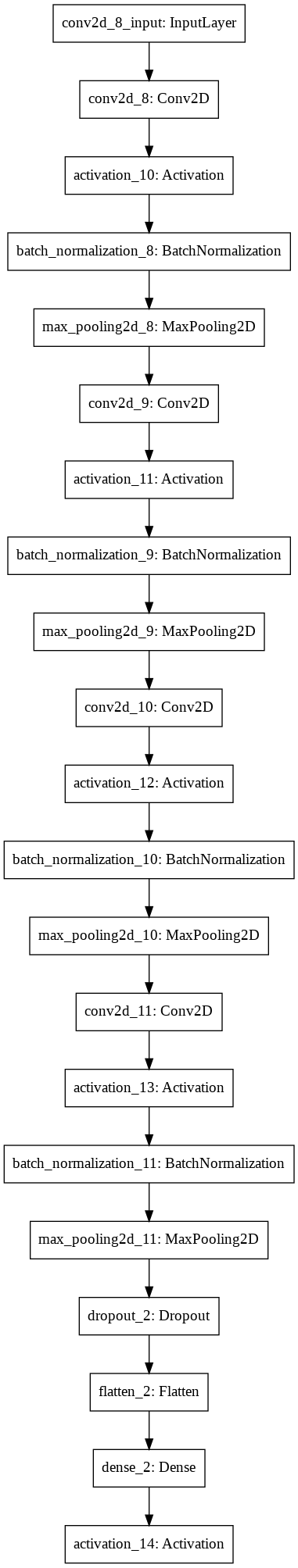


Fig. 1. Model Architecture

loss and validation loss comparisons between the above-mentioned optimizers.

Adam Optimizer with batch size 32.



Fig. 2. Adam optimizer Validation vs training accuracy

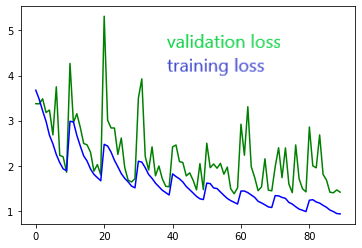


Fig. 3. Adam optimizer Validation vs training loss

RMSprop Optimizer with a batch size 32.



Fig. 4. RMSprop optimizer Validation vs training accuracy

In the above images X axis denoted the number of epochs and the Y axis denotes the loss and accuracy percentage.

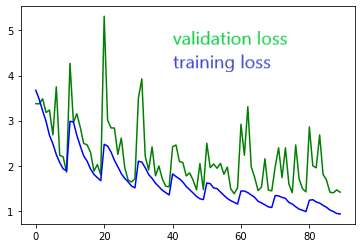


Fig 5. RMSprop Optimizer validation vs training loss

SGD Optimizer with a batch size 32.



Fig. 6. SGD Optimizer Validation vs Training accuracy



Fig. 7. SGD Optimizer Validation vs Training loss

The following results show us Adam and RMSprop optimizers have outperformed when compared with the SGD optimizer for the batch size of 32. When we compare Adam and RMSprop optimizers, Adam optimizer has recorded slightly better accuracy when compared with the RMSprop optimizer

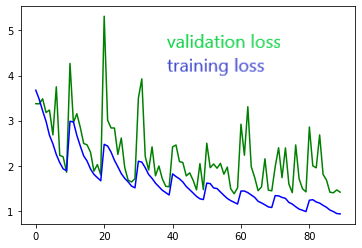
**Analyzing the batch size:** While training the model the following batch sizes 16, 32 and 64 were considered. Out of the three batch sizes, batch size of 32 has outperformed when it comes to validation accuracy and loss percentages. The plotted graph for Adam optimizer [14], with batch size 32 are shown below.

**Validation vs Training Accuracy**



(a). Adam Optimizer with batch size 32

**Validation vs Training Loss**



(b). Adam Optimizer with batch size 32

# Discussion

For the model which is built we were able to show that 62.4% of accuracy was obtained on the validation set. By using 4 convolutional layers having batch normalization, relu activation function, Max pooling 2D, dropouts and softmax activation function to predict the class our model was built. Later using Adam optimizer, we have improved our validation accuracy and minimized our validation loss when compared to other optimizers. Our results shows that this aspect of object state classification have a more scope to research and using advance deep learning techniques the accuracy can be optimized. For developing a kitchen robot cooking state identification has been a great challenge because cooking involves generic tasks which keep changing and the environment of cooking involves various states and stages. If these states can be identified by a robot the cooking process can be easy and accurate. For the future research, many unexplored aspects can be considered, and more accurate models can be developed.

References

1. Y. Lin and Y. Sun, “Robot grasp planning based on demonstrated grasp strategies,” The International Journal of Robotics Research, vol. 34, no. 1, pp. 26–42, 2015.W.-K. Chen, *Linear Networks and Systems* (Book style)*.* Belmont, CA: Wadsworth, 1993, pp. 123–135.
2. 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.
3. <https://www.semanticscholar.org/paper/Grasp-planning-based-on-strategy-extracted-from-Lin-Sun/689202b000543384b72d33cf1cbd9a220c2e5774>
4. <https://www.semanticscholar.org/paper/Robotic-grasping-for-instrument-manipulations-Sun-Lin/7ca6025657882364367ef773073c7a1840e70c0b>.
5. <https://www.groundai.com/project/learning-hybrid-object-kinematics-for-efficient-hierarchical-planning-under-uncertainty2354/2>.
6. <https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/#:~:text=Batch%20normalization%20is%20a%20technique,required%20to%20train%20deep%20networks>.
7. <https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5>.
8. <https://deeplizard.com/learn/video/ZjM_XQa5s6s>.
9. <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>.
10. <https://towardsdatascience.com/image-pre-processing-c1aec0be3edf>
11. <https://missinglink.ai/guides/tensorflow/tensorflow-image-recognition-object-detection-api-two-quick-tutorials/>
12. <https://towardsdatascience.com/image-detection-from-scratch-in-keras-f314872006c9>.
13. Classifying Cooking Object’s State using a Tuned VGG Convolutional Neural Network, Rahul Paul, Department of Computer Science and Engineering, University of South Florida, Tampa, Florida, USA
14. State Classification of Cooking Objects Using a VGG CNN, Kyle Mott

1. \*Research supported by Computer Science Engineering department of University of South Florida.

   Sachit Anand B, Author is with the Computer Science Department, University of South Florida, Fowler Ave, Tampa, FL 33613 USA. Deep Learning Project 1 (e-mail: sachitanand@usf.edu). [↑](#footnote-ref-1)