Classification of object states using VGG Neural Network

Oluwaseun Otunuga

Computer Science and Engineering Department
University of South Florida
Tampa, Florida 33620

April 11, 2019

Abstract

In this paper, we talked about dataset of images that consists of eleven cooking object states. With this dataset, we recognized object by classifying and knowing the state of the object. This was done by designing a convolution neural network. The neural network was trained using vgg-19 and we obtained our accuracy to be 70.3percent.

KEY WORDS: Convolutional Layers, VGG, dropout, Optimizers, Learning Rate.

1 Introduction

Our aim in this project work is to develop a model that will recognize and classify the state of the given object with good accuracy.

Convolutional Neural Network (CNN) is one of the category of Neural Networks that is effective in areas such as image recognition and classification. It (CNN) identify faces, objects and traffic sign. It also power vision in robots, Ujjwalkarn, 2016 [13]. According to R. Arbiol and V. Pala [2], there has been an evolution in the methods of image classification such as, Support Vector Machine, Artificial Immune System, Deep Learning, etc. New classification algorithms like Artificial Immune System (AIS) present innovative approaches to the unsupervised classification of remote sensing images (Zhong 2006) [14]. The aim of artificial Immune System is to make robots imitate human being to carry out some specific task e.g cooking, Tianze Chen [5]. On the other hand, deep learning is a subfield of machine learning that is based on learning data representations, as opposed to task-specific algorithms. For a robot to learn how to cook, the robot must know how to recognize the object, state the object, Y. Lin and Y. Sun [10], and also be able to grasp the object.

There are different types of Convolutional Neural Networks (CNN) that analyze an image by producing a very good accuracy. Some of which are Inception V3, C. Szegedy *et al.* [3], AlexNet, A. Krizhevsky *et al.* [1], GoogleNet, C. Szegedy *et al.* [4], VGGNet, K. Simonyan

et al. [12] and so on.

In this paper, a modified and fine-tuned model was used. For my network, in other to solve 11 class recognition problem, a baseline which is VGG19 [12] was used after which another layers was added to this baseline. The new model generated gave an accuracy of 70.3 percent on the test dataset.

The following sections include the data processing, methods used to get the final model, and the conclusion.

2 Data Preprocessing

Preprocessing is the process of transforming data before applying algorithm. The data used in this project includes eleven different classes namely: creamy paste, diced, floured, grated, juiced, julienne, mixed, other, peeled, sliced and whole.

Figure 1: Jullienne

The data was partitioned into training data and validation data. In our work, we performed our experiment on a dataset of size 6348 for training images and a dataset of size 1377 for validation images. ImageDataGenerator was used for data augmentation. Data augmentation is the process of increasing the number of images in our dataset. This method was done so as to rescale our dataset, do

Figure 2: creamy paste

the shear and zoom, flip the images horizontally and vertically, change the width and so on. For the data augmentation, rescaling of $1/255$ was done on the images. We also rotate the images in the range of 180, the width was shifted with range of 0.1, shear and zoom was also done with the range of 0.2, we also flipped the images vertically and horizontally with fill in mode to the nearest. Batch size of 64 was used and 'categorical' was used as the class mode since we have 11 classes. For the training set, we shuffled the data and for the test set, we left the data shuffled . In the next section, we will talk about the methodology used in this paper.

3 Methodology

We have different types of activation function, some of which are ReLu, Sigmoid, softmax, tanh and so on. According to Jarrett *et al.* [8], the rectified linear activation function train deep neural networks by alleviating the difficulties of weight-initialization and vanishing gradients. Relu is defined as

$$F(x) = max(0, x) = \begin{cases} 0, & x < 0 \\ \\ x, & x \geq 0, \end{cases} \quad (1)$$

Forest *et al.*[7] also proposed a more powerful adaptive activation function called the parametrized, piecewise linear activation function which is learned independently for each neuron using gradient descent. We also have softmax. We use softmax when we have multinomial output. Softmax has output probabilities in the range of 0 to 1 with total probability being equal to 1. We define softmax probability as

$$f(x_i) = \frac{Exp(X_i), i = 0, ...k}{\sum_{j=0}^{k}(Exp(X_j))} \quad (2)$$

A modified version of VGG19 convolutional neural networks was proposed in this research for cooking state identification problem. Our modified version of VGG19 network layers include the first 15 layers of the original vgg19 layer, the global average pooling, activation layer, dropout and another activation layer. We added the global average pooling to the vgg19 model so as to minimize overfitting and also to reduce the total number of parameters in the model. After this was done, the activation layer (where ReLu is the activation function) was added with 1024 nodes, and dropout of about 0.5 was added after this so as to reduce overfitting and regularization. According to Astha [11], by using dropout, the nodes in the network become more insensitive to the weights of the other nodes. Finally, another activation layer(with softmax being the activation function was used since this is our final layer) was added with 11 nodes since our output layer has 11 classes.

After creating this new version of VGG19 CNN, we then trained our data on the new model created. We did this training in two stages. Stage 1: We trained the added layers and freeze the remaining layers. We recompiled the model for this modifications to take effect using optimizers, loss function and metrics. One of the aim of deep learning is to reduce the loss function by finding the optimized value for the weights. We used the 'categorical crossentropy' in this paper work for the loss function since we have many classes (actually 11 classes).

We have different types of optimizer namely Adam, SDG, Adamax, Nadam. According to Diederik and Jimmy (2015)*et al.*[6], the name Adam is derived from adaptive moment estimation. Adam is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement.

Table 1: Proposed New model

Layer (Type)	Output shape	Param
Input 1 (InputLayer)	(N, N, N, 3)	0
block1 conv1	(N, N, N, 64)	1792
block1 conv2	(N, N, N, 64)	36928
block1 pool(Maxpooling)	(N, N, N, 64)	0
block2 conv1	(N, N, N, 128)	73856
block2 conv2	(N, N, N, 128)	147584
block2 pool (MaxPooling)	(N, N, N, 128)	0
block3 conv1	(N, N, N, 256)	295168
block3 conv2	(N, N, N, 256)	590080
block3 conv3	(N, N, N, 256)	590080
block3 conv4	(N, N, N, 256)	590080
block3 pool (MaxPooling)	(N, N, N, 256)	0
block4 conv1	(N, N, N, 512)	1180160
block4 conv2	(N, N, N, 512)	2359808
block4 conv3	(N, N, N, 512)	2359808
block4 conv4	(N, N, N, 512)	2359808
global ave pooling2d 1	(N, 512)	0
dense 1 (Dense)	(N, 1024)	525312
dropout 1 (Dropout)	(N, 1024)	0
dense 2 (Dense)	(N, 11)	11275

The optimizer used in this work is Adam. We used Adam to update the weight in other to reduce the loss function. Learning rate can affect the learning efficiency. Therefore, if we use low learning rate, then we won't get our result on time and too high of a learning rate causes instability. In between, the too low and too high, we picked our right learning rate for our optimizers and this trained our model successfully. For the optimizer used, learning rate of lr=0.0001 was used.

In other to judge the performance of our model we used metrics. We have different types of metrics: binary accuracy, categorical accuracy, sparse categorical accuracy. In this project, we used 'categorical accuracy' so as to check if the highest true value is equal to the highest predicted value. We then trained the model on different image sizes using 20 epochs with 200 steps per epoch. Validation steps to train this model was obtained by dividing the number of test set by the number of batch size.

Stage 2: In this stage, we trained only block 4 and we freeze the remaining layers. We recompiled the model for this modifications to take effect using Adam optimizer with learning rate of 0.00001. Also 'categorical crossentropy' was used as the loss function and 'categorical accuracy' as the metrics.

We then trained the model on different image sizes using 20 epochs with 200 steps per epoch.

4 Result

In this work, we used dataset that include images of eleven different classes of cooking state. The data was partitioned into training data and validation data. The training set has images of size 6348 with 11 classes and the validation set has images of size 1377 with 11 classes. We used the training dataset and the validation data set to train our model and we later test our model with the test data. In other to verify our best model, we created a test set of images Jelodar *et al.* [9]. We performed different experiment while training our model so as to get the best model with the highest accuracy. These experiments include: Changing the layers in the model, changing the steps per epoch size and using different learning rate.

Table 2: Tested model

model	opt1(LR), Opt2(LR)	steps	ACC
NewVGG	Adam(1.0^{-4}), SGD(1.0^{-5})	200	0.63
NewVGG	Adam(1.0^{-4}), Adam(1.0^{-4}	100	0.68
NewVGG	Adam(1.0^{-5}), Adam(1.0^{-6})	100	0.58

Changing the layers in the model: Using the vgg-19, we used the first 19th layers then the global average pooling was added to it followed by the activation layer(ReLu as the activation function), with dropout of 0.5 and then added another activation layer(Softmax as the activation function). After the second stage, we got an accuracy of 66percent for our test set. Using our proposed model, with Adam (with learning rate of 0.0001) as the optimizer for our first stage and SGD as the optimizer for the second stage (with learning rate of 0.00001 and momentum of 0.9) and also using the dataset giving, we tested the test set for 20 epochs with steps per epochs of 200 and we got an accuracy of 63 percent.

Using Adam(with learning rate of 0.0001 and 0.00001 respectively) as the

Table 3: Proposed model

model	stage	epoch	Val acc	Val
New model	1	5	0.5171	1.4
New model	1	10	0.5490	1.2
New model	1	20	0.5933	1.1
New model	2	5	0.6558	1.0
New model	2	10	0.6855	0.95
New model	2	20	0.7219	0.8702

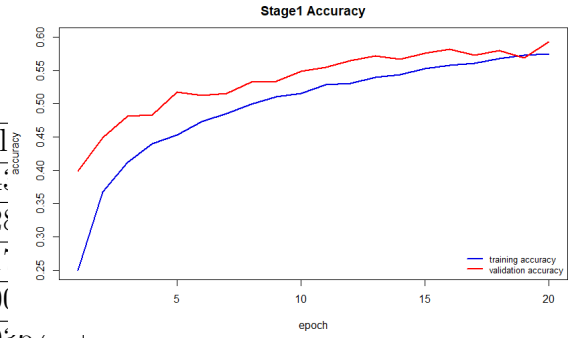

Figure 3: Stage1 accuracy

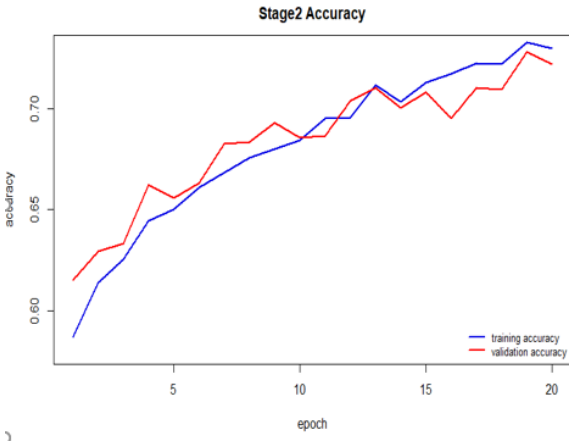

Figure 4: Stag2 accuracy

optimizer for both the first stage and the second stage, we used 20 epochs with 100 steps per epochs and we obtained an accuracy of 68percent for the test data.

Using our proposed model, we increased the learning rate for each of the optimizers used and we obtained a lower accuracy compared to using learning rate of 0.0001 and 0.00001.

After trying all these experiments, we decided to choose the best model(the one with the highest accuracy).

With our proposed model, we trained the model on different image sizes using 20 epochs with 200 steps per epoch. Adam was used as the optimizer for the first stage with learning rate of 0.0001. For this first stage, we got an accuracy of 59.33percent with validation loss of 1.17. We moved to the next stage and for this stage, we also used Adam as our optimizer but with different learning rate of 0.00001. We got our accuracy to be 72.2percent with validation loss of 0.87. The accuracy for both stages and the loss are given in Fig(3-6)

We tested our model on the test set and we obtained and accuracy of 70.3 percent

5 Conclusion and Future work

In this work, a new model was proposed, discussed and introduced extensively. This model was designed using the vgg-19 as the baseline. We added some layers to this baseline to get our proposed model. We trained the model in two stages using the training and the validation dataset. This dataset consists of 11 classes of object state. The training set consists of 6348 training images and the validation sets consists of 1377 validation images. Augmentation was done to resize the shape of the images. Adam optimizers was used for this two stages with different learning rates. We tried different method

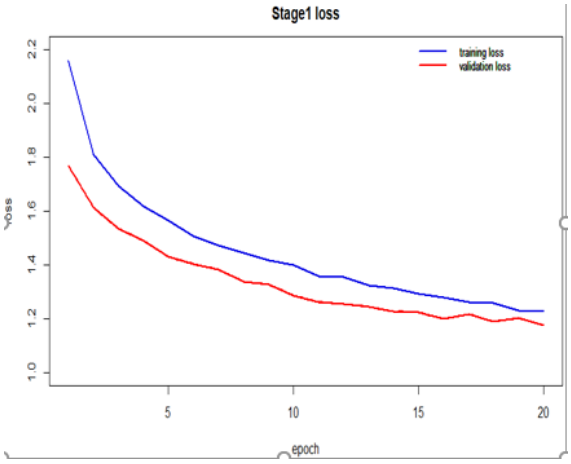

Figure 5: Stage loss

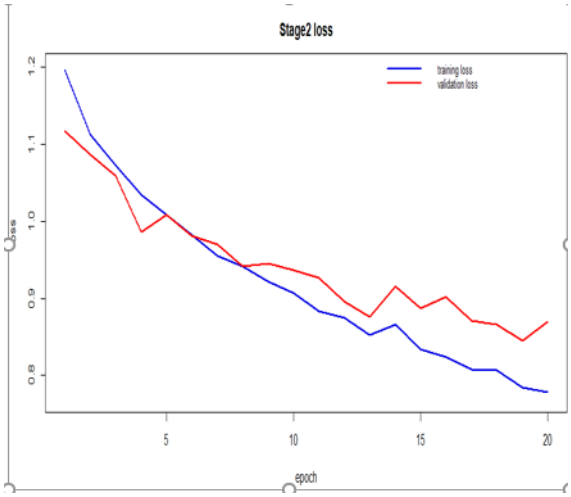

Figure 6: Stage2 loss

to get the best model by using different learning rate, steps per epoch, optimizers. We then chose this our proposed model as the best model because it classify the test set with the highest accuracy of 70.3 percent. For our future work, we can try Inception V3. We can use this inception V3 as our baseline and then add some layers to it like we did in this paper. We can also use different optimizer e.g RMSProb to see if we will get higher accuracy.

References

[1] I. Sutskever A. Krizhevsky and G. E. Hinton. Imagenet classification with deep convolutional neural networks, in advances in neural information processing systems,. page 1097–1105, 2012.

[2] R. Arbiol and V. Pala. Advanced classification techniques: A review. *Institute Cartogràfic de Catalunya (ICC), Parc de Montjuïc, E-08038 Barcelona, Spain, Y. Zhang, University of New Brunswick, Department of Geodesy and Geomatics engineering, P.O. Box 4400, Fredereicton NB, E3B 5A3, Canada*, 42(10):2159–2179, 2015.

[3] S. Ioffe J. Shlens C. Szegedy, V. Vanhoucke and Z. Wojna. Rethinking the inception architecture for computer vision. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 111:2818–2826, 2016.

[4] Y. Jia P. Sermanet S. Reed D. Anguelov D. Erhan V. Vanhoucke A. Rabinovich C. Szegedy, W. Liu. Going deeper with convolutions. cvpr. 2015.

[5] Tianze Chen. Identifying states of cooking objects using vgg network. 2018.

[6] Jimmy Lei Ba. Diederik P. Kingma. Adam: A method for stochastic optimization. *Published as a conference paper at ICLR, arXiv:1412.6980v9 [cs.LG]*, 2015.

[7] Peter Sadowski Pierre Baldi Forest Agostinelli, Matthew Hoffman.

Learning activation functions to improve deep neural networks. accepted as a workshop contribution at iclr. *arXiv:1412.6830v3 [cs.NE] 21*, 2015.

[8] Kavukcuoglu Koray Ranzato M Jarrett, Kevin and Yann LeCun. What is the best multi-stage architecture for object recognition? *In Computer Vision, IEEE 12th International Conference*, page 2146–2153, 2009.

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

[10] Y. Lin and Y. Sun. *Grasp planning based on strategy extracted from demonstration in Intelligent Robots and Systems (IROS 2014)*. 2014.

[11] Astha Sharma. State classification with cnn.

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

[13] ujjwalkarn. An intuitive explanation of convolutional neural networks. 2016.

[14] Zhang L. Huang B. Li P. Zhong, Y. An unsupervised artificial immune classifier for multi/hyperspectral remote sensing imagery. *Transaccions on Geoscience and Remote Sensing*, 44(2), 2006.