Transfer Learning for Object State Classification (March 2019)

Ivan Shindev, University of South Florida

***Abstract*—There have been many state-of-the-art neural network developments in recent years that have been trained to achieve great accuracy by using a lot of data and GPU processing power. In this paper, we examine the use of pre-trained neural network for fast and accurate prediction of new models. This paper used image object state classification as a test model. It demonstrates different ways a neural network can be constructed by using the pre-trained neural networks ResNet [3], InceptionV3 [1] and VGG19 [7]. It further explores fine tuning techniques to improve accuracy. The final neural network model had 62.8% accuracy on the validation data and 70.4% accuracy on the training data.**

***Index Terms*— networks, neural**

I. Introduction

To properly create and train a neural network, that can accurately represent and predict a model, one would need a lot of training data and GPU processing power. Unfortunately, this type of processing power and data collection can be very expensive and time consuming. Depending on the task, getting the necessary training data can be very complicated and most likely will require resource consuming annotation. In recent years, very accurate neural network models have been designed and trained on the ImageNet data [5]. ImageNet represents a database of thousands of images, training a neural network with this amount of data results in very accurate networks such as VGG19 as reported by Simonyan and Zisserman [7]. This type of convolutional neural networks can effectively learn complex pixel relationships and patters. This paper explores using the feature detection layers of these pre-trained neural networks as an alternative to building a neural network from scratch. It further evaluates fine tuning techniques to improve accuracy.

In this work, an object state classification in images is being explored. Publicly available neural networks are used in combination with each other and by themselves to find the most accurate pre-trained model. The models used are (Diagrams 1-6):

- VGG19
- InceptionV3
- ResNet50
- VGG19 + InceptionV3
- VGG19 + ResNet50
- ResNet50 + InceptionV3

The idea behind using two networks together is that one network might have learned features that the other hasn't. This experiment consists of two stages. In stage one, the pre-trained networks are used without their top output layers and an identical structure of dense layers is trained as a top layer to the previous mentioned models. In stage two, the most accurate model is fine tuned.

II. Related Work

Choi et al. report a model designed for retinal disease detection using a pre-trained VGG19 network [2]. According to Choi et al., using transfer learning yield more accurate results in retinal disease detection than using other machine learning methods or training a neural network from scratch.

Lagunas and Garces use the VGG19 neural network with a State Vector Machine to create a machine learning model with 86.61% accuracy in image illustration classification [4]. This is another example of how transfer learning can help create accurate neural network models.

Both papers argue that transfer learning reduces training time and improves accuracy. Similar to this paper's approach, multiple stage training can be observed in Lagunas and Graces paper. The learning rate reduces as the layers get closer to the input layer. The idea behind this is that networks like VGG19, trained on large dataset such as ImageNet, already have generalized and optimized weights in the first couple of layers and should not be changed a lot in order to repurpose the network.

III. Methodology

The goal is to find a model that can accurately classify

objects in images into 11 distinct states. The object are different kind of food and the states are food state classification such as whole, chopped, sliced, crushed etc. Training and testing data were gathered using video annotation software and cooking video clips from a previous semester class students. A total of 6348 images belonging to 11 classes were used for training purposes.

Anaconda 3 was used as the main programming platform on a Windows 10 Pro operating system with Intel Core i9-9900k and Nvidia GeForce GTX 1060 ti (6GB). Keras with Tensorflow GPU backend was used to train and test the neural network models. Each stage training took 20-30 min and there were total of six experiment with only one stage of training and testing and a total of ten experiments with three stages of training and testing. Stage one training and testing includes freezing all base layers of the pre-trained neural network and training only the newly added top layers with a learning rate of 0.01 for 400 epochs. The second stage training and testing includes freezing the first 15 layers of the network and training the model with a learning rate of 0.0001 for 100 epochs. The third stage training and testing includes freezing the first 10 layers of the network and training the model with a learning rate of 0.0001 for 200 epochs. The accuracy was reported for both training and validation dataset for each stage. Other tests included variation in the learning rate, data augmentation, optimizer, L1 and L2 regularization and layer modifications.

Each pre-trained network has a 224x224 input image layer, so the data augmentation used converts each input image to a 224x224 image with variations in shear and zoom range. For networks that require two inputs, the Keras flow_from_directory class was used along with a custom function to provide the same image as input to two different networks (Diagrams 4-6).

The first three experiments include using the base, pre-trained, networks VGG19, InceptionV3 and ResNet50 with ImageNet weights and replacing the top dense layers with a Global Average Pooling layer, two fully connected layers with ReLu activation function, a dropout layer and finally a SoftMax classification layer with 11 classes (Diagrams 1-3). The next three experiments include combining the base, pre-trained, networks in pairs of two and replacing the top dense layers with a Global Average Pooling layer in both networks. The output of the two Global Average Pooling layers was then concatenated and used as an input to two fully connected layers with ReLu activation function, a dropout layer and finally a SoftMax classification layer with 11 classes (Diagrams 4-5).

The accuracy from the first six experiments was used to select a model for finetuning. VGG19 was selected for finetuning and the following experiments were performed:

- L1 and L2 regularization
- Adding and Removing Dropout Layers
- Adam vs. RMSprob Optimizer
- Learning rate variation

The final network (Diagram 7) was selected based on accuracy and robustness by comparing the training vs. validation set accuracy.

IV. Data and Network Models

Diagram 1 represents the first network structure used in the experiment. The base VGG-19 network represents the pre-trained network with ImageNet weights and without the top SoftMax and Dense layers. The ImageNet database contains images that belong to a total of 1000 object classes. Since our dataset contains only 11 classes and the category of the class is object state, we cannot use the pre-trained Dense and SoftMax layers. A Global Average Pooling layer is added to reduce the dimension of the last layer. This is very important for experiment 4-6, to concatenate the two networks the layers must have the same dimension over the axis that is not being concatenated. A Dropout layer is also added after the two fully connected layers to help reduce overfitting.

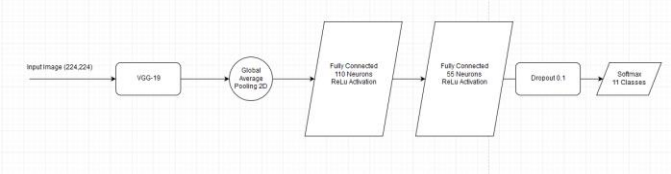

Diagram 1. VGG-19 pre-trained network with new top layers

Diagram 2 and 3 represents the second and the third experiment and are the same as Diagram 1 with only difference being the pre-trained base networks is ResNet/InceptionV3 instead of VGG-19

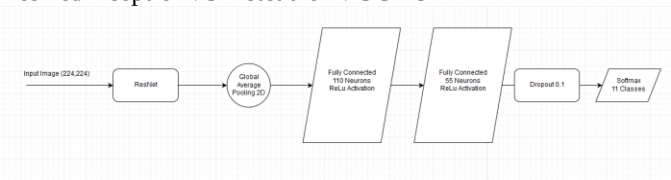

Diagram 2. ResNet pre-trained network with new top layers

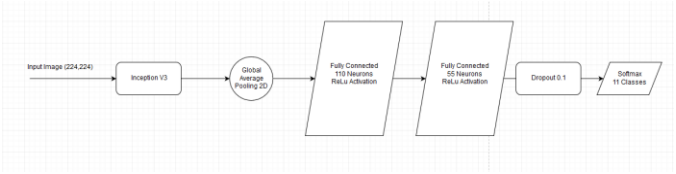

Diagram 3. Inception V3 pre-trained network with new top layers

Diagram 4 represents the neural network structure for the fourth experiment. Here, two base neural networks (ResNet and Inception V3) are being fed the same image as an input, and the output is being concatenated into the same output networks as in Diagram 1, 2 and 3.

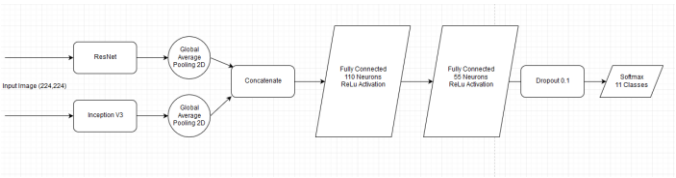

Diagram 4. ResNet and Inception V3 pre-trained networks with Global Average Pooling layers, Concatenate Layer and new top layers

Diagrams 5 and 6 represent the fifth and the sixth experiment and are the same as Diagram 4 with the only difference being the pre-trained base network pairs are VGG-19 and InceptionV3/VGG-19 and ResNet instead of ResNet and Inception V3.

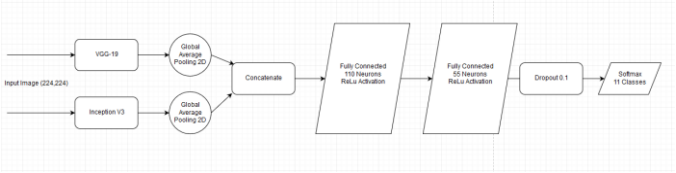

Diagram 5. VGG-19 and Inception V3 pre-trained networks with Global Average Pooling layers, Concatenate Layer and new top layers

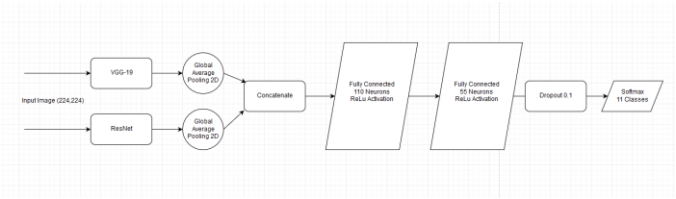

Diagram 6. VGG-19 and Resnet pre-trained networks with Global Average Pooling layers, Concatenate Layer and new top layers

Table 1 represents the accuracy obtained from Stage 1 training and testing in the first six experiments. The structure in Diagram 1 was identified as the most accurate and Stage 2 and 3 training was performed. The accuracy from Stage 2 and 3 is reported in Table 2.

Network	Train Data Accuracy	Validation Data Acuracy
VGG-19	54.74%	51.85%
Inception V3	14.39%	16.26%
ResNet	38.56%	39.21%
VGG-19 + ResNet	36.79%	35%
VGG-19 + Inception V3	8.88%	8.93%
ResNet + Inception V3	13.43%	15.61%

Table 1. First six experiments Stage 1 accuracy

Training Stage	Train Data Accuracy	Validation Data Acuracy	Layers Frozen
VGG-19 Stage 1	54.74%	51.85%	All VGG-19 layers frozen
VGG-19 Stage 2	50.37%	48.22%	First 15 layers frozen
VGG-19 Stage 3	57.10%	54.10%	First 10 layers frozen

Table 2. VGG-19 Stage accuracy for Diagram 1 Network

Diagram 7 represents the final network model after performing fine tuning. The final model was trained with Stage 1, 2 and 3 and the accuracy is reported in Table 3.

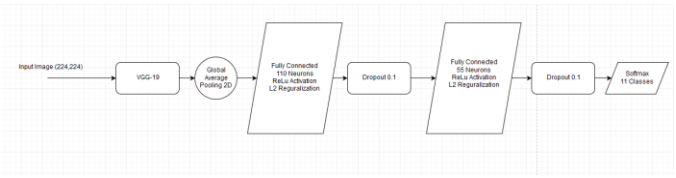

Diagram 7. VGG-19 Final Network Structure

Training Stage	Train Data Accuracy	Validation Data Acuracy	Layers Frozen
VGG-19 Stage 1	46.78%	46.40%	All VGG-19 layers frozen
VGG-19 Stage 2	58.20%	53.23%	First 15 layers frozen
VGG-19 Stage 3	70.41%	60.63%	First 10 layers frozen

Table 3. VGG-19 Stage accuracy for Diagram 7 Network

V. Results

From the three base models we can see that VGG-19 had the highest accuracy (54.74% on training data and 51.85% on validation data) during Stage 1 training and Inception V3 had the lowest accuracy (14.39% on training data and 16.26% on validation data) (Table 1). An interesting result from the first six experiments is that the Inception V3 and ResNet50 had higher accuracy on the validation data then on the training data. VGG-19 did not perform better when paired with ResNet or Inception V3. This might be due to the Dropout layer where it is possibly dropping out more information coming in from VGG-19 thus reducing the accuracy. This is highly likely since ResNet50 and InceptionV3 both have 2048 features after the Global Pooling Average layer where as VGG-19 has only 512 features after this layer. This means that concatenating VGG-19 with either of the other two networks will make it less dominant.

Because of the high accuracy of the base VGG-19 network, it was chosen for fine tuning. To evaluate the baseline for tuning, Stage 2 and 3 training was performed on the Diagram 1 structure. Stage 2 training resulted in reduced accuracy in both the training data and validation data testing (50.37% and 48.22% respectively in Table 2). We can see that we did not get a lot of variation in accuracy between the testing and the training data and this might be due to the Dropout layer. Stage 3 training resulted in slightly improved accuracy and not much variation in testing vs. validation accuracy. The reduced accuracy in Stage 2 might be due to weight variation so L1 and L2 regularization were tested on the two Dense layers. L2 regularization provided a better accuracy increase but slightly decreased the robustness. To increase robustness another Dropout layer was added which gives us the final neural network structure (Diagram 9). Between Stage 1 and Stage 3 training the accuracy almost double for the final structure (Table 3). Overfitting also increased from Stage 1 to Stage 3 so further training was not performed in order to prevent additional overfitting.

The loss function used was categorical cross entropy and the optimizers used were RMSProb and Adam. The Adam

optimizer seemed to give better accuracy and less variation in the loss function during Stage 2 and Stage 3 training where in Stage 1 training both RMSProb and Adam optimizer performed within the same range.

VI. Conclusion and Future Work

Training a neural network from scratch requires a lot of GPU processing power and resources that we might not always have. This paper evaluates using pre-trained neural networks for building an acceptable neural network model for object state classification in images. The final accuracy achieved on the validation set was 60.63%. This accuracy can most likely improve with modification to the neural network top output layers. Future work includes using parts of Inception V3 or ResNet50 in combination with the base VGG-19 network. Using parts of Inception V3 and ResNet50 with less features may help in making VGG-19's features more dominant in the classification.

References

[1] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2] J. Y. Choi, T. K. Yoo, J. G. Seo, J. Kwak, T. T. Um, and T. H. Rim, "Multi-categorical deep learning neural network to classify retinal images: A pilot study employing small database," Plos One, vol. 12, no. 11, 2017.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[4] Lagunas M. and Garces E., "Transfer Learning for Illustration Classification," Spanish Computer Graphics Conference, 2017.

[5] Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, et al. "ImageNet Large Scale Visual Recognition Challenge." Int J Comput Vis., 2015.

[6] Shallu and R. Mehra, "Breast cancer histology images classification: Training from scratch or transfer learning?," ICT Express, vol. 4, no. 4, pp. 247–254, 2018.

[7] Simonyan K. and Zisserman A., "Very deep convolutional net-works for large-scale image recognition," CoRR abs/1409.1556, 2014.