**Transfer Learning**

Transfer learning was applied to the fruits 360 dataset, in a second Jupyter notebook, by using a pretrained model (VGG16). VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in their paper “Very Deep Convolutional Networks for Large-Scale Image Recognition” (ul Hassan, 2018). The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes (ul Hassan, 2018). It was one of the models submitted to ILSVRC-2014 (ul Hassan, 2018). The architecture of the VGG16 network is below:

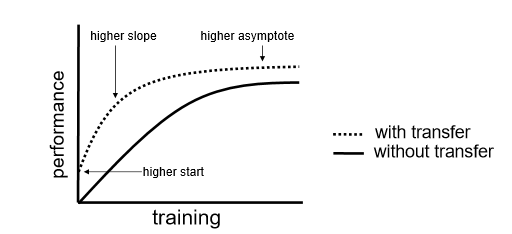


VGG16 Architecture

Retrieved from:<https://neurohive.io/en/popular-networks/vgg16/>

In transfer learning, a pre-trained model (i.e. the weights and parameters of a network that has been trained on a large dataset by somebody else) is implemented and then “fine-tuned” with the dataset of interest (the fruits-360 dataset) (IS, 2018). The pre-trained model will either provide the initialized weights leading to a faster convergence or it will act as a fixed feature extractor for the task of interest (IS, 2018). In this case, the pretrained VGG16 model provides the initialized weights and then the network was retrained on the fruit-360 dataset.

There are three benefits to transfer learning as shown in the figure below. First, there is a higher start or initial performance achieved compared to the initial performance of a model that is trained from scratch (Torrey & Shavli, n.d). Second, the amount of time it takes to learn the target task is shorter resulting in a higher slope (Torrey & Shavli, n.d). Third, a higher ﬁnal performance level is achieved (i.e. higher asymptote in the figure below) (Torrey & Shavli, n.d).



<http://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf>

The VGG16 pretrained model can be imported from keras.applications. When importing the model, certain arguments must be specified. Since we want to provide our own fully connected input layer, include\_top is set to False. This will exclude the first 3 fully-connected layers at the top of the network (Keras, n.d). The input shape can then be specified. It should have 3 input channels, and width and height should be no smaller than 32 (Keras, n.d). An input\_shape of (64,64,3) was used in this model. The weights were set to 'imagenet' meaning the weight were not randomly initialized but were based on the trained ImageNet model (Keras, n.d).

To prevent that the model from retraining the imported existing layers in the VGG16 model, layer.trainable is set to False.

The model is then constructed in a similar fashion as the previous CNN model. A flattening layer is added, and an output layer with units =111 and activation function = softmax is specified. The model is compiled and then trained in the same fashion as the original CNN model.

Results of Transfer Learning: Results of Original Model:

Time to Early Stopping ~ 20 hours Time to Early Stopping ~ 12 hours

Epochs = 41 Epochs = 38

|  |  |  |
| --- | --- | --- |
|  | Accuracy (%) | Loss |
| Train Set: | 99.1 | 0.033 |
| Test Set: | 95.8 | 0.140 |

|  |  |  |
| --- | --- | --- |
|  | Accuracy (%) | Loss |
| Train Set: | 97.8 | 0.072 |
| Test Set: | 96.3 | 0.137 |

Since this model took longer to run, and the original CNN model showed a bit of overfitting, the patience argument was set to 10 and not 20 epochs (as was set in the original model).

In comparing the 2 models, the training accuracy was higher and the loss was lower when using transfer learning. The three effects as describes above (higher start, higher slope and higher asymptote) were all seen in the training set when using the transfer learning method. However, the test set accuracy and loss were very similar between the two methods.

Although the training accuracy was higher for the transfer learning method, the time to early stopping was nearly double. If these models were run using a GPU, these computational times may be much faster. However, based on the computation time and the results of the test set accuracy and loss, transfer learning did not provide improvement over running the model from scratch.

**References**

IS (2018). *Fruits-360 - Transfer Learning using Keras and ResNet-50*. Retrieved July 14, 2019 from :<https://www.kaggle.com/amadeus1996/fruits-360-transfer-learning-using-keras>

Keras Documentation (n.d). *Applications*. Retrieved July 12, 2019 from: <https://keras.io/applications/#vgg16>

Torrey, L. & Shavli, J. (n.d) *Transfer Learning*. University of Wisconsin, Madison, WI. USA. Retrieved July 14, 2019 from: <http://ftp.cs.wisc.edu/machine-learning/shavlik-group/torrey.handbook09.pdf>

ul Hassan, M. (2018). *VGG16 – Convolutional Network for Classification and Detection.* Retrieved July 14, 2019 from: <https://neurohive.io/en/popular-networks/vgg16/>