Use of Pretrained Models and Recent Architectures on Equus Classification

Miguel Corona

April 2020

1 Introduction

It is known that building a deep learning network that generalizes is a difficult task. Studies and advancements in the field have yielded promising network architectures that may be further improved upon. These architectures have established benchmarks that other models attempt to meet and potentially surpass. A benefit of deep learning is the portability of neural networks that permits the reuse of network components onto different problems or the fundamental architectures that can be applied into the development of new architectures. Observations were performed on the Equus classifier as it was trained using the features identified from a pretrained VGG16. Observations of an Equus classifier with the use of residual connections were also performed.

2 Pretrained Architecture

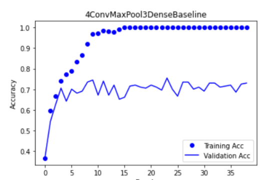
Pretrained networks are networks whose parameters have been saved after training on their respective dataset. The use of pretrained networks is an effective strategy to develop deep learning on a limited sized dateset as the Equus dataset even if the outputs of the pretrained network and Equus classifier are disjoint. The information learned from the pretrained network can be utilized by the developed classifier in its training via feature extraction and fine-tuning. Both strategies involve the use of the convolution base of a pretrained network to be connected onto the dense network.

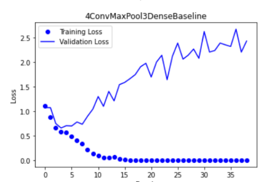
3 Experiment

The objective of the experiment was to observe the performance on the Equus classifier with the use of a pretrained network's convolution base instead of its existing convolution base. The Equus classification model that was determined to yield the best performance is given by the architecture:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 12)      336
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 127, 12)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 125, 125, 12)      1308
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 62, 62, 12)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 60, 32)        3488
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 24)                602136
_________________________________________________________________
dense_1 (Dense)              (None, 12)                300
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 39
=================================================================
Total params: 616,855
Trainable params: 616,855
Non-trainable params: 0
```

The model was trained on the development set without data augmentation to yield the benchmark results:





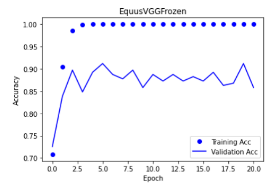
Dataset	Loss	Accuracy
Training	0.5632	0.7705
Validation	0.7661	0.6569
Test	0.5760	0.7217

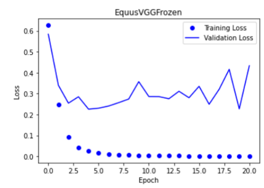
The benchmark execution indicates that the model has statistical power on the data. The model's drawback is that it overfits the data early on in the training as depicted in the loss learning curves and that the model's accuracy plateaus at 70% accuracy on the validation set.

3.1 Feature Extraction

Feature extraction is the practice of utilizing the patterns learned from the convolution base of a pretrained network to train a classifier. The convolution base knows generic patterns from its training that can be applied onto another classifier such as the Equus classifier. The VGG16 convolution base was selected for feature extraction to feed its recognized features into the Equus classifier.

The layers within the convolution base were frozen to ensure that only the trainable parameters of dense network were modified across the learning iterations. The dense classifier was trained with the frozen weights to yield:



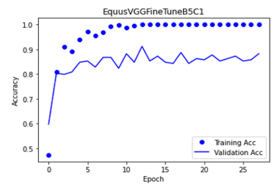


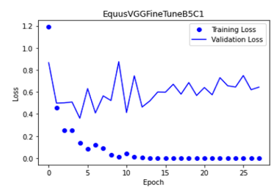
Dataset	Loss	Accuracy
Training	0.0151	1.0000
Validation	0.2398	0.8971
Test	0.2199	0.9087

The results indicate that the classifier benefited from the use of the VGG16 convolution base. The patterns known by the convolution base were utilized in the training of the dense classifier and its performance and accuracy across the datasets improved compared to those of the benchmark. The loss values acquired with feature extraction are minimized to yield stronger accuracy metrics that the benchmark could not achieve. The learning curves indicate that the validation loss fluctuating during the epochs, but was bound as the loss never reached the extent of the benchmark. The loss learning curves indicate that the model overfits the data as is expected due to the small sample size and the omission of data augmentation. The accuracy on the training set converges to its maximum value and the accuracy of the validation set improves considerably with respect to the benchmark. The convolution base of the VGG16 model is portable and can be used additional classifiers.

3.2 Fine-Tuning

Fine-tuning is the complement to feature extraction in which components of the convolution base are trained along with the classifier albeit asynchronously at first. The classifier is initially trained with frozen convolution base layers to prevent the error from the initial stages of training from influencing the layers in the convolution base. Layers from the convolution base are then unfrozen and trained along with the classifier to inject information of the problem space onto the unfrozen layers. The weights that contribute the lowest loss on the validation set were loaded onto the model and layers within the convolution base were unfrozen to permit their training. The experiment contains the unfreezing of the first layer in the fifth convolution block and yields:





Dataset	Loss	Accuracy
Training	0.0753	0.9770
Validation	0.4425	0.8088
Test	0.1838	0.9348

The results indicate that the classifier benefited with the use of fine-tuning with the VGG16 convolution base. An improvement in performance was observed compared to the benchmark, but the performance in loss and accuracy did not match those of feature extraction. The loss and accuracy improved on the test set, but the loss and accuracy of the validation set decreased. It is not believed that loading the weights from feature extraction prior to fine-tuning contributed to the degraded results as the feature extraction classifier held strong performance on the entire dataset. Deviations in the accuracy and loss between the feature extraction execution and the fine-tuning execution may be due to the added complexity involved in training. The unfrozen layers trainable parameters increased the complexity in the model as more weights were adjusted per iteration compared to the number of weights in feature extraction. A complex model has difficulty generalizing the data. Even though the model is more complex it training, it's performance outmatched those that did not utilize the VGG16 convolution base. The learning curves indicate that the the accuracy on the validation set plateaus at a higher value than the benchmark at 80%. The loss on the validation set also fluctuates, but is bound as it was found in feature extraction.

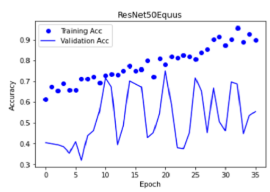
4 ResNet Architecture

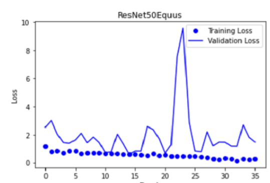
The ResNet architecture is a deep network, exceeding the depth of the VGG16, that introduced residual connections. The residual connections permit the network to have knowledge of previous representations of the data by reinjecting the data downstream. the reinjection is performed by adding the output tensor from a previous layer to a later output to prevent information loss that may occur as the data is processed across the network.

5 Experiment

5.1 ResNet50

A ResNet50 model was instantiated without its ImageNet weights so that the initial weights were set randomly. The model was also instantiated to be aware of the three classes of interest: horse, zebra, and donkey. The training was performed with the datasets of the Equus classifier. The ResNet50 model was trained to yield the following:



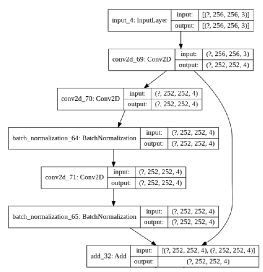


Dataset	Loss	Accuracy
Training	0.5271	0.8000
Validation	0.6267	0.7108
Test	0.5404	0.7652

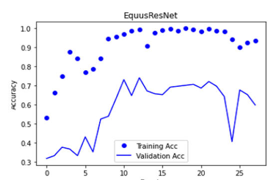
The results indicate that the ResNet50 model was capable to outperform the benchmark results. The loss and accuracy improved in the ResNet50 execution although it appears as though the model can benefit from further executions. The training of the ResNet50 model stopped prematurely due to the EarlyStopping callback and it was observed that the architecture requires a large number of training iterations to converge to a suitable model compared to the benchmark. The accuracy learning curves gradually increase on the training set, but oscillate in the validation accuracy. The updates in the the parameters have a drastic change as denoted by the magnitude of the fluctuations. The loss learning curves indicate that the validation loss is bound to the loss of the training set with a minor exception in which the loss spikes around epoch 23. Deviations exist between the losses as the loss also appears to oscillate across the epochs. These oscillations are believed to be due to the complexity of the model as the number or parameters increased to 23,540,739 parameters compared to the 616,855 parameters of the benchmark. The ResNet50 model shows promise of yielding a suitable model after sufficient training.

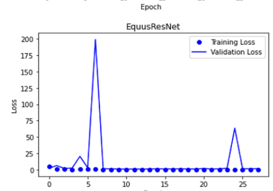
5.2 Residual Connections

An implementation of the Equus classifier was developed with the use of residual connections. Each convolution layer contains four filters and a five by five kernel size for depth of 17 layers prior to entry into the dense network. A residual block for the network is depicted as:



The model was trained on the Equus dataset to yield the following:





Dataset	Loss	Accuracy
Training	0.0592	0.9852
Validation	0.8215	0.6520
Test	0.8465	0.6739

The results of an Equus classifier utilizing residual connections was unable to outperform the benchmark in terms of the loss across the validation and test sets. The residual connections permitted the network to yield strong performance on the training set indicating the the model overfit the data. The accuracy learning curves indicate that the validation accuracy is no better than the the benchmark as it is shy of the 70% mark established by the benchmark. The loss learning curves indicate that the validation loss is bound to the training loss with the exception of a few points in which the loss exceeded any previously observed loss. The magnitude of the loss was larger in models containing residual connections as ResNet50 also contained large loss. The subpar performance may be due to the configuration of the model as the ResNet architecture is described to support a large number of filters whereas the Equus classifier residual connection configuration only contains four filters per layer. A low number of filters is more suitable for a dense network, but this configuration was the one found to yield the smallest loss on the validation and test sets. Model complexity played a role in the performance of the model and the use of four filters provided the model whose loss was comparable to those of the benchmark. Increasing the number of filters yielded larger loss values across the datasets.

6 Conclusion

The experiments indicate that the reuse of the convolution base of a pretrained network may benefit the Equus classifier. The use of the VGG16 convolution base for feature extraction yielded the most benefit in loss and accuracy across the datasets indicate that the general features learned from the pretrained network were useful for the Equus' dense network. Fine tuning also yielded beneficial results compared to the benchmark albeit not meeting those of the feature extraction. The deviation in results between feature extraction and fine-tuning may be attributed to the complexity involved in the model's training as the number of trainable parameters increased dramatically between these executions due to the unfreezing of layers. The experiment also indicates that recent architectures like the ResNet architecture can generalize the problems and yield strong performance to match those of a custom convolution neural network. The drawback in using the ResNet architecture is the time involved in the training as it is much slower to converge to a suitable model. The ResNet50 model was capable of learning from the Equus dataset, but the Equus classifier adopting residual connections failed to outperform the benchmark. Modifications may need to be performed on the Equus model utilizing residual connections for further assesment.