Homework 4: Fine-tuning

Akwasi D. Akwaboah

April 2021

Abstract

Face verification is an integral part of our modern society. It transcends electronic access control and security systems in various institutions as well as the lives of individuals. Machine learning models have shown prowess at distinguishing disparate subjects from much simple algorithms such as nearest neighbor, eigenfaces, Fisherfaces and sparse representation-based classifier algorithms. The multiclass classification method for recognizing distinct faces, however, is known to produce low recognition rates. To address this, one-shot learning approach involving the use of siamese network-based discriminators are rather used. In this homework, I explore face verification by this approach using pretrained convolutional neural networks – AlexNet and VGG16 on the Labeled Faces in the Wild (LFW) Dataset. I present findings from the performing face verification with original pre-trained and fine-tuned versions of the CNN models.

1 Introduction

Deep learning models have significantly advanced automated identification and verification tasks in the recent times. While this is true, these models often require a lot of compute. A common practice in the field to circumvent this requirement is the use of pretrained models along with fine-tuning. Here, models are initialized with pretrained weights from training on sometimes different data domains. While the apparent domain disparities may seem to detract recognition of new data types, this is often not the case as models such as CNNs like the human visual cortex are essentially good at edge detection. Fine-tuning pretrained models can be executed diverse ways including the freezing a section of layers, while others are trained. In this homework, I demonstrate the relevance of leveraging pre-trained CNNs, namely AlexNet [1] and VGG16 [2] in a Siamese network-based discriminator for the Labeled Faces in the Wild Dataset. I equally present findings from fine-tuning the suc models.

2 Dataset

The Labeled Faces in the Wild dataset [3] is an open-source dataset popular for its use as in benchmarking face verification tasks typically formulated as pair matching problem instead of a multiclass classification. It is composed of over 13,000 examples of two or more images for each of the 1680 popular people on the internet used. These face images were prepared using the Viola-Jones face detector algorithm. The Dataset was downloaded and paired using the scikit.datasets.fetch_lfw_pairs() module. Pairing involving generating positive (same person) and negative (different persons) with corresponding labels. Samples of both pairing have been shown in Figure 1.

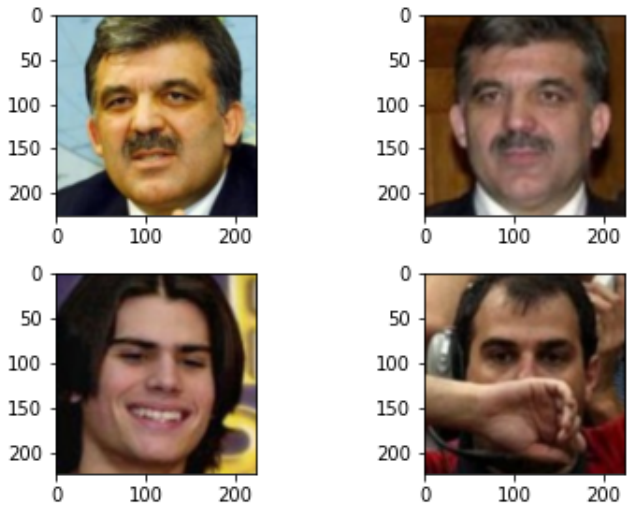

Figure 1: Samples of the LFW Dataset. Top Row: Positive Pairing; Bottom Row: Negative Pairing

3 Methods

The Siamese network was implemented using tensorflow keras which contains the VGG16 model callable by the keras.applications.vgg16.VGG16() function but however lacked the AlexNet pre-trained model as such the AlexNet architecture was replicated and weights from caffe, which on the other hand had AlexNet pretrained weights, were converted and uploaded to the replicated keras model. For fine-tuning, the binary cross-entropy loss was used as goal was to determined whether pairs matched or not. The RMSprop optimizer was used along with a batch_size of 4. A train-validation split of 80 : 20% was used. The network was trained over 100 epochs. [4] was useful reference for creating and training the siamese network. The last layer of the pretrained models corresponding to the 1000-way classification for the The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), thus a 4096 feature is extracted for each sister network of the Siamese network. Appropriate input image resizing was done at source by specifying custom slicing of 224×224 and 227×227 for VGG16 and AlexNet respectively. IN both cases all 3 RGB channels were used.
Fine-tuning was done by making the connections to the final fully connected layer(fc2, 4096) trainable, consequently resulting in 16.7 million trainable parameters out of the total 134 million parameters. The remainable layers had weights frozen during training. The performance of the fine-tuned model were quantified with respect to the initial pre-trained model (as control) using the Area Under Curve (AUC) of the Receiver Operator Characteristic (ROC) curve generated using the sklearn.metrics() module. Training was done in Google Colab.

4 Results and Discussion

4.1 VGG16

Fine tuning strategies described in the methods section were applied to the VGG16. Figures 2 and 3 present the model summary for the initial pretrained network (control) and the fine-tuning version respectively.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
input_2 (InputLayer)	[(None, 224, 224, 3)	0	
model (Functional)	(None, 4096)	134260544	input_1[0][0] input_2[0][0]
lambda (Lambda)	(None, 1)	0	model[0][0] model[1][0]
dense (Dense)	(None, 1)	2	lambda[0][0]

Total params: 134,260,546
Trainable params: 2
Non-trainable params: 134,260,544

Figure 2: Summary for Control: initial pretrained network without fine-tuning

Layer (type)	Output Shape	Param #	Connected to
input_5 (InputLayer)	[(None, 224, 224, 3)	0	
input_6 (InputLayer)	[(None, 224, 224, 3)	0	
model_2 (Functional)	(None, 4096)	134260544	input_5[0][0] input_6[0][0]
lambda_1 (Lambda)	(None, 1)	0	model_2[0][0] model_2[1][0]
dense_1 (Dense)	(None, 1)	2	lambda_1[0][0]

Total params: 134,260,546
Trainable params: 16,781,314
Non-trainable params: 117,479,232

Figure 3: Summary for fine-tuned model: Training only the final fully connected layer

The training and validation curves for training the fine-tuned network have been shown in Figure 4. While there is apparently overfitting, the performance of the face verification improved relative to the control pretrained model with an AUC of 0.74 relative to the 0.5 indicative that fine-tuning adjusted weights to be extract descriptor features for better verification.

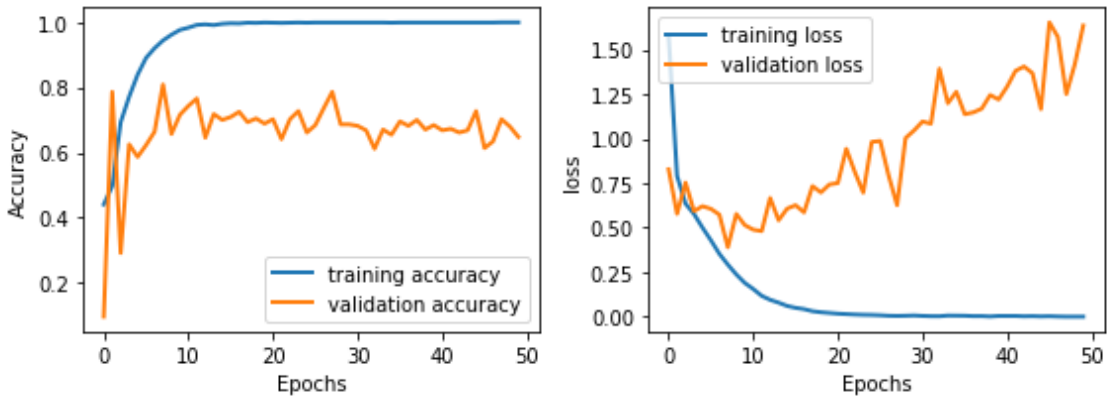

Figure 4: Training metrics: accuracy and loss plots

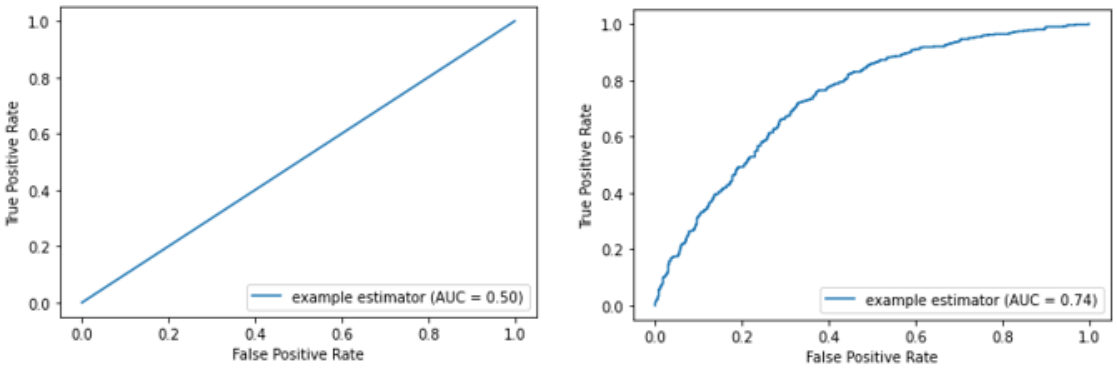

Figure 5: VGG16 – Outcomes of fine-tuning in comparison with initial pretrained model

4.2 AlexNet

Similarly, the parameter counts and similar fine-tuning strategy (captured in the number of trainable parameters) are presented in Figures 6 and 7.A similar trend in training and validation curves for fine-tuning the AlexNet model as this resulting in a better ROC than the control albeit at a lower value relative to the VGG16. This is intuitively reasonable as VGG16 was a sequel to AlexNet and incorporated improvements. Training and Testing results are shown in Figures 8 and 9

5 Conclusion

In this Homework, I present a siamese-network based one shot learning approach for executing Face verification. Fine-tuning of pretrained models are explored. Code available at `https://github.com/Adakwaboah/Face_Verification`

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================
input_1 (InputLayer)         [(None, 227, 227, 3)      0
__________________________________________________________________________________
input_2 (InputLayer)         [(None, 227, 227, 3)      0
__________________________________________________________________________________
model_1 (Functional)         (None, 4096)              56869632    input_1[0][0]
                                                                   input_2[0][0]
__________________________________________________________________________________
lambda_3 (Lambda)            (None, 1)                 0           model_1[0][0]
                                                                   model_1[1][0]
__________________________________________________________________________________
dense (Dense)                (None, 1)                 2           lambda_3[0][0]
==================================================================================
Total params: 56,869,634
Trainable params: 2
Non-trainable params: 56,869,632
__________________________________________________________________________________
```

Figure 6: AlexNet – Summary for Control: initial pretrained network without fine-tuning

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================
input_4 (InputLayer)         [(None, 227, 227, 3)      0
__________________________________________________________________________________
input_5 (InputLayer)         [(None, 227, 227, 3)      0
__________________________________________________________________________________
model_4 (Functional)         (None, 4096)              56869632    input_4[0][0]
                                                                   input_5[0][0]
__________________________________________________________________________________
lambda_7 (Lambda)            (None, 1)                 0           model_4[0][0]
                                                                   model_4[1][0]
__________________________________________________________________________________
dense_1 (Dense)              (None, 1)                 2           lambda_7[0][0]
==================================================================================
Total params: 56,869,634
Trainable params: 16,781,314
Non-trainable params: 40,088,320
__________________________________________________________________________________
```

Figure 7: AlexNet – Summary for fine-tuned model: Training only the final fully connected layer

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

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

[3] G. B. Huang, V. Jain, and E. Learned-Miller, "Unsupervised joint alignment of complex images," in *ICCV*, 2007.

[4] "Siamese networks with keras, tensorflow, and deep learning," Apr 2021. [Online]. Available: https://www.pyimagesearch.com/2020/11/30/siamese-networks-with-keras-tensorflow-and-deep-learning/

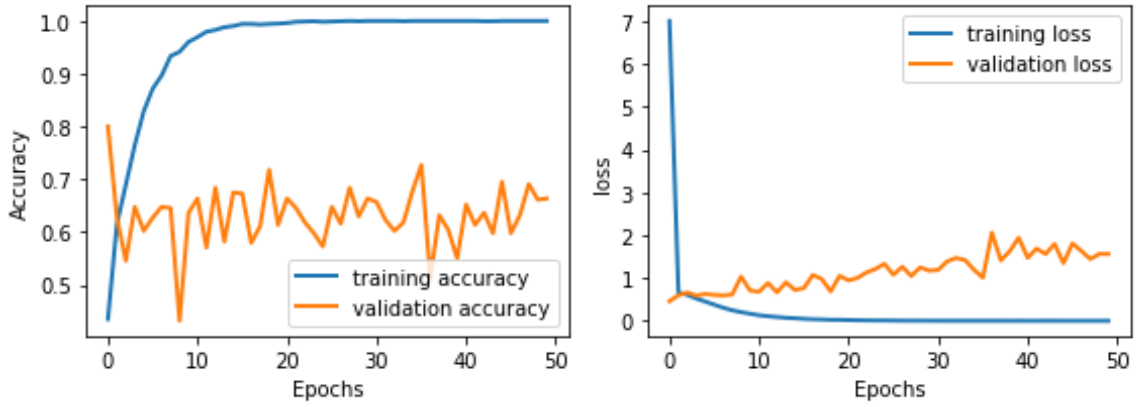

Figure 8: Training metrics: accuracy and loss plots

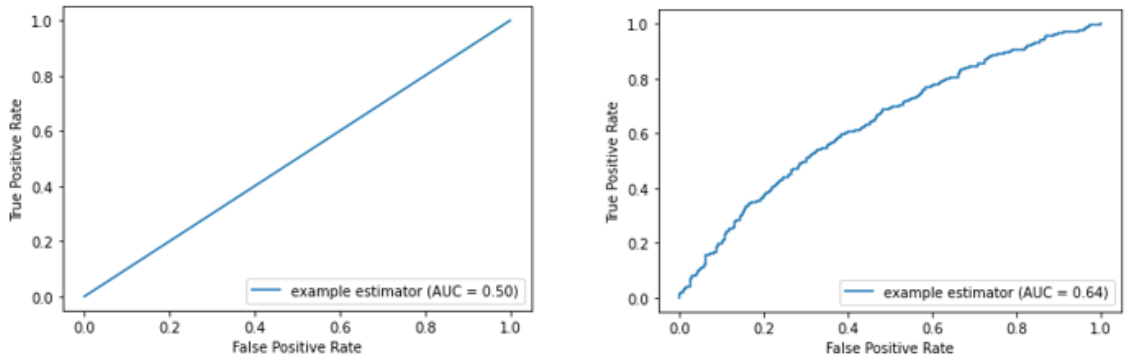

Figure 9: AlexNet – Outcomes of fine-tuning in comparison with initial pretrained model