Deep Learning Analysis of the Oxford-IIIT Pet Dataset

Viktor Zólyomi
(Dated: March 20, 2019)

Transfer learning is used to train deep learning networks for the purpose of classifying a database of ≈ 8000 images. Pre-trained networks implemented in Keras are used as base models, keeping their weights fixed while their top layers are replaced by 2 to 5 trainable fully connected layers.

Deep neural networks (DNNs) provide a powerful tool for image classification through the use of convolutional layers which identify features within images. The task of training a DNN for a specific dataset is made difficult by the computational expense of training convolutional networks of many layers, and by the need for large datasets. A way around this is to use the method of transfer learning [1]. Numerous DNNs are available publicly in the Keras library [2], which can be used as a starting point for transfer learning. One simply loads the pretrained DNN, discards its top layer which is specific to the classification task for which it was trained, then replaces it with a small number of fully connected layers designed for one's own image classification task at hand.

In this work, the transfer learning approach is used on a relatively modest dataset, the Oxford-IIIT Pet Dataset [3]. This dataset consists of roughly 8000 total images spanning 37 different classes of cat and dog breeds, i.e. on the order of 200 images per class. As such, training a DNN from scratch for classifying cat and dog breeds using this dataset would not be appropriate as sufficient accuracy requires a much bigger statistical ensemble. Instead, three pre-trained networks available in Keras are used: MobileNet [4], DenseNet [5], and Xception [6], comprising 88, 121, and 126 layers, respectively. In each case, the top layer is replaced with 2 to 5 fully connected layers to test the effects of increasing complexity at the top of the DNNs. Only the newly added layers are trained, all pre-trained weights within the base DNNs are retained. The Adam optimizer [7] and the categorical cross entropy loss function are used. 75% of the images are used for training and the rest for validation. Only the original images are used, ignoring the trimap and bounding box information supplied with the IIIT pet dataset, in order to gauge the effectiveness of the DNNs on the images before any manual preprocessing.

Training was performed by a Python program making use of the Keras 2.2.4 library and the TensorFlow 1.12 [8] backend, over 20 epochs. The source code used is hosted on GitHub [9] along with training history for the DNNs and png images visualizing the training metrics.

Fig. 1 shows the results of the training. Loss is shown on the left, accuracy on the right. Each row corresponds to the number of fully connected layers (2L, 3L, 4L, 5L) that were added to the base DNNs (MobileNet, DenseNet, Xception), as labelled.

Convergence of the training loss and accuracy is achieved rapidly. Past 10 epochs the training accuracy consistently remains greater than 95% in all cases except for the MobileNet-based DNN with 5 extra layers, and even in that case it is consistently above 90%. Validation accuracy slightly exceeds 80% uisng MobileNet and Xception with a small improvement using the latter. DenseNet-based DNNs underperform here with the validation accuracy fluctuating between 65% and 80%, except for the case of 5 fully connected layers at the top level, where fluctuations are even larger.

In terms of model complexity, Fig. 1 shows signs of overfitting when the number of fully connected layers at the top level increases. Validation loss rises when going from 4 to 5 layers in both MobileNet and DenseNet, while in the Xception-based networks the validation loss is best using just 2 fully connected layers at top level. As mentioned, there are fluctuations present in the validation accuracy, especially in the DenseNet-based DNNs. On the example of the MobileNet-based DNNs training was continued for an additional 30 epochs, i.e. a total of 50; these are stored in the GitHub repository [9]. The fluctuations observed in Fig. 1 remain even after 50 epochs with no significant change in the moving mean validation accuracy, suggesting that the DNNs used here reach peak performance after 10 epochs.

It is somewhat disappointing that the training accuracy does not reach even 90%, however, considering the small size of the training dataset this validation accuracy is not surprising. Much higher accuracy could likely be achieved by increasing the size of the training image dataset.

[1] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," arXiv:1808.01974.
[2] https://keras.io.
[3] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, in *IEEE Conference on Computer Vision and Pattern Recognition* (2012).
[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," arXiv:1704.04861.
[5] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," ArXiv:1608.06993.
[6] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," arXiv:1610.02357.

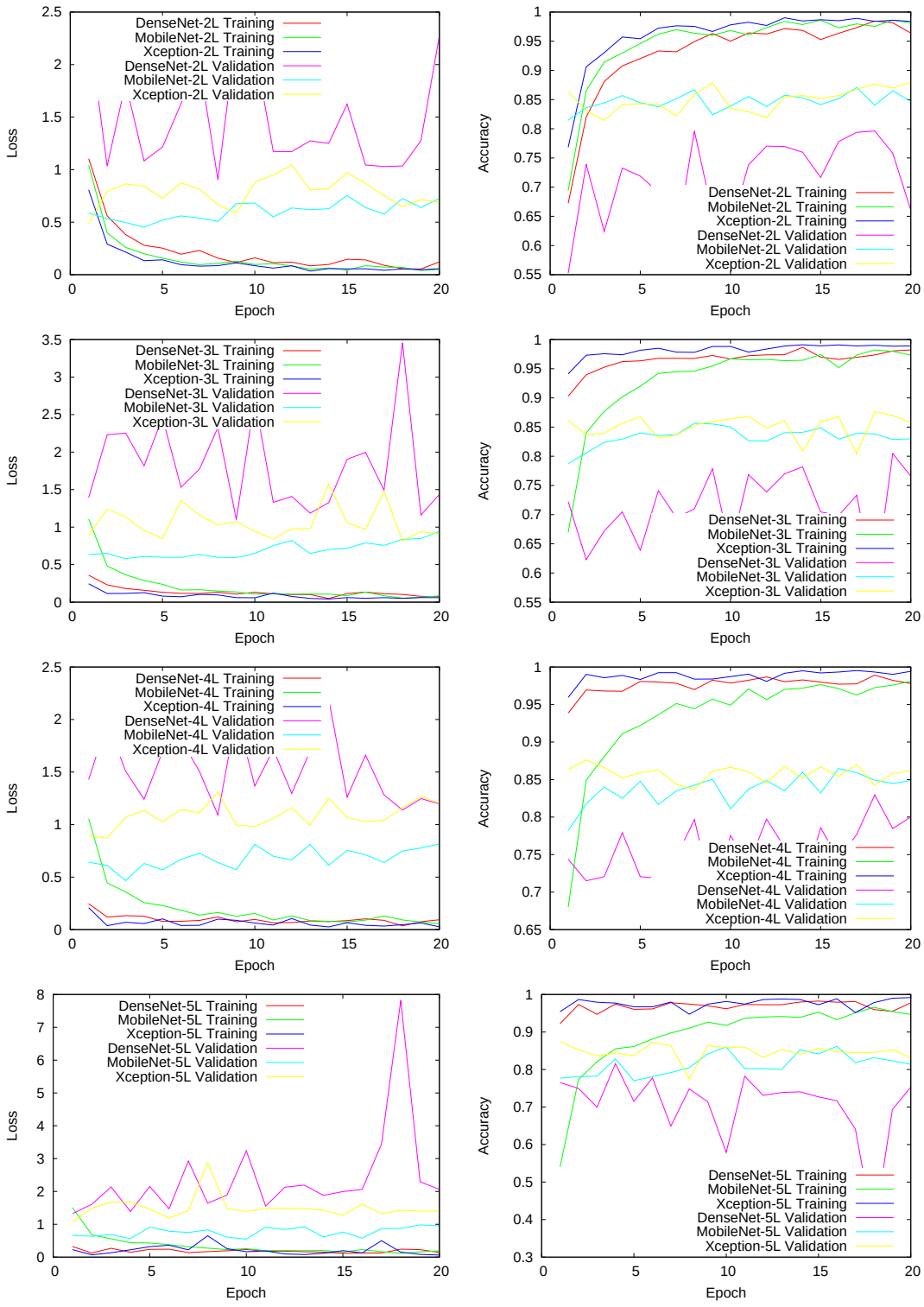

FIG. 1: Accuracy and loss function in the 12 deep neural networks trained in this work.

[7] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980.

[8] https://www.tensorflow.org.

[9] https://github.com/zachary469/TL-DNN.