Università
della
Svizzera
italiana

Faculty
of Informatics

January 16, 2018

Analysis on Transfer Learning

Report

Sumeet Gyanchandani, Hrishikesh Gupta

Abstract

In this paper, we analysis the performance of Transfer Learning for various datasets. The problem that we explore is of image classification based on retraining the deepest layer of a pre-trained Convolutional Neural Network. To evaluate the performance of Transfer Learning, we vary the amount of training data and use different datasets, viz., Animals with Attributes, Animals on the Web and NIH Chest X-ray. The results show a high accuracy for the first two data sets, while poor accuracy on the last one.

1 Introduction

Humans have inherent ways to transfer knowledge between tasks, i.e., we recognize and apply relevant knowledge from previous learning experiences when we encounter new tasks. The more related a new task is to our previous experience, the more easily we can master it.

In common machine learning algorithms, the assumption is that the future data used will be not only from same feature space but also their distribution will be same. But in practical scenarios, this is not the case. It could be that the domain in which we want to perform classification on, its data isn't enough [1]. But there might be an availability of the data from some other domain with different feature space. In such cases, knowledge transfer from data learned in one domain to another is key and would improve the learning performance. Transfer learning attempts to change this by developing methods to transfer knowledge learned in one or more source tasks and use it to improve learning in a related target task.

Through this project, we attempt to provide an analysis of the performance of Transfer Learning. But first, let us provide a short explanation about Transfer Learning and Google's Inception model.

1.1 Transfer Learning

Transfer learning makes use of the knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars can be used to some extent to recognize trucks.

In the classic supervised learning scenario of machine learning, if we intend to train a model for some task and domain A, we assume that we are provided with labeled data for the same task and domain [2].

We can now train a model A on this dataset and expect it to perform well on unseen data of the same task and domain. On another occasion, when given data for some other task or domain B, we require again labeled data of the same task or domain that we can use to train a new model B so that we can expect it to perform well on this data.

The traditional supervised learning paradigm breaks down when we do not have sufficient labeled data for the task or domain we care about to train a reliable model.

Transfer learning allows us to deal with these scenarios by leveraging the already existing labeled data of some related task or domain. We try to store this knowledge gained in solving the source task in the source domain and apply it to our problem of interest as can be seen in Figure 1.

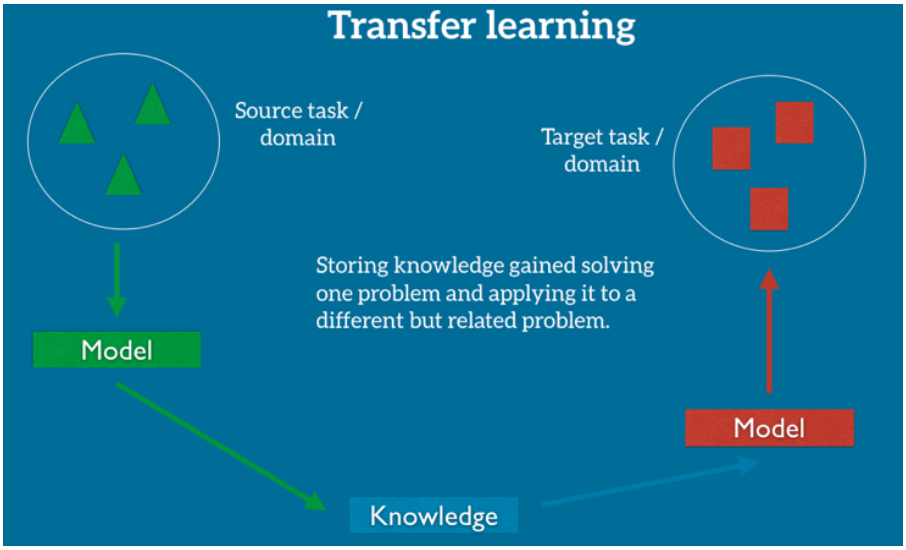


Figure 1. The Transfer Learning Setup (source[2])

There are various methods of transfer learning which have been implemented. In an inductive learning task or inductive transfer learning, the objective is to induce a predictive model from a set of training examples. The predictive model learned by an inductive learning algorithm and should make accurate predictions not just on the training examples, but also on future examples that come from the different distribution of another dataset. Another type of inductive learning involves modeling probability distributions over interrelated variables [3]. Examples of these systems are Bayesian networks and Markov Logic Networks [4].

1.2 Google's Inception v3

The "Inception" micro-architecture was first introduced by Szegedy et al. in their 2014 paper [5]. This is visualized as below:

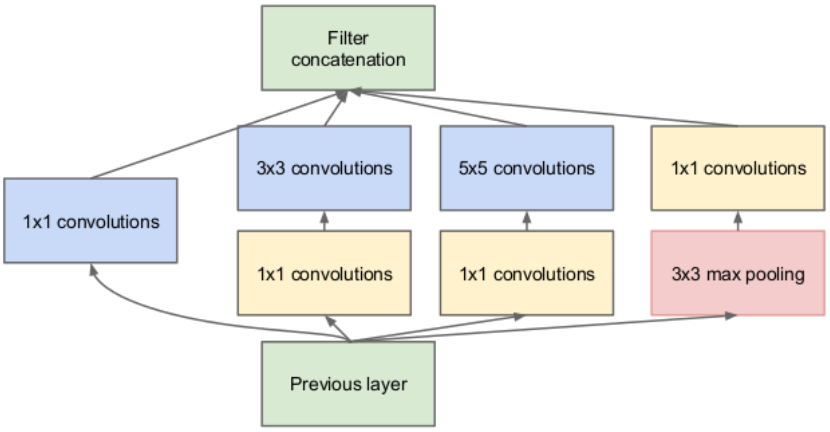


Figure 2. The Inception Module (source[5])

What is an inception module? The goal of the inception module is to act as a "multi-level feature extractor" by computing 1×1, 3×3, and 5×5 convolutions within the same module of the network. The output of these filters is then stacked along the channel dimension before being fed into the next layer in the network.

The inspiration comes from the idea to do away with the need to make a decision as to what type of convolution to make at each layer: 3×3 or 5×5?. The idea is to use all of them and let the model decide. This is done by doing each convolution in parallel and concatenating the resulting feature maps before going to the next layer. Assume that the next layer is also an Inception module, then each of the convolution's feature maps will be passed through the mixture of convolutions of this layer. The idea is to do away with the need to know ahead of time if it was better to do, a 3×3 and then a 5×5. Instead, just do all the convolutions and let the model pick what's best.

The Architecture proposed in the paper "Rethinking the inception Architecture for Computer Vision" describes significant changes over the previous model "GoogleNet"[5]. In this section, we aim at explaining the architecture of Inception model V3 by listing out these changes.

The traditional 7×7 convolution has been factorized to 3×3 Convolution. This was to reduce the Computation complexity of the network, and was possible as follows, considering a computation graph of the 5×5 convolution, wherein each output looks like a small fully-connected network sliding over 5×5 tiles over its input (Fig.1). By exploiting the translation invariance and replacing the fully connected component by a two layer convolutional architecture: the first layer being a 3×3 convolution and second a fully connected layer on top of the 3×3 output grid of the first layer [6]. Figure 3. shows an example for factorization of 5×5 convolution.

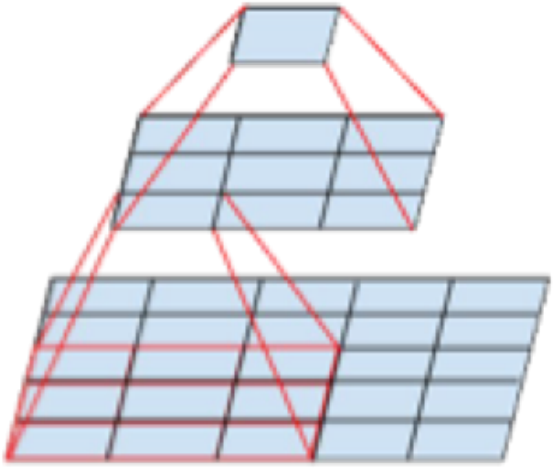


Figure 3. The 5×5 convolution (source[5])

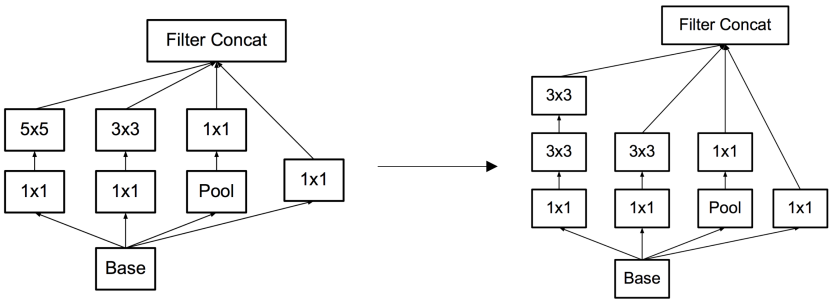


Figure 4. Factorization into smaller convolution (source[5])

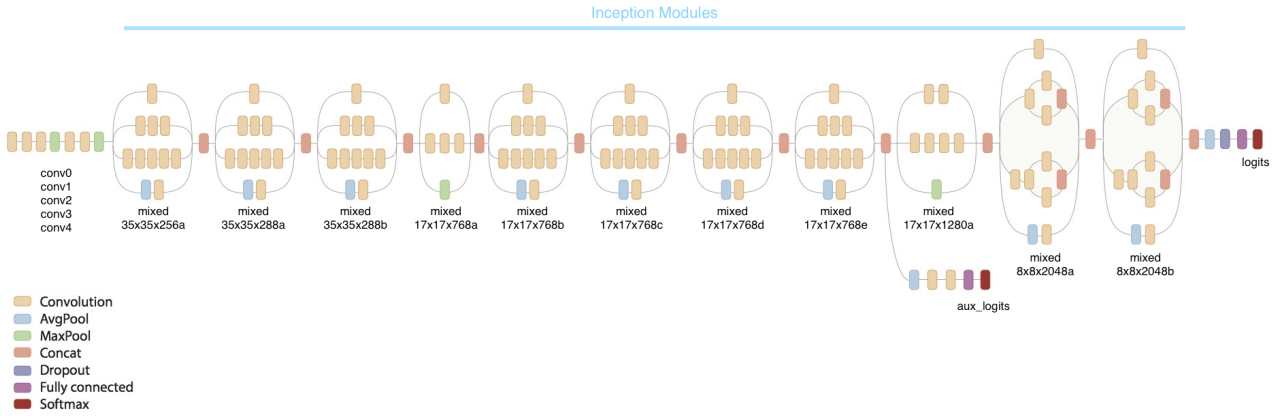


Figure 5. The Inception Architecture

For the Inception part of the network, there are 3 traditional inception modules at the 35×35 with 288 filters each. which is reduced to a 17×17 grid with 768 filters using the grid reduction technique [6]. Wherein the representational bottleneck is removed. This is followed by 5 instances of the factorized inception modules, which is reduced to a 8×8×1280 grid with the grid reduction technique [6]. At the coarsest 8×8 level, there are two Inception modules, with a concatenated output filter bank size of 2048 for each tile. To above, BN-auxiliary is added for inception V3 [6], which means the fully connected layer of the auxiliary classifier is also batch-normalized, apart from convolutions.

2 Method

The following is a very short description of the method that we are using for retraining the Google Inception v3:

2.1 Preprocessing

The standard input dimensions for Google Inception v3 are 299 × 299 and it recognizes only JPEG compressions. We to convert every image to JPEG and resize to the above dimension.

2.2 Creating bottleneck files

Next step after image conversion, was calculating the bottleneck values of these images. We did this just once in the beginning for all the images in datasets and saved them for future use.

2.3 Create Graph object

We first download the model that we want to retrain, in our case Google Inception v3 and then parse it's GraphDef file into a buffer, that is then used to import the GraphDef as the graph for the current TensorFlow session.

2.4 Retraining the last layer

We create a new softmax and fully-connected layer and train only this layer for the new classes that we want to classify. Once, the training is completed, we save the trained graph as a GraphDef file to use for prediction later.

3 Transfer Learning on Animals on the Web dataset

The **Animals on the Web** [7] was the first dataset that we used for analyzing the accuracy that Transfer Learning achieves. The idea of using this dataset was taken by Aniruddha Tapas's re-implementation [8] of the TensorFlow retraining example [9].

We retrained the last layer of the Google's Inception Model v3 with 300 images per class for 500 training steps. Figure 6 to Figure 9 show the results of classification of the unseen images on our retrained Model.

```
leopard (score: 0.848937 )
giraffe (score: 0.0609134 )
monkey (score: 0.0169987 )
frog (score: 0.0148349 )
alligator (score: 0.014431 )
ant (score: 0.014057 )
dolphin (score: 0.0110906 )
penguin (score: 0.0095556 )
beaver (score: 0.0091819 )
```

Figure 6. Classification of an Leopard image

```
giraffe (score: 0.838444 )
leopard (score: 0.0963956 )
alligator (score: 0.0145426 )
beaver (score: 0.0098485 )
ant (score: 0.00973451 )
dolphin (score: 0.0091504 )
monkey (score: 0.00821851 )
penguin (score: 0.00685848 )
frog (score: 0.00680741 )
```

Figure 7. Classification of an Giraffe image

```
alligator (score: 0.815673 )
leopard (score: 0.0466529 )
monkey (score: 0.022044 )
frog (score: 0.0219493 )
ant (score: 0.0215884 )
beaver (score: 0.0206687 )
giraffe (score: 0.0205339 )
dolphin (score: 0.0182061 )
penguin (score: 0.0126839 )
```

Figure 8. Classification of an Alligator image

```
leopard (score: 0.139981 )
alligator (score: 0.129963 )
giraffe (score: 0.122523 )
frog (score: 0.12094 )
monkey (score: 0.117933 )
ant (score: 0.104893 )
penguin (score: 0.0997984 )
beaver (score: 0.0858829 )
dolphin (score: 0.078085 )
```

Figure 9. Classification of an Panda image

Figure 10 shows the mean Cross Entropy for the retraining and Figure 11 shows the Accuracy for the retraining. In both the figures Training data is shown in orange and Validation data is shown in blue.

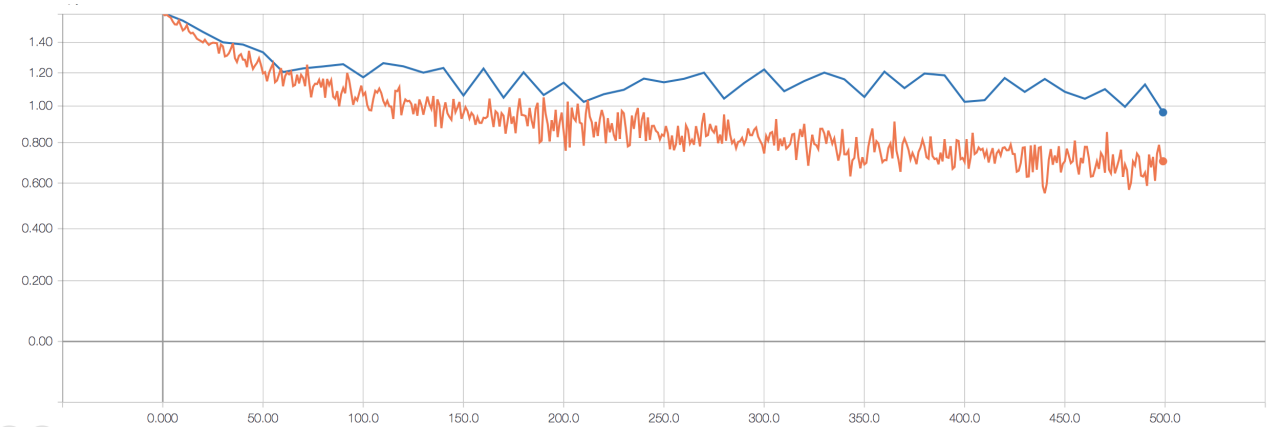


Figure 10. The mean cross entropy for Transfer Learning on Animals on the web dataset

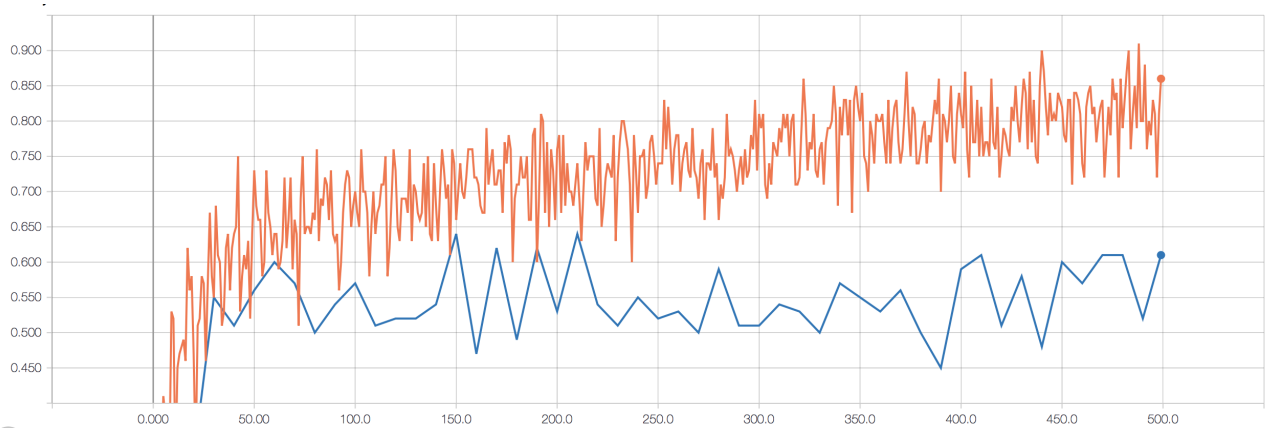


Figure 11. The accuracy for Transfer Learning on Animals on the web dataset

We also did an analysis to test the accuracy that the retraining yielded, if we varied the training data. We tried a different number of images per class and increased the number of classes during this analysis. Figure 12 shows the result of the analysis.

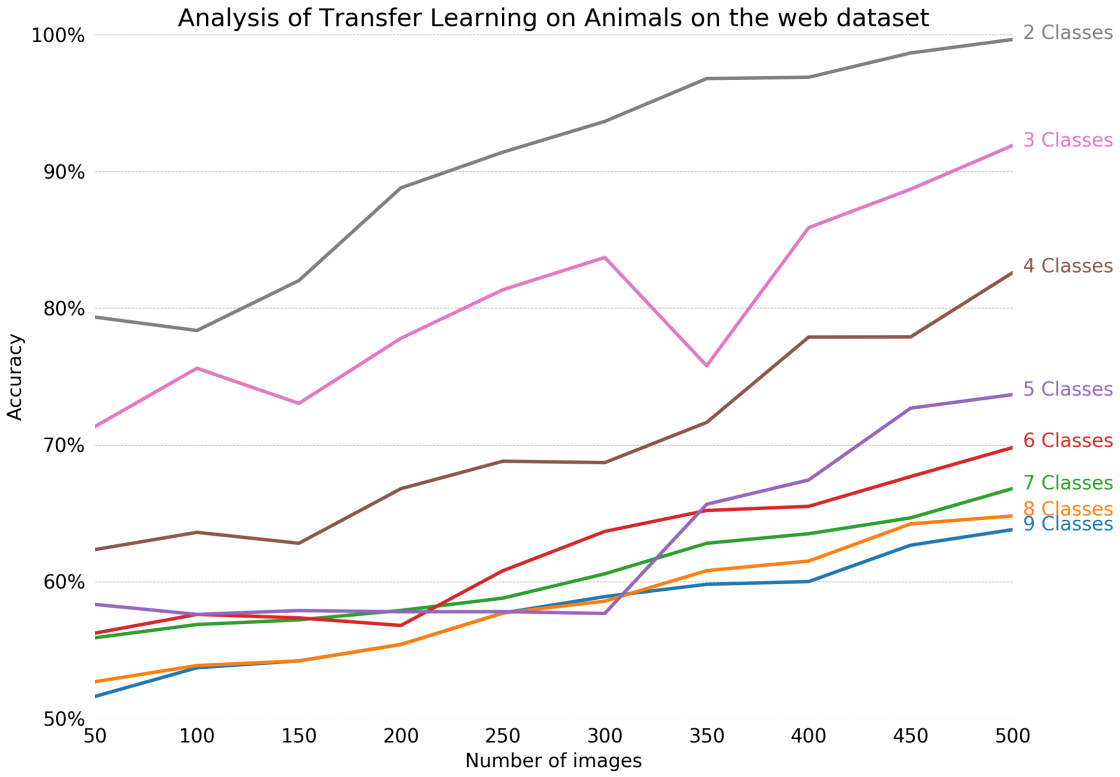


Figure 12. Analysis of training with varied amount of data on Animals on the web dataset

4 Transfer Learning on Animals with Attributes 2 dataset

The **Animals with Attribute 2** [10] was the second dataset that we used and the idea of using this was our own.

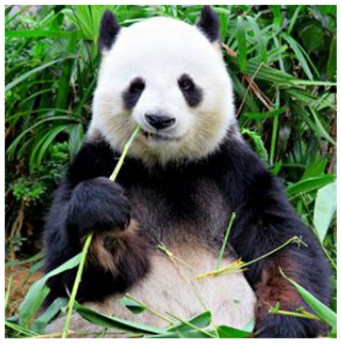
We retrained the last layer of the Google's Inception Model v3 with 300 images per class for 500 training steps. Figure 13 to Figure 16 show the results of classification of the unseen images on our retrained Model.

```
Classification:
lion (97.21%)
ox (0.28%)
grizzly bear (0.27%)
tiger (0.27%)
rhinoceros (0.25%)
squirrel (0.21%)
gorilla (0.18%)
zebra (0.18%)
bat (0.17%)
hamster (0.17%)
sheep (0.17%)
rabbit (0.15%)
polar bear (0.14%)
giant panda (0.13%)
fox (0.11%)
elephant (0.10%)
```

Figure 13. Classification of an Lion image

```
Classification:
tiger (95.23%)
zebra (1.32%)
hamster (0.33%)
lion (0.32%)
fox (0.31%)
grizzly bear (0.28%)
polar bear (0.26%)
giant panda (0.26%)
squirrel (0.25%)
bat (0.24%)
rabbit (0.23%)
ox (0.21%)
elephant (0.21%)
rhinoceros (0.20%)
gorilla (0.18%)
sheep (0.17%)
```

Figure 14. Classification of an Tiger image



```
Classification:
giant panda (95.01%)
grizzly bear (0.71%)
polar bear (0.68%)
fox (0.40%)
hamster (0.37%)
bat (0.36%)
gorilla (0.32%)
tiger (0.30%)
ox (0.29%)
sheep (0.28%)
rabbit (0.24%)
lion (0.23%)
rhinoceros (0.21%)
squirrel (0.21%)
elephant (0.20%)
zebra (0.20%)
```

Figure 15. Classification of an Panda image

```
Classification:
polar bear (94.88%)
grizzly bear (1.16%)
giant panda (0.47%)
sheep (0.41%)
hamster (0.37%)
fox (0.34%)
tiger (0.34%)
lion (0.34%)
elephant (0.32%)
gorilla (0.28%)
squirrel (0.21%)
bat (0.20%)
ox (0.19%)
rabbit (0.18%)
rhinoceros (0.17%)
zebra (0.14%)
```

Figure 16. Classification of an Polar Bear image

Figure 17 shows the mean Cross Entropy for the retraining and Figure 18 shows the Accuracy for the retraining. In both the figures Training data is shown in orange and Validation data is shown in blue.

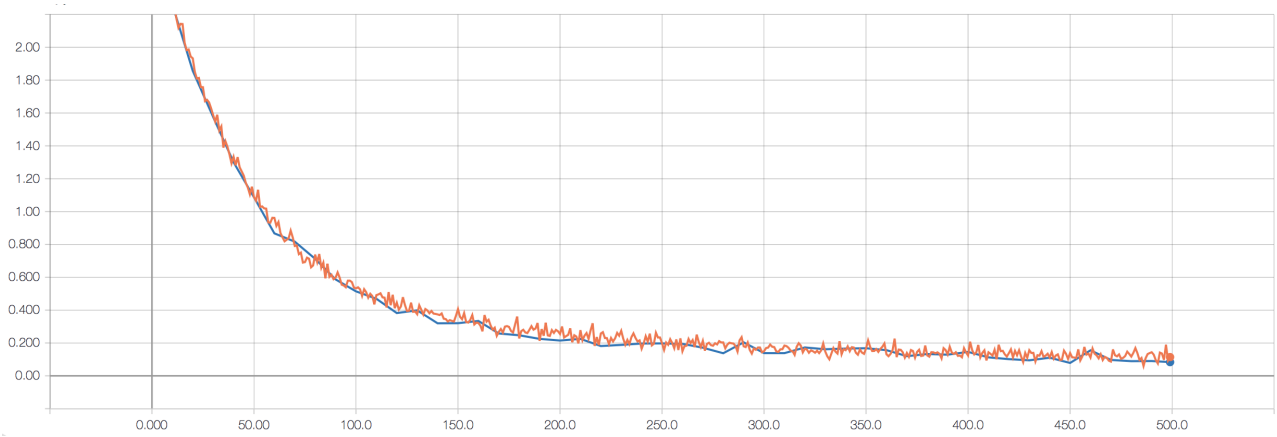


Figure 17. The mean cross entropy for Transfer Learning on Animals with Attributes 2 dataset

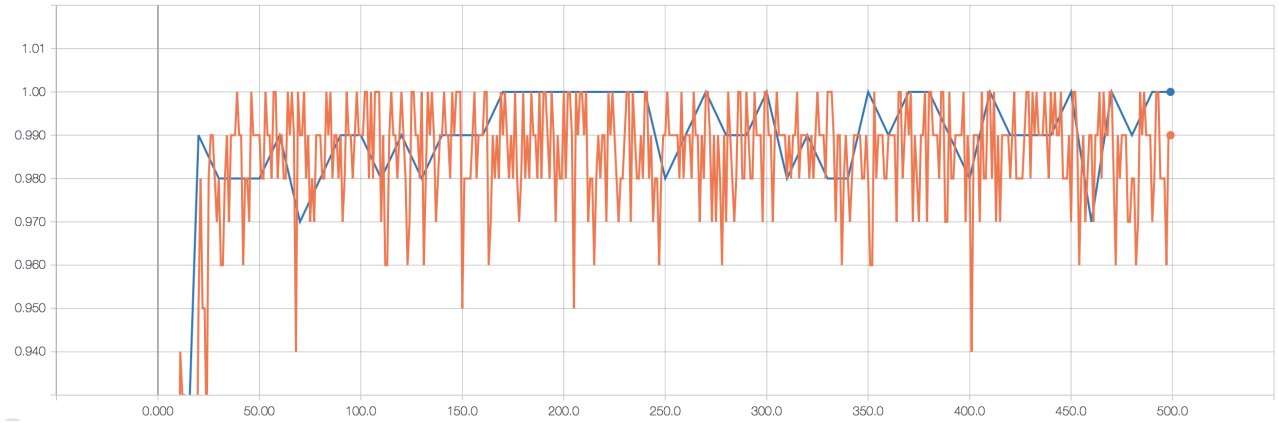


Figure 18. The accuracy for Transfer Learning on Animals with Attributes 2 dataset

We also did an analysis to test the accuracy that the retraining yielded, if we varied the training data. We tried a different number of images per class and increased the number of classes during this analysis. Figure 19 shows the result of the analysis.

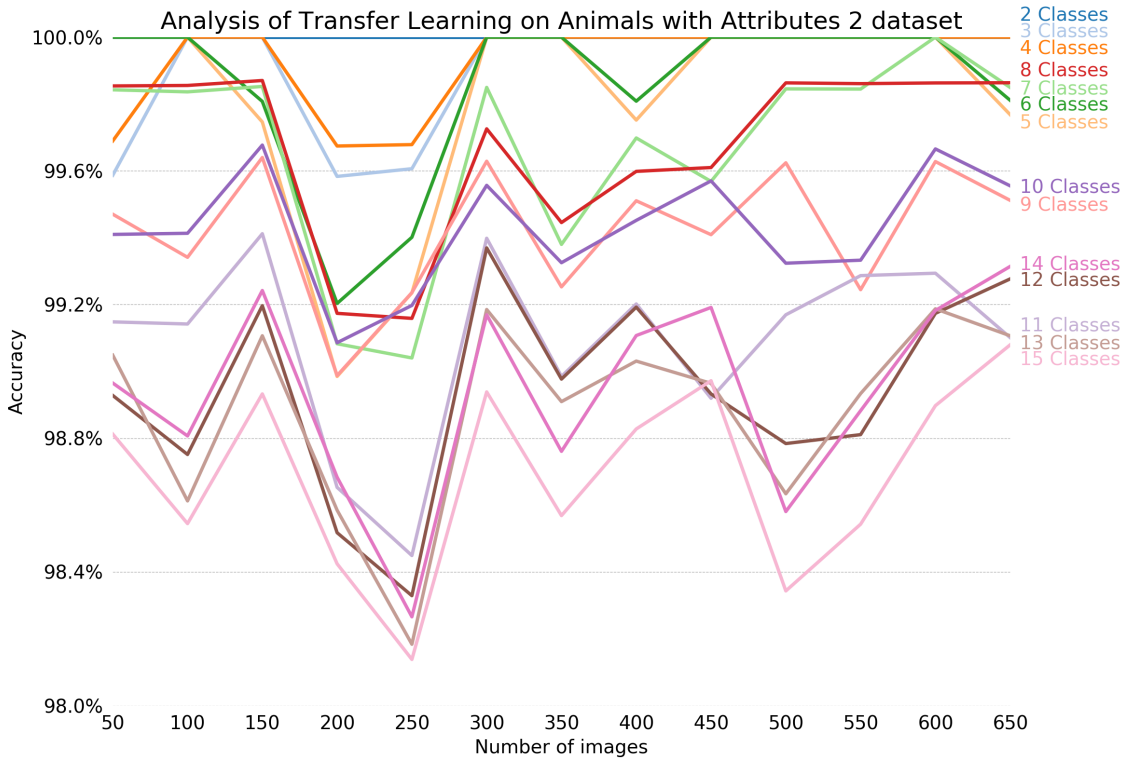


Figure 19. Analysis of training with varied amount of data on Animals with Attributes 2 dataset

5 Transfer Learning on NIH Chest X-ray dataset

The images of animals belong to similar data distribution, as that of ImageNet. We wanted to see how well Google's Inception v3 performs if we retrain the last layer with medical x-ray images.

That led to us analyzing a third dataset, viz. **NIH Chest X-ray dataset** [11]. Figure 20 and Figure 21 show correct classification by the trained model (though not very confidence), while Figure 22 and Figure 23 show misclassification by the trained model.

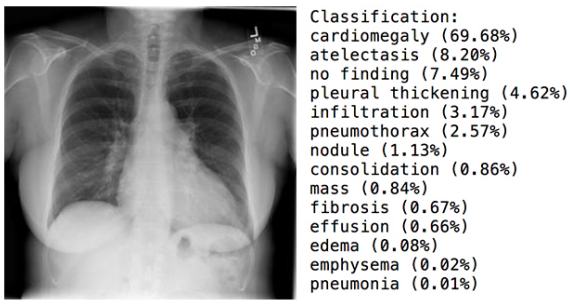


Figure 20. The results of classification of unseen chest x-ray labeled Cardiomegaly

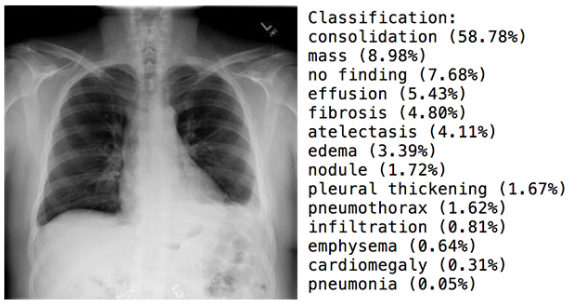


Figure 21. The results of classification of unseen chest x-ray labeled Consolidation

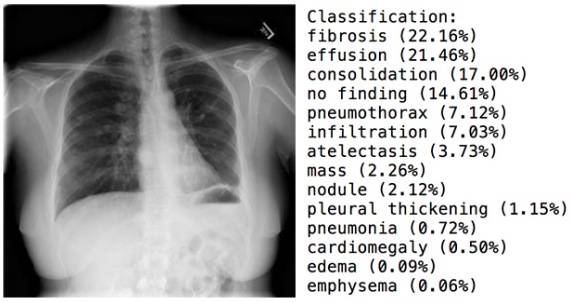


Figure 22. The results of classification of unseen chest x-ray labeled Nodule

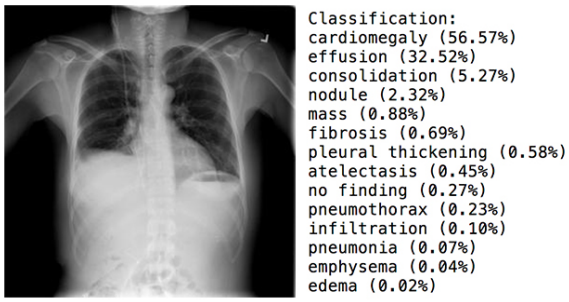


Figure 23. The results of classification of unseen chest x-ray labeled No Finding

Figure 24 shows the mean Cross Entropy for the retraining and Figure 25 shows the Accuracy for the retraining. In both the figures Training data is shown in orange and Validation data is shown in blue.

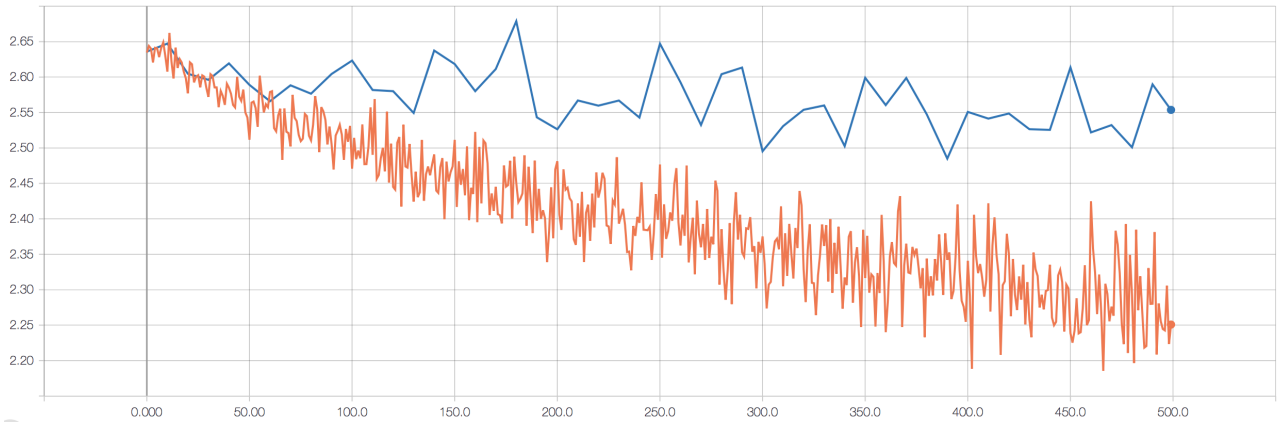


Figure 24. The mean cross entropy for Transfer Learning on NIH Chest XRay dataset

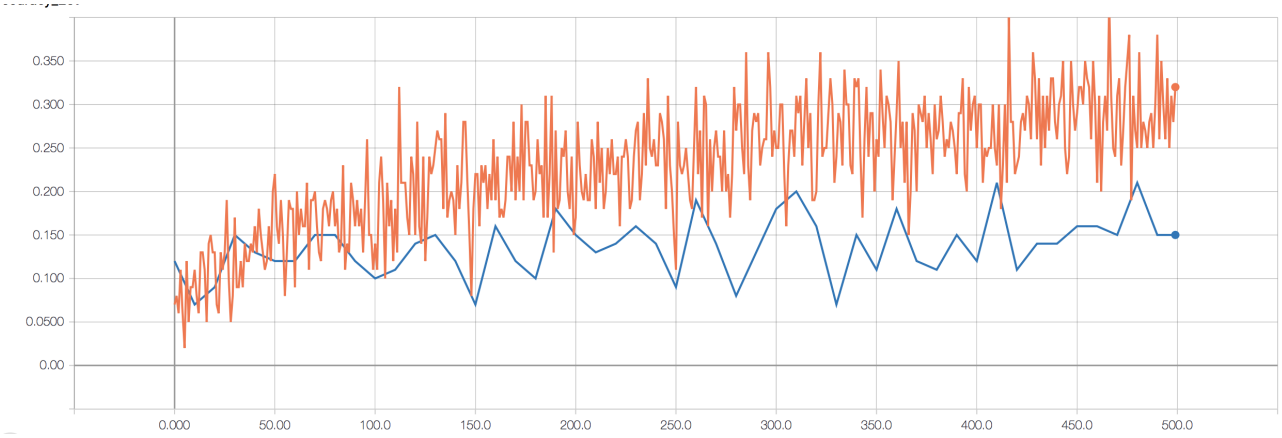


Figure 25. The accuracy for Transfer Learning on NIH Chest XRay dataset

We also did an analysis to test the accuracy that the retraining yielded, if we varied the training data. We tried a different number of images per class and increased the number of classes during this analysis. Figure 26 shows the result of the analysis.

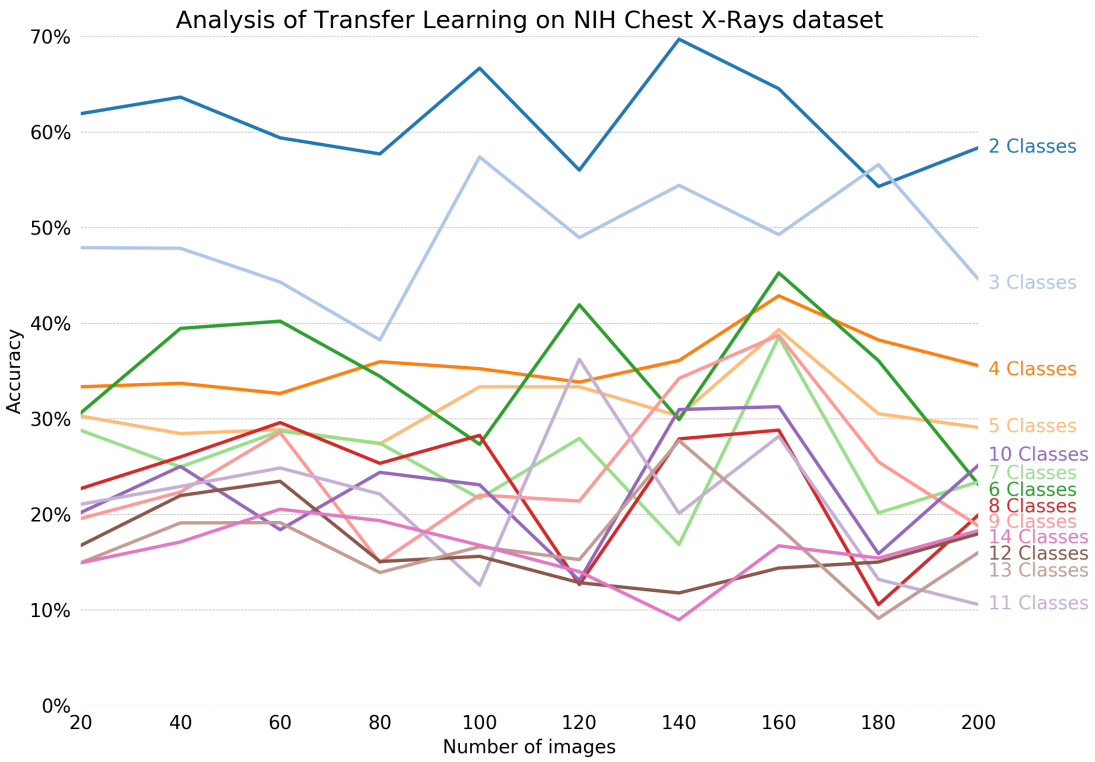


Figure 26. Analysis of training with varied amount of data on NIH Chest X-ray dataset

6 Conclusions

For the first dataset, **Animals on the Web**, the test accuracy was between 50% to 100%. By looking at Figure 12 we can conclude that:

1. If we increase the images data per class, the accuracy increases
2. If we increase the number of classes in the model, the accuracy decreases

For the second dataset, **Animals with Attributes 2**, the test accuracy was very high, between 98% to 100%. The plots of the number of classes intersect more than the previous dataset. From this analysis, we can conclude that:

1. If we increase the images data per class, the accuracy increases
2. If we increase the number of classes in the model, the accuracy decreases
3. We notice more fluctuations here as the range of y-axis is between 98.0% to 100%. Hence, these fluctuations are very small and do not mean any change to the above two conclusions
4. We got relative higher accuracy here when compared to the first dataset. We mostly attribute this to Animals with Attribute 2 being much cleaner than the Animals on the Web dataset. By less clean we mean the Animal on the Web dataset had images like these:

Figure 27. Example of images in the Beaver class of the Animals on the Web dataset

For the third dataset, **NIH Chest X-ray** images, the test accuracy was between 10% to 70% for the similar amount of data and same number of training steps as the previous two datasets. The only conclusion that we draw here is Transfer Learning did not perform as well as the above two datasets because xray images belong to a different data distribution than the images in ImageNet.

The overall conclusion of our project is that Transfer Learning is very successful and less time consuming technique, at least for similar data distribution. We don't claim that the xray images were successful in breaking the Transfer Learning technique, but the accuracy was too low to be acceptable.

References

[1] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, 2009.

[2] S. Ruder, "Transfer learning - machine learning's next frontier." http://ruder.io/transfer-learning/index.html#relatedresearchareas.

[3] L. Torrey and J. Shavlik, "Transfer Learning," 2009.

[4] M. Richardson and P. Domingos, "Markov logic networks, Machine Learning," 2006.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[6] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[7] T. L. Berg and D. A. Forsyth, "Animals on the Web," *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[8] A. Tapas, "Classifying giraffe and leopard images." https://github.com/Aniruddha-Tapas/Transfer-Learning-for-Animal-Classification-in-Tensorflow/blob/master/Classifying-Giraffe-And-Leopard-Images.ipynb.

[9] TensorFlow, "Tensorflow example on retraining." https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/image_retraining.

[10] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly," *Computing Research Repository (CoRR)*, vol. arXiv:1707.00600, 2017.

[11] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases," *IEEE CVPR*, 2017.