Best Practices for Fine-tuning Visual Classifiers to New Domains

Brian Chu*, Vashisht Madhavan*, Oscar Beijbom, Judy Hoffman, Trevor Darrell

University of California, Berkeley. {brian.c@,vashisht.madhavan@, obeijbom@eecs.,jhoffman@eecs.,trevor@eecs.}berkeley.edu

Abstract. Recent studies have shown that features from deep convolutional neural networks learned using large labeled datasets, like ImageNet, provide effective representations for a variety of visual recognition tasks. They achieve strong performance as generic features and are even more effective when fine-tuned to target datasets. However, details of the fine-tuning procedure across datasets and with different amount of labeled data are not well-studied and choosing the best fine-tuning method is often left to trial and error. In this work we systematically explore the design-space for fine-tuning and give recommendations based on two key characteristics of the target dataset: visual distance from source dataset and the amount of available training data. Through a comprehensive experimental analysis, we conclude, with a few exceptions, that it is best to copy as many layers of a pre-trained network as possible, and then adjust the level of fine-tuning based on the visual distance from source.

1 Introduction

One of the key factors which contributes to the impact of convolutional neural networks (CNNs) is the transferability of their internal deep representations for a variety of visual recognition tasks. Such deep representations, or 'features,' are empirically superior [1, 2] to traditional features when classifying datasets such as Caltech-101, Caltech-256, and SUN397 [3–5]. Recently, methods of fine-tuning pre-trained networks towards new target datasets have become very popular, as they are usually more effective than training deep networks from scratch. With fine-tuning, the first n layers from a pre-trained network are copied to the target network, while other layers are randomly initialized and trained towards the target dataset. One can either choose to adapt, or fine-tune, the copied layers or leave them unchanged (frozen) during training [6, 7]. Past work has even shown the superiority of fine-tuning over using generic CNN features for different visual tasks like detection [8, 7, 9].

When faced with a new dataset, however, there is little guidance on how many layers to copy and whether to fine-tune or freeze these layers. We argue that the best practice may vary depending upon a few key factors of the transfer setup.

* These authors contributed equally to this work.

As a result, we choose to analyze the performance of fine-tuning methods across seven target datasets, comparing these methods among two intrinsic properties of the datasets: difference from source dataset and the amount of available target training data. Through our analysis, we uncover the best training methods in each scenario and use the results to provide two main recommendations:

1. **Copy all layers except the classification layer**. This is often standard practice, though we are the first to provide comprehensive evidence across a variety of datasets and many different operating points of the amount of labeled data available in the target dataset.
2. **Fine-tune the copied layers.** We find that even with very few examples, fine-tuning is possible and beneficial. The exception being if the dataset distance is small and there is only a small amount of training data. In this case, freeze the copied layers.

2 Related Work

Although networks have been shown to increase in class and representation specificity from lower to higher layers [8, 6], for many datasets the best performance of generic AlexNet [10] features occurs at the third-to-last (fc6) [2], or second-to-last (fc7) fully-connected layer [1]. Moreover, Girshick et al.'s ablation studies found it is best to copy all layers from a network pre-trained on ImageNet and fine-tune these layers towards the PASCAL VOC detection task [7].

Yosinski et al. [6] varied the number of pre-trained layers copied and examined the target dataset accuracy for networks that were fine-tuned and frozen (unchanged). With the source dataset as one half of ImageNet's classes and the target dataset as the remaining half, Yosinski et al. found that fine-tuning was the optimal technique, with performance slightly improving as more layers were copied. When layers were frozen, they saw performance degrade as more layers were copied. However, Yosinski et al.'s work studied a target dataset that was virtually identical to the source dataset and had an extremely large number of samples (approximately 645,000). This directly motivates our broader analysis of target datasets of varying distances to the source dataset, and datasets with scarce and plentiful training data.

The need to characterize distance between source and target is further motivated by Azizpour et al. and Zhou et al., who both demonstrated substantial variation in the performance of generic features depending on the source dataset and qualitative characteristics of the target dataset [11, 9]. Zhou et al. specifically demonstrated that a CNN pre-trained on a scene dataset is superior to an ImageNet model when fine-tuned towards other scene datasets.

3 Experiments

We follow the experimental setup used by Yosinski et al. [6]. Specifically we use the 8-layer Caffe implementation of AlexNet (CaffeNet), pre-trained on the entire ImageNet training set [12].

We evaluate the performance of fine-tuning and freezing when adapting the pre-trained model to 6 target datasets. For each dataset, we define 3-4 dataset splits with a varying number of images per class, for a total of 23 dataset splits. We also define fixed validation and test sets which, due to different amounts of training data, differ in size between datasets. The splits are outlined in Table 1.

For each dataset split, we randomly initialize the top 1, 3, or 5 layers in the pre-trained model while copying the rest of the layers. Additionally, we either freeze the copied layers, setting the learning rate to 0, or fine-tune them, setting the learning rate to 0.2 times that of random initialization. We follow this procedure for all target datasets, resulting in 138 experiments. For notation, *T(a-b)* denotes that layers a-b are copied and fine-tuned, whereas *F(a-b)* denotes that layers a-b are copied and frozen. *R(a-b)* denotes that layers a-b are randomly initialized. For example, *T(1-7)R(8)* denotes the experiment where we copy and fine-tune layers 1-7 and randomly initialize the final fully-connected layer, fc8.

Table 1. Properties of datasets and dataset splits. $*$ indicates we used the dataset's provided validation or test set as our test set.

Dataset	# Categories	Classification Task	# Images per Class		
			Val	Test	Train
Caltech256	256	Object	2	25	1, 10, 25, 53
SUN397	397	Scene	2	25	1, 10, 50, 70
MITIndoor	67	Scene	2	$*$	1, 10, 25, 75
CUB-200	200	Object (fine-grained)	2	$*$	5, 20, 35
Coral	9	Coral	50	300	10, 50, 200, 450
Plankton	103	Plankton	50	85	1, 10, 300, 550
Yosinski	500	Object	20	$*$	1, 10, 25, 53, 120

We evaluated our experiments on 6 datasets: Caltech256, SUN397, MIT Indoor Scene Recognition (MITIndoor) [13], Caltech-UCSD-Birds-200 (CUB) [14], Moorea Labeled Corals (Coral) [15], and Imaging Flow Cytobot Data Plankton (Plankton) [16]. Properties of these datasets are summarized in Table 1. We only used 8 of the Coral categories and 34 of the Plankton categories in our experiments due to insufficient training data per class.

In addition to these 138 experiments, we chose to include an additional artificial dataset from Yosinksi et al. [6], denoted as Yosinski in Table 1. In their experiments, ImageNet is randomly split into two disjoint 500-category datasets (dataset 500A and dataset 500B) of roughly equal size (approximately 645,000 images). CaffeNet is pre-trained on dataset 500A, with the target as dataset 500B. We use this setup to analyze the case when datasets are essentially identical and also compare our results to theirs. We use the same experiment setups as before for an additional 40 experiments.

3.1 Difference from Source

To measure the difference between the source and target datasets, we compute the cosine distance, $1-\frac{\mu_s^T \mu_t}{||\mu_s||_2||\mu_t||_2}$, between the mean fc7 responses of the source, μ_s, and target, μ_t, datasets. Although not a formal distance measure due to its violation of the triangle inequality, cosine distance effectively measures the similarity of two vectors, which in our case measures the similarity of two datasets. In Table 2, we compare the cosine distance to other metrics such as MMD between source and target dataset in fc7 feature space which, when using a linear kernel, is equivalent to the Euclidean distance between the means of the fc7 responses. We also consider the accuracy of two classifiers trained to distinguish between datasets: a linear SVM in fc7 feature space and a small CNN model used by Krizhevsky for CIFAR-10 classification in pixel space [17]. This approach was recently used to minimize domain difference for adaptation [18–20]. When generating fc7 responses, we use CaffeNet pre-trained on the source dataset: this source dataset is ImageNet for all datasets setups except Yosinski, where source is 500A.

Table 2. Distance between source and target dataset. For Yosinski, source is dataset 500A and target is dataset 500B. For others, source is ImageNet and target is listed.

Dataset	Cosine distance	MMD	Linear SVM	CNN
Yosinski	0.003	2.3	57.3%	51.0%
Caltech-256	0.071	10.6	71.4%	69.0%
SUN397	0.194	17.8	81.5%	76.4%
MIT-Indoor	0.307	23.9	90.0%	84.5%
CUB-200	0.358	37.2	92.9%	86.5%
Coral	0.455	38.7	97.3%	99.4%
Plankton	0.534	39.1	97.2%	99.7%

Although these metrics differ in the distances between entries, they yield the same ordering. This suggests that the other metrics are viable substitutes for cosine distance, yet we settle on cosine distance because its computation does not require training a classifier and because it is bounded between 0 and 1.

4 Results

In this analysis we will refer to a *low* source-target distance as a cosine distance of between 0-0.2, a *medium* distance as 0.2-0.4, and a *high* distance as 0.4-1. We will refer to a *low* amount of target data as 1-20 images per class, a *medium* amount as 21-99 images per class, and a *large* amount as 100 or more images per class. All raw experimental results can be found in Fig. 3 in the Appendix.

We begin by studying whether random initialization or initializing with copied parameters yields higher performance across a variety of dataset shifts.

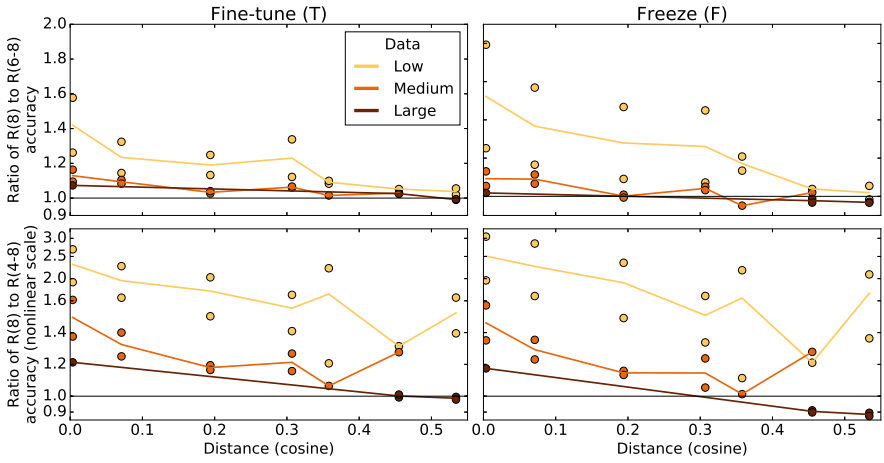

Fig. 1. Top: Ratio between accuracy for *R(8)* and accuracy for *R(6-8)*. **Bottom:** Ratio between *R(8)* and *R(4-8)*. Each point represents a pair of experiments with other conditions fixed. At **Left** are fine-tune (T) experiments only, at **Right** are freeze (F) experiments only. Values above 1.0 imply copying is better than random initialization. Values below 1.0 imply copying is worse. Trend lines are averages.

To do this, we hold all parameters of a particular experiment fixed except for whether certain layers are randomly initialized or copied. The results across all experiments are shown in Fig. 1, where the ratio of performance between copying and randomly initializing layers is indicated. Here, we found that randomly initializing layers beyond the necessary fc8 layer almost always degrades performance (all numbers across experiments are > 1). The notable exception being when source and target datasets have high difference and there are a large number of labeled target examples available for fine-tuning. In this setting randomly initializing offers a stronger benefit when the lowest layers are frozen during the final training step, but marginal or no improvement over copying when all layers are fine-tuned. Therefore, we conclude that copying all but the last layer of the network is generally the best practice for fine-tuning to a new dataset.

Our finding of copying all but the last layer is in direct contrast with Yosinski et al., who showed that copying fewer layers is better when freezing [6]. The contrasting results are easily explained by the much larger amounts of data used in that study, whereas we seek to analyze scenarios in which large amounts of data are not available: the largest amount of data we use for the Yosinski target dataset is 120 images/class (60,000 images). Clearly, there is an inflection point between 60,000 and approximately 645,000 training examples where it no longer becomes beneficial to copy more layers when freezing. For the remainder of our analysis, we follow our first finding to copy all parameters from the initial source network and therefore choose between *F(1-7)R(8)* and *T(1-7)R(8)*. We present our recommendations in Table 3 and discuss evidence from our experiments next.

Table 3. Best practices for adapting pre-trained networks where all but the last layer are copied. We compare freezing (*F(1-7)R(8)*) against fine-tuning (*T(1-7)R(8)*).

		Images per Class		
		L (1-20)	M (21-99)	H ($\geq$ 100)
Cosine Distance	L (0.0-0.2)	Freeze	Try Freeze or Tune	Tune
	M (0.2-0.4)	Try Freeze or Tune	Tune	Tune
	H (0.4-1.0)	Try Freeze or Tune	Tune	Tune

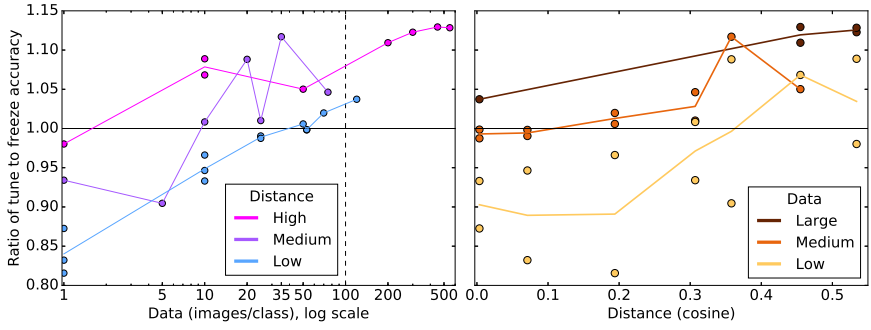

Fig. 2. Ratio between accuracy when fine-tuning and accuracy when freezing, measured across pairs of experiments with other conditions fixed. Only showing *R(8)* experiments. Values above 1.0 imply fine-tuning is better. Values below 1.0 imply freezing is better. **Left** examines the effect of data on this ratio. The dotted line denotes large amounts of data. **Right** examines the effect of source-target distance on this ratio. Trend lines are averages.

Fig. 2 (left) shows that with a large amount of target data, fine-tuning is always best, fitting our intuition that more data reduces overfitting. But Fig. 2 (right) shows that when there is a low or medium amount of training data, the distance between source and target plays a more important role. Broadly, as distance increases, fine-tuning improves relative to freezing, supporting the notion that learned features are less transferable to distant datasets. At one end, in the low data and low distance setting, freezing outperforms fine-tuning. At the other end, in the medium data and medium-to-high distance setting, fine-tuning outperforms freezing.

The inflection point occurs in settings where (1) target data is low and distance is medium or high, or (2) target data is moderate and distance is low. Here there is no consistent winner between fine-tuning and freezing. In these situations we recommend trying both fine-tuned and frozen networks. An additional consideration, though not shown in the figure, is that fine-tuned models generally take more time to train - one might choose a preferred training technique based on preference towards training speed or accuracy.

References

1. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proceedings of the European Conference on Computer Vision (ECCV). (2014)
2. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: International Conference in Machine Learning (ICML). (2014)
3. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Comput. Vis. Image Underst. (April 2007)
4. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)
5. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2010)
6. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems (NIPS). (2014)
7. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2014)
8. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: Proceedings of the European Conference on Computer Vision (ECCV). (2014)
9. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems (NIPS). (2014)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (NIPS). (2012)
11. Azizpour, H., Razavian, A., Sullivan, J., Maki, A., Carlsson, S.: Factors of transferability for a generic convnet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2015)
12. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
13. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2009)
14. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology (2011)
15. Beijbom, O., Edmunds, P.J., Kline, D.I., Mitchell, B.G., Kriegman, D.: Automated annotation of coral reef survey images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2012)
16. Orenstein, E.C., Beijbom, O., Peacock, E.E., Sosik, H.M.: Whoi-plankton- A large scale fine grained visual recognition benchmark dataset for plankton classification. CoRR abs/1510.00745 (2015)
17. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical report (2009)

18. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: International Conference in Computer Vision (ICCV). (2015)
19. Ganin, Y., Lempitsky, V.: Unsupervised Domain Adaptation by Backpropagation. In: ICML. (2015)
20. Long, M., Wang, J.: Learning transferable features with deep adaptation networks. In: ICML. (2015)