On the Impact of data set Size in Transfer Learning using Deep Neural Networks

Deepak Soekhoe, Peter van der Putten, and Aske Plaat

LIACS, Leiden University,
P.O.Box 9512, 2300 RA Leiden, The Netherlands
d.p.soekhoe@umail.leidenuniv.nl,
{p.w.h.van.der.putten,a.plaat}@liacs.leidenuniv.nl

Abstract. In this we paper we study the effect of target set size on transfer learning in deep learning convolutional neural networks. This is an important problem as labelling is a costly task, or for new or specific classes the number of labelled instances available may simply be too small. We present results for a series of experiments where we either train on a target of classes from scratch, retrain all layers, or subsequently lock more layers in the network, for the Tiny-ImageNet and MiniPlaces2 data sets. Our findings indicate that for smaller target data sets freezing the weights for the initial layers of the network gives better results on the target set classes. We present a simple and easy to implement training heuristic based on these findings.

Keywords: deep learning, convolutional neural networks, transfer learning, learning curves, AlexNet

1 Introduction

Current deep learning research achieves state-of-the-art performance in image classification tasks [6, 13, 15, 18]. Modern models make use of deep convolutional neural networks (CNN) such as AlexNet [8]. However, training these models on large data sets such as ImageNet [1] can take up a significant amount of time, and the number of labelled examples per class available may be limited, so learning from scratch has its downsides. One approach to overcome this problem is to use transfer learning. The objective of transfer learning is to use knowledge of a source task and *transfer* that to a new target task [10]. It provides considerable benefits over learning from scratch (i.e. from a random initialisation of the weights). One obvious advantage is that a model can learn more efficiently since it starts with a pre-initialised weight matrix.

In their study, Yosinski *et al.* [17] trained AlexNet on the ImageNet data set and found that the first three layers in a CNN contain generic and reusable features. Beyond the third layer, the features gradually become more specific with respect to the source data set. However, the authors did not take into

account the size of the target data set, on which the model with the transferred features will be trained.

The size of the target data set plays an important role, since it affects how much impact transfer learning will have on the performance. Thus, it is logical to ask how well extracted features generalise to smaller data sets. It would be helpful to know at what data set size transfer learning would be still beneficial. More specifically, at what layer is the model still able to generalize to a small data set size? Therefore, it is of both academic and practical interest to investigate at what *target data set* size transfer learning can still provide any additional value. Furthermore, Yosinski *et al.* only used the ImageNet data set [17]. It would be interesting to find out whether the transfer learning properties are different when using a data set from a different domain.

In this work we will expand the study by [17], and measure the effect of target data set size on the transferability of parameters in convolutional neural networks. Our main contribution is to quantify the extent to which features are able to generalise to the target data set when we systematically reduce its size. We will investigate this for each individual layer by evaluating the accuracy as a function of the data set size. We will have three variants of this. First, we will obtain a base score, without applying any form of transfer learning. In the second condition we will completely fine-tune all the layers of the network. In the third one, we will freeze the transferred features per individual layer. We will investigate this for different sizes of the target set. Moreover, we will test this on two different subsets of data sets, each with a different domain, ImageNet and Places2.

The remainder of this paper is organised as follows. Section 2 provides an overview of related work in deep learning and transfer learning. Section 3 describes our experimental setup and result, which addresses the data pre-processing steps we took, details about our feature transfer process and information regarding the training of the networks. In section 4 we elaborate on our results and report our main findings, and conclude the paper in section 5.

2 Related work

Several studies have investigated the generalizability of features and have proven the success of transfer learning [4, 9]. A popular strategy for transfer learning is fine-tuning, by training a linear classifier on top of the final layer of a CNN. Zeiler *et al.* [18] examined this by pre-training a CNN on ImageNet, and then training a linear classifier on three target data sets, PASCAL VOC 2012, Caltech-101 [3] and Caltech-256 [5]. They varied the target data set size, as well as the layer from which the classifier is trained on. They found that the model generalizes extremely well to Caltech-101 and Caltech-256, however less so to PASCAL. Nonetheless, the study proved the benefits of applying transfer learning. Similarly, good results were yielded in [11] using this approach of transfer learning. The authors pre-trained on ImageNet in combination with a SVM classifier, and use Pascal VOC and MIT-67 Indoor Scenes as target tasks.

In [2] the researchers investigated how well features transfer to different domain target problems, and they investigated at what layer in the network this is most optimal. They first trained AlexNet on the ImageNet data set, and tested these features on a basic object recognition task using the Caltech-101 [3] data set. Second, they tested the network on domain adaptation, where there is a small amount of data is available, using the Office database [12]. Thirdly, they tested how well their model performs on a more fine grained data set, using the Caltech-UCSD birds data set [16]. Since the images in this data set are very similar to each other, this is a rather difficult image classification task. Finally, the authors tested their model on the SUN-397 Large-Scale Scene Recognition database. This task is quite different from the source task, where the task was to classify objects. The objective of the SUN-397 data set is to classify scenic categories. In every experiment the authors improved the benchmark scores, indicating that the features learned from ImageNet provide substantial generalisable properties.

Our research is a direct extension of the work by Yosinski *et al.* [17]. They investigated how transferable features are between layers in the AlexNet architecture. To this end they trained two networks, N_1 and N_2, each on a random split of the ImageNet data set containing half of the data, split A and split B. After both networks were trained on their respective splits, the features of the first layer from network N_1, the base, were *transferred* to the first layer of network N_2, the target. The remaining layers in network N_2 were randomly initialised. Finally, network N_2 gets trained on the B partition of the ImageNet data set. Thus, what happens is that network N_2 does not train from scratch, but rather, it uses the pre-initialised features from network N_1. The researchers do this for layer one to seven in the network, transferring both from A to B as well as from B to A. They found that the features in the first three layers are fairly general and could be transferred and boost performance. However, features in deeper layers of the network are more specific to the source task and therefore, transferring them worsens the performance.

We hypothesize that a transfer learning approach by fixing the first layers is more valuable if the target set is smaller, and that for larger data sets updating all layers will give better results, and validate this on data sets from two different image classification domains.

3 Experiments and results

In this section we present an overview of our experimental set up and results. We will start with a general overview of the approach, and then provide further detail in further sub sections.

3.1 Overall approach

We will transfer features from a CNN trained on a source task, to a target task, i.e. data sets with disjunct outcome classes. We will consider the scenario where

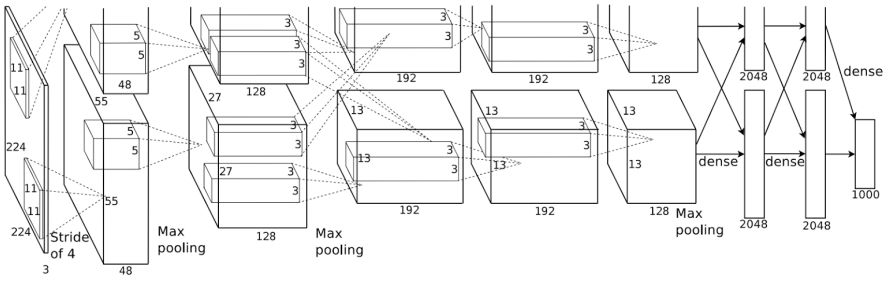

Fig. 1: AlexNet architecture (illustration taken from [8]).

the target data set is the same size, as well as smaller in size as the source data set. The latter condition is the conventional setting in transfer learning [10]. Hypothetically the value of transfer learning should increase with smaller transfer data sets. Moreover, for each scenario we will investigate the first case where we will fine-tune all the layers with the transferred features. In the second case we will transfer the features but freeze the network weights in the first layers.

The CNN architecture we will use is AlexNet, developed by Krizhevsky *et al.* [8] (see figure 1), which was the winning model in the ImageNet Large Scale Visual Recognition Challenge 2012. The model consists of five convolutional layers and three fully connected layers. The first two convolutional layers are followed by a max pooling layer and a normalization layer respectively. The fifth convolutional layer is followed only by a max pooling layer. The first two fully connected layers contain 4,096 neurons. The final fully connected layer contains 1000 neurons for the target class scores. It is interesting to note that the authors used Rectified Linear Units (ReLUs) as activation functions instead of the regular sigmoid. Moreover, they applied a regularization technique called dropout to reduce overfitting [14].

3.2 Data pre-processing

In our experiments, we use a subset of the ImageNet data set [1], **Tiny-ImageNet**. This data set contains 100,000 images with 200 classes, where each class contains 500 images, each of size 64 X 64 pixels. The data set contains images of a wide range of objects such as cats, parking meters, cliffs and rugby balls. The validation set contains 10,000 separate images.

Moreover, we extend the work by Yosinski *et al.* [17] by also repeating the experiments on a second data set, **MiniPlaces2**. This is a scaled down version of the larger MIT Places database [19]. The data set is made up of images with settings such as a food court, golf course, an office, and ice skating rink. It contains 100,000 images with 100 classes. Each class consists of 1000 pictures of size 128 X 128 pixels, however we resize them to 64 X 64 pixels to keep the

(a) Lighthouse

(b) Sulphur butterfly

(c) Umbrella

(d) Museum

(e) Baseball field

(f) Valley

Fig. 2: *Top:* A sample of training images from the Tiny-ImageNet data set. *Bottom:* A sample of training images from the MiniPlaces2 data set.

image size consistent with Tiny-ImageNet. Again, the validation set contains 10,000 images. In figure 2 we present several image classes of both data sets to underline the difference between the two domains.

To measure the effect of data set size on the generalizability of features, we transfer the features from a source task to a target task, where the latter has a variable size. We will test this on a subset of the ImageNet and Places2 data set. We use a subset of the data sets rather than training on the full data sets of ImageNet and Places2 (respectively containing 1.2 million and 8.1 million images for training) due to computational limitations. We denote our target data set as N_{target}. Moreover, we define the data set splits with a variable size as M_{target_i} where $M_{target_i} \subseteq N_{target}$. To obtain M_{target_i} from N_{target} we execute the following procedure:

1) We randomly split the entire data set into a source and a target partition, N_{source} and N_{target} respectively, where each partition contains 50,000 images. In both the source and target partition the images are equally distributed over $k = 100$ classes with 500 images per class for Tiny-ImageNet. In MiniPlaces2 the split is $k = 50$ classes per partition, with 1,000 images per class.
2) We artificially reduce N_{target} by drawing random samples of size M_{target_i} from each class k, where i equals 500[1], 400, 300, 200, 100 and 50 in case of Tiny-ImageNet. For MiniPlaces2 i equals 1000[1], 900, 800, 700, 600 and 500.

[1] Note that in the case where $i = 500$ and $i = 1{,}000$ we do not reduce N_{target} for Tiny-ImageNet and MiniPlaces2 respectively.

Moreover, for both Tiny-ImageNet and MiniPlaces2, we split the respective validation sets in half to create V_{target} and V_{source}, each containing 5,000 test images. The classes of V_{target} correspond to the classes in N_{target}. Therefore, V_{target} will be the validation set for M_{target_i} in our experiments. The other validation half, V_{source}, contains classes corresponding to N_{source}. In sum, we train our model on M_{target_i}, and evaluate it on a separate validation set V_{target}, to obtain our accuracy a.

3.3 Transferring features

To create a model from which we can transfer the features, we first train our network on N_{source}. The parameters of the source model are stored in a Caffemodel object (see section 3.4), which we use to transfer the parameters from the source model to the target model.

To obtain our baseline score we do not apply any transfer learning at all, and let the model train on the given training set. In our first experiment we fine-tune the network by transferring all the features from the source task to the model, and continue with backpropagation on the new task.

However, since we are also interested in at what layer l of the network features are able to generalize, we transfer the features from the source to the target task, one layer at a time. AlexNet has eight layers in total. Therefore, we transfer from layer $l = 1$, up until layer $l = 7$. When we transfer the parameters to the target model, we keep them fixed. That is to say, we do not update the parameters by gradient descent. The remaining 8 - l layers of the network we randomly initialize and let the errors backpropagate through the layers.

Finally, to get a mean accuracy score, we run the experiments again by following the same procedure, but now use N_{source} as N_{target} and vice versa.

3.4 Training

To conduct our experiments, we use the Caffe deep learning framework developed at UC Berkeley [7]. We make use of a single Nvidida GTX Titan X graphics card to enable Caffe in GPU mode, to speed up our training time. We use the AlexNet reference model which is included in Caffe. Detailed information about the model architecture can be found in [8]. Moreover, we follow the same training regime as specified by the reference model.

Furthermore, in terms of data augmentation we take a random crop in the training phase and use random mirroring as specified by Caffe. In the test phase we take a center crop of the images. Since our input images are 64 X 64, we change the crop size to 57, rather than upscaling the images to 256 X 256 and applying the default crop size of 227. Thus, we stay consistent with the ratio used in the AlexNet reference model. Moreover, we subtract the image mean from each image.

Finally, to determine for how many iterations we should train the models, we trained on N_{source} of both data sets and validated on the respective V_{source}, without applying any form of transfer learning.

We found that the model began to overfit on the training data around 10,000 iterations (see figure 3 and 4). Therefore, we found it reasonable for subsequent experiments to let each model run for 10K iterations in order to measure the positive effect of transfer learning. Moreover, the more we reduce N_{target}, the faster the model will reach the point of overfitting, which is evidenced by the decreasing accuracy of the *base* training conditions across our experiments.

3.5 Results Tiny-ImageNet

In figure 5 we see the results of transfer learning on different data set sizes. The plot shows the accuracy on the validation set after 10K iterations of training. The first two conditions are the base case and fine-tune all. The condition *base* indicates we did not apply transfer learning. Condition $FTall$ means we fine-tuned through all the layers, and the notation SnT denotes up until which layers we freeze the transferred features from the source in the target model. For instance, $S3T$ implies we transferred the first three feature layers from the model trained on N_{source} to the model trained on M_{target_i}. The final seven scores are the accuracies where we transfer the parameters per layer from the source, and freeze that particular layer. We notice an effect of data set size on the accuracy of the baseline score. As we decrease the data set size, we find that the accuracy decreases as well. In figure 5 we observe that the accuracy worsens as we keep more layers fixed when transferring parameters from the source task.

3.6 Results MiniPlaces2

As can be seen from figure 6, even though this is a task from a different domain, the results follow a pattern very similar to Tiny-ImageNet. With smaller target set sizes the benefits of locking the first few layers increases. Only for $M_{target_{1000}}$ the graphs seem to indicate that training from scratch is better, but this is truly just a baseline. In a real deployment one would probably expect that the source classes also still need to be recognized, and performance of tuning all layers is still lower then locking some of the initial layers.

4 Discussion

Our results reveal that data set size affects the accuracy in transfer learning with deep convolutional neural networks. The first effect we notice is on the baseline case (to repeat, just training the network with randomly initialized weights). We can see that the model starts to overfit on the training data when we artificially reduce the data set size, which leads to a steady decline in accuracy on both Tiny-ImageNet as well as MiniPlaces2. This can be explained by a sub-optimal parameter configuration as a result of overfitting on a small data set size.

Furthermore, fine-tuning all the layers only appears to have a positive effect with smaller data sets for Tiny-ImageNet where i in M_{target_i} ranges from 400 until 50, and MiniPlaces2 where M_{target_i} equals 500. This is an interesting result,

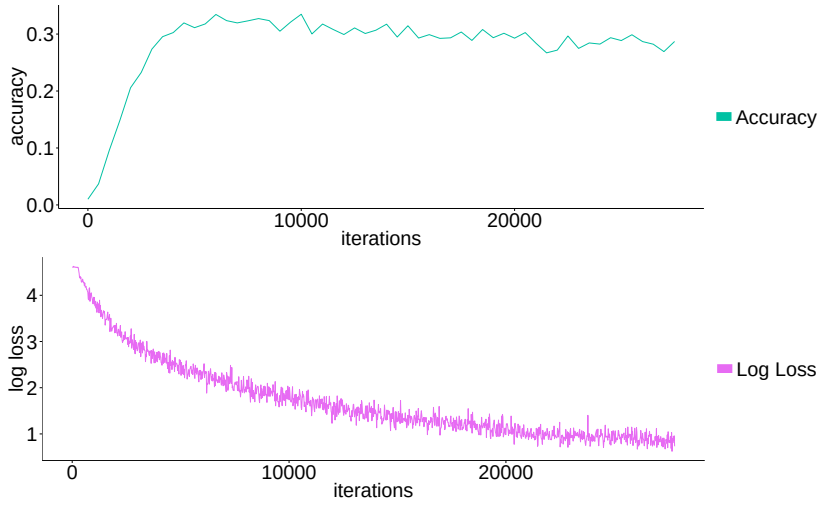

Fig. 3: *Top:* The accuracy on V_{source} after training on N_{source} of the Tiny-ImageNet data set after 25K iterations. This split contains 100 classes, with 500 images per class. *Bottom:* The log loss over the training set with the identical split.

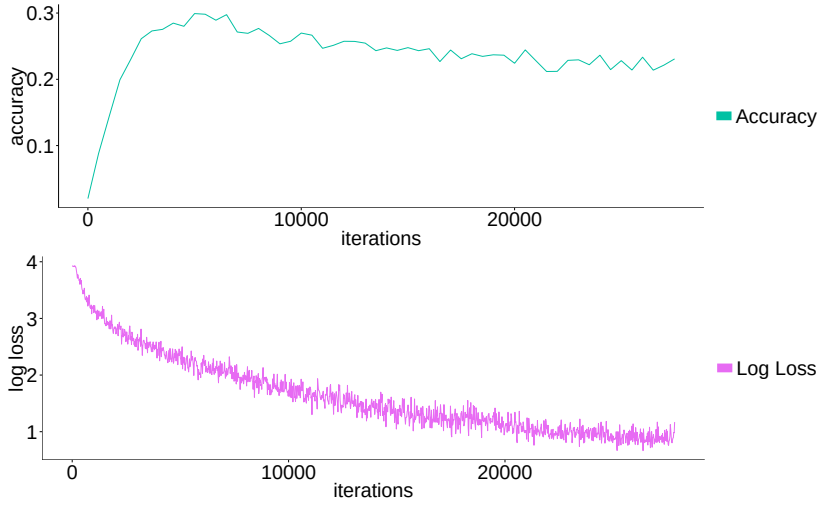

Fig. 4: *Top:* The accuracy on V_{source} after training on N_{source} of the MiniPlaces2 data set after 25K iterations. This split contains 50 classes, with 1000 images per class. *Bottom:* The log loss over the training set with the identical split.

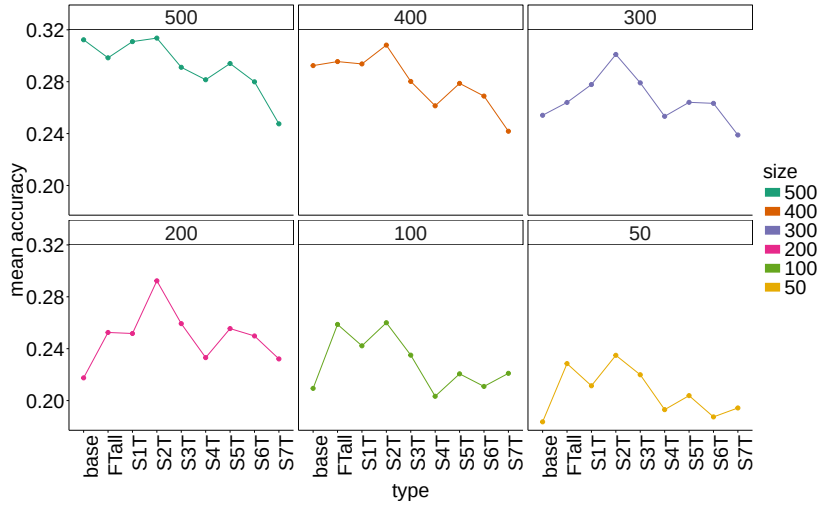

Fig. 5: Mean accuracy obtained after training on the target splits of Tiny-ImageNet where i in M_{target_i} equals 500, 400, 300, 200, 100 and 50 and validating on V_{target}. Note that we ran the same experiments again, but used N_{source} as N_{target} and vice versa. Thus, we obtained our mean accuracies by averaging the scores.

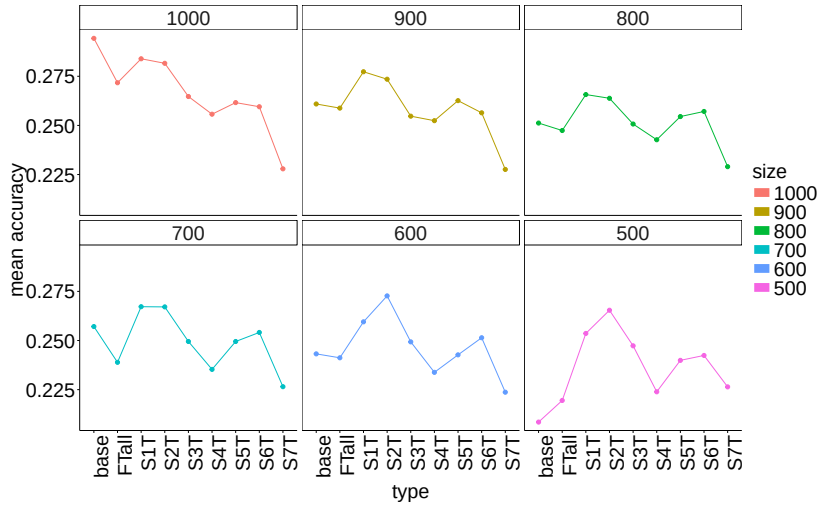

Fig. 6: Mean accuracy after training on the target splits of MiniPlaces2, where i in M_{target_i} equals 1000, 900, 800, 700, 600 and 500 and validating on V_{target}.

as a network for which all layers can be adapted still benefits from potentially valuable initialization of the weights. We speculate that the source features are important for the target data set splits as well. Thus, the effect of initializing the model with parameters obtained from a model trained on a larger data set clearly shows its advantage. Moreover, we notice a visible spike in accuracy in all our graphs, when we transfer parameters from the first two layers. Likewise, there is a considerable decline in accuracy when transferring four layers, compared to transferring the first three layers.

The results in figure 5 generally follow the findings of the study by Yosinski *et al.* [17]. As we transfer more and more features (layers) from the source task, the accuracy initially goes up but then decreases. This can be attributed to feature specificity with regards to the source task. However, we observe a second positive spike in the accuracy at layer $l = 5$ in nearly all of our experiments. This result is quite surprising since the features have become substantially specific to the source, and yet generalize well to the new task. Evidently, the transferred features from the source task in this layer hold the same, or even superior, representational power compared to the features solely learned from a target data set.

All these results can be summarized into a fairly straightforward heuristic. For the first n instances of a new class, freeze the first l layers of the network. Once you have obtained more than n instances for new class, training can simply affect all layers. Obviously the values for n and l depend on the data and task at hand, in our experiments freezing the first 3 layers until 300 (Tiny-ImageNet) and respectively 900 (MiniPlaces2) instances per class gave the best results.

Our study could have benefited from having more samples per data point, by running repeated experiments. Since the initialization of the parameters happens at random, the parameters might converge at different local minima each time the model is run. This could effect the accuracy score in the test phase. Our results still indicate that transferring features from a larger source data set to a smaller target data set adds value by reducing the risk of overfitting, and improves performance.

5 Conclusion

In this paper we investigated the effect of data set size on the generalizability of features in deep convolutional neural networks. To this end, we transferred features from a pre-trained network to a new network. We systematically reduced the size of the target training set and trained our new network on these splits with the pre-initialized features. In support for a general rule of thumb heuristic, we found that freezing the first two to three layers of features results in a significant performance boost over the baseline score, especially for smaller target set sizes under a thousand instances per class.

References

1. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. pp. 248–255. IEEE (2009)
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. arXiv preprint arXiv:1310.1531 (2013)
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. Computer Vision and Image Understanding 106(1), 59–70 (2007)
4. 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. pp. 580–587 (2014)
5. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset (2007)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
7. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia. pp. 675–678. ACM (2014)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
9. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1717–1724 (2014)
10. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22(10), 1345–1359 (2010)
11. Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: CNN features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 806–813 (2014)
12. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: Computer Vision–ECCV 2010, pp. 213–226. Springer (2010)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
14. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1), 1929–1958 (2014)
15. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015)
16. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-ucsd birds 200 (2010)
17. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems. pp. 3320–3328 (2014)

18. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Computer vision–ECCV 2014, pp. 818–833. Springer (2014)
19. 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. pp. 487–495 (2014)