What makes ImageNet good for transfer learning?

Minyoung Huh Pulkit Agrawal Alexei A. Efros
Berkeley Artificial Intelligence Research (BAIR) Laboratory
UC Berkeley
{minyoung,pulkitag,aaefros}@berkeley.edu

Abstract

The tremendous success of features learnt using the ImageNet classification task on a wide range of transfer tasks begs the question: what are the intrinsic properties of the ImageNet dataset that are critical for learning good, general-purpose features? This work provides an empirical investigation of various facets of this question: Is more pre-training data always better? How does feature quality depend on the number of training examples per class? Does adding more object classes improve performance? For the same data budget, how should the data be split into classes? Is fine-grained recognition necessary for learning good features? Given the same number of training classes, is it better to have coarse classes or fine-grained classes? Which is better: more classes or more examples per class?

1. Introduction

It has become increasingly common within the computer vision community to treat image classification on ImageNet [37] not as an end in itself, but rather as a "pretext task" for training deep convolution neural networks (CNNs [27, 25]) to learn good general-purpose features. This practice of first training a CNN to perform image classification on ImageNet (i.e. pre-training) and then adapting these features for a new target task (i.e. fine-tuning) has become the de facto standard for solving a wide variety of computer vision problems. Using ImageNet pre-trained CNN features, impressive results have been obtained on several image classification datasets [13, 35], object detection [16, 39], action recognition [41], human pose estimation [9], image segmentation [10], optical flow [47], image captioning [12, 22] and others [26].

Our work can be found on our website
http://minyounghuh.com/papers/analysis

Given the success of ImageNet pre-trained CNN features, it is only natural to ask: what is it about the ImageNet dataset that makes the learnt features as good as they are? One school of thought believes that it is the sheer size of the dataset (1.2 million labelled images) that forces the representation to be general. Others argue that it is the large number (1000) of distinct object classes, which forces the network to learn a hierarchy of generalizable features. Yet others believe that the secret sauce is not just the large number of classes, but the fact that many of these classes are visually similar (e.g. many different breeds of dogs), turning this into a fine-grained recognition task, and therefore pushing the representation to work harder. But, while almost everyone in computer vision seems to have their own opinion on this hot topic, very little hard empirical evidence has so far been produced.

In this work, we systematically investigate which aspects of the ImageNet task are most critical for learning good general-purpose features. We evaluated the features by finetuning for object detection on PASCAL-VOC 2007 dataset (PASCAL-DET), action classification on PASCAL-VOC 2012 dataset (PASCAL-ACT-CLS) and scene classification on the SUN dataset (SUN-CLS); see Section 3 for more details. The following is a summary of our main findings:

1. How many pre-training ImageNet examples are sufficient for transfer learning? Pre-training with only *half* the ImageNet data (500 images per class instead of 1000) results in only a small drop in transfer learning performance (1.5 mAP drop on PASCAL-DET). This drop is much smaller than the drop on the ImageNet classification task itself. See Section 4 and Figure 1 for details.

2. How many pre-training ImageNet classes are sufficient for transfer learning? Pre-training with an order of magnitude *fewer* classes (127 classes instead of 1000) results in only a small drop in transfer learning performance (drop of 2.8 mAP on PASCAL-DET). Quite interestingly, we also found that for some transfer tasks, pre-training with

fewer number of classes leads to *better* performance. See Section 5.1 and Figure 2 for details.

3. How important is fine-grained recognition for learning good features for transfer learning? The above experiment also suggests that transferable features are learnt even when a CNN is pre-trained with a set of classes that do not require fine-grained discrimination. See Section 5.2 and Figure 2 for details.

4. Given the same budget of pre-training images, should we have more classes or more images per class? Training with fewer classes but more images per class performs slightly better than training with more classes but fewer images per class. See Section 5.5 and Table 2 for details.

5. Is more data always helpful? We found that training using 771 ImageNet classes that *excludes* all PASCAL VOC classes, achieves nearly the same performance on PASCAL-DET as training on complete ImageNet. Further experiments confirm that blindly adding more training data does not always lead to better performance and can sometimes hurt performance. See Section 6, and Table 9 for more details.

2. Related Work

Understanding internal representations of CNN: For understanding what information is encoded by different CNN features, several works have developed feature visualization methods using either deconvolution [50], backpropagation[40, 52] or by training a decoder network [14]. Another line of prior work investigated questions such as how distributed are CNN representations [2, 43], what is the the effect of feature transformations such as binarization [2, 14], how do CNN features change with systematic variations in scene factors [3], how invariant are features to different viewpoints [5] and how is class specific information organized in groups of CNN units [46, 44]. An interesting observation that CNNs have "blindspots" was made by [43]. Recently, a mathematical framework for analyzing CNN features was provided by [28]. In contrast to analyzing features learned by training on the full ImageNet dataset, our paper analyzes what aspects of data are important for learning transferable features.

Factors that affect fine-tuning: The question of whether pre-training should be terminated early to prevent overfitting and what layers should be used for transfer learning was studied by [2, 49]. A thorough investigation of good architectural choices for transfer learning was conducted by [4]. In contrast to these works, we use a fixed finetuning procedure and investigate the factors of variation in pre-training data that effect transfer performance.

Other pre-training methods: A common assumption in supervised pre-training is that large quantity of expensive manually-supervised training data is required. The possibility of using large amounts of unlabelled data for feature learning has therefore been very attractive. Numerous methods for learning features by optimizing some auxiliary criterion of the data itself have been proposed. The most well-known such criteria are image reconstruction [7, 38, 31, 29, 34, 23] (see [6] for a comprehensive overview) and feature slowness [48, 18].

Unfortunately, these unsupervised methods turned out not to be competitive with those obtained from supervised ImageNet pre-training. To try and force better feature generalization, more recent "self-supervised" methods use more difficult data prediction auxiliary tasks in an effort to make the CNNs "work harder". Attempted self-supervised tasks include predictions of ego-motion [1, 20], spatial context [11, 33, 30], temporal context [45], and even color [51] and sound [32]. While features learned using these methods often come close to ImageNet performance, to date, none has been able to beat it. Therefore, in this work, we are trying to understand what is the secret to ImageNet's continuing success.

3. Experimental Setup

The process of using supervised learning to initialize CNN parameters using the task of ImageNet classification is referred to as pre-training. The pre-trained CNN is adapted by continued training on a target dataset, this process is referred to as finetuning. All of our experiments use the Caffe [21] implementation of the a single network architecture proposed by Krizhevsky et al. [25]. We refer to this architecture as AlexNet.

We closely follow the experimental setup of Agrawal et al. [2] for evaluating the generalization of pre-trained features on three transfer tasks: PASCAL VOC 2007 object detection (PASCAL-DET), PASCAL VOC 2012 action recognition (PASCAL-ACT-CLS) and scene classification on SUN dataset (SUN-CLS). For PASCAL-DET, we used the PASCAL VOC 2007 train/val for finetuning using the experimental setup and code provided by Faster-RCNN [36] and report performance on the test set. For PASCAL-ACT-CLS, we used PASCAL VOC 2012 train/val for finetuning and testing using the experimental setup and code provided by R*CNN [17]. For SUN-CLS we used the same train/val/test splits as used by [2]. Finetuning on SUN was performed by first replacing the FC-8 layer in the AlexNet model with a randomly initialized, and fully connected layer with 397 output units. Finetuning was performed for 50K iterations using stochastic gradient descent (SGD) with an initial learning rate of 0.001 which was reduced by a factor of 10 (1/10-th) every 20K iterations.

In order to account for difference in performance that may result from different runs of Faster-RCNN and R*CNN, we ran each finetuning experiment 3 times to com-

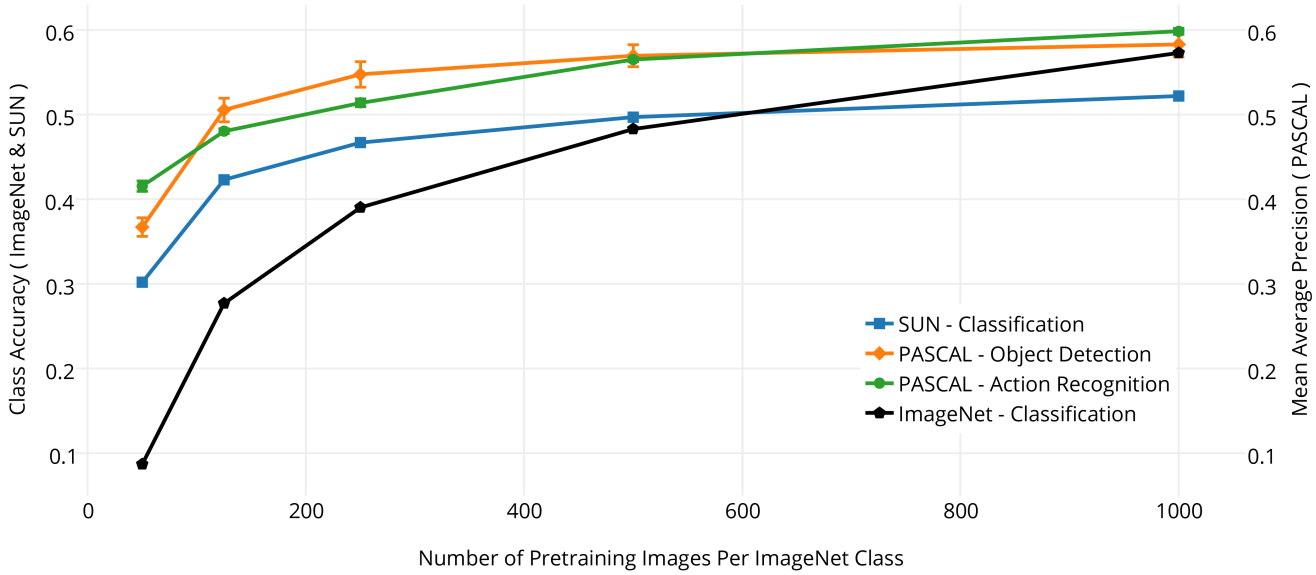

Figure 1: The change in transfer task performance of a CNN pre-trained with varying number of images per ImageNet class. The number of ImageNet classes are varied using the technique described in Section 5.1. The y-axis on the left represents mean class accuracy and is used for SUN-CLS and ImageNet-CLS. The y-axis on right measures mean average precision (mAP) and is used for PASCAL-DET and PASCAL-ACT-CLS. Accuracy on the ImageNet classification task increases faster as compared to performance on transfer tasks with increase in amount of ImageNet pre-training data.

pute the variance in performance. We report the performance as mean $\pm$ standard deviation. Due to computational constraints we only report SUN-CLS results using a single run. As reported by [2], error bars on SUN-CLS are small, therefore having a single run should not affect the conclusions that we draw.

In some experiments we pre-train on ImageNet using a different number of images per class. The model with 1000 images/class uses the original ImageNet ILSVRC 2012 training set. Models with N images/class for $N < 1000$ are trained by drawing a random sample of N images from all images of that class made available as part of the ImageNet training set.

4. How does the amount of pre-training data affect transfer performance?

For answering this question, we trained 5 different AlexNet models from scratch using 50, 125, 250, 500 and 1000 images per each of the 1000 ImageNet classes using the procedure described in Section 3. The variation in performance with amount of pre-training data when these models are finetuned for PASCAL-DET, PASCAL-ACT-CLS and SUN-CLS is shown in Figure 1. For PASCAL-DET, the mean average precision (mAP) for CNNs with 1000, 500 and 250 images/class is found to be 58.3, 57.0 and 54.6. A similar trend is observed for PASCAL-ACT-CLS and SUN-CLS. These results indicate that using half the amount of pre-training data leads to only a *marginal* reduction in performance on transfer tasks. It's important to note that the performance on the ImageNet classification task (the pre-training task) steadily increases with the amount of training data, whereas on transfer tasks, the performance increase with respect to additional pre-training data is significantly slower.

5. How does the taxonomy of the pre-training task affect transfer performance?

In the previous section we investigated how varying number of pre-training images per class effects the performance in transfer tasks. Here we investigate the flip side: keeping the amount of data constant while changing the nomenclature of training labels. Through manipulating different aspects of the ImageNet dataset, we hope to narrow in on what makes ImageNet good for transfer tasks.

5.1. The effect of number of pre-training classes on transfer performance

The 1000 classes of the ImageNet challenge [37] are derived from leaves of the WordNet tree [15]. Using this tree, it is possible to generate different class taxonomies while keeping the total number of images constant. One can generate taxonomies in two ways: (1) bottom up clustering, wherein the leaf nodes belonging to a common parent are iteratively clustered together (see Figure 3), or (2) by fixing the distance of the nodes from the root node (i.e. top down clustering). Using bottom up clustering, 18 possible

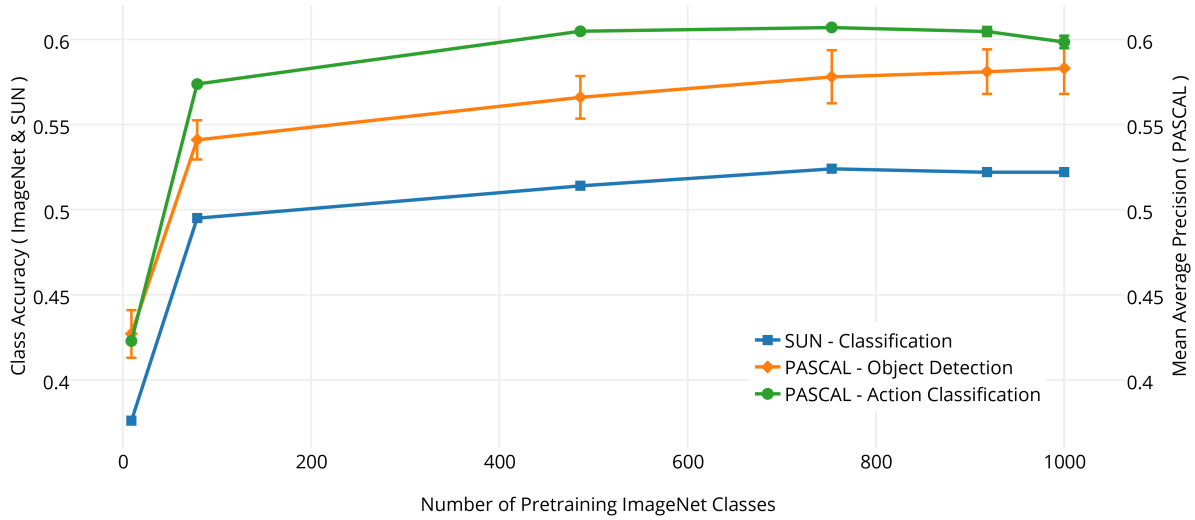

Figure 2: Change in transfer task performance with varying number of pre-training ImageNet classes. The x-axis shows the number of pre-training ImageNet classes. The y-axis on the left represents mean class accuracy and is used for SUN-CLS and ImageNet-CLS. The y-axis on right measures mean average precision (mAP) and is used for PASCAL-DET and PASCAL-ACT-CLS. With only 486 pre-training classes, transfer performances are unaffected and only a small drop is observed when only 79 classes are used for pre-training.

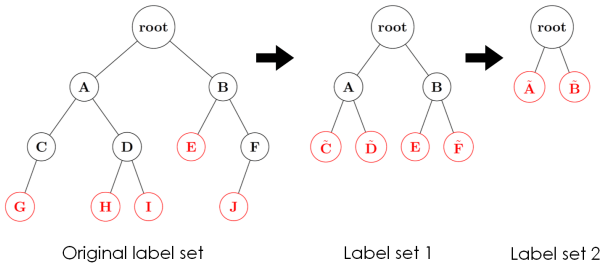

Figure 3: An illustration of the bottom up procedure used to construct different label sets using the WordNet tree. Each node of the tree represents a class and the leaf nodes are shown in red. Different label sets are iteratively constructed by clustering together all the leaf nodes with a common parent. In each iteration, only leaf nodes are clustered. This procedure results into a sequence of label sets for 1.2M images, where each consequent set contains labels coarser than the previous one. Because the WordNet tree is imbalanced, even after multiple iterations, label sets contain some classes that are present in the 1000 way ImageNet challenge.

taxonomies can be generated. Among these, we chose 5 sets of labels constituting 918, 753, 486, 79 and 9 classes respectively. Using top-down clustering only 3 label sets of 127, 10 and 2 can be generated, and we used the one with 127 classes. For studying the effect of number of pre-training classes on transfer performance, we trained separate AlexNet CNNs from scratch using these label sets.

Figure 2 shows the effect of number of pre-training classes obtained using bottom up clustering of WordNet tree on transfer performance. Table 1 shows the same results for 127 classes obtained from top down clustering. Using only 486 classes results in a performance drop of 1.7 mAP for PASCAL-DET, 0.8% accuracy for SUN-CLS and a boost of 0.6 mAP for PASCAL-ACT-CLS. Moreover, only diminishing returns in transfer performance are observed when more than 127 classes are used.

It can be argued that the PASCAL task requires discrimination between only 20 classes and therefore pre-training with only 127 classes should not lead to substantial reduction in performance. However, the trend also holds true for SUN-CLS that requires discrimination between 397 classes. These two results taken together suggest that although training with a large number of classes is beneficial, diminishing returns are observed beyond using 127 distinct classes for pre-training.

Furthermore, for PASCAL-ACT-CLS and SUN-CLS, finetuning on CNNs pre-trained with class set sizes of 918, and 753 actually results in better performance than using all 1000 classes. This can indicate a few things: either having too many classes for pre-training works against learning good generalizable features, or that the original labeling set of ImageNet is suboptimal. If latter is the case, then there exists a better labeling set for ImageNet that can improve the performance on transfer tasks. Though nontrivial, it may be interesting to investigate discovering such an optimal label set.

5.2. Is fine-grain recognition necessary for learning transferable features?

ImageNet challenge requires a classifier to distinguish between 1000 classes, some of which are very fine-grained, such as different breeds of dogs and cats. Indeed, most humans don't perform well on ImageNet unless specifically trained [37], and yet are easily able to perform most everyday visual tasks. This raises the question: is fine-grained

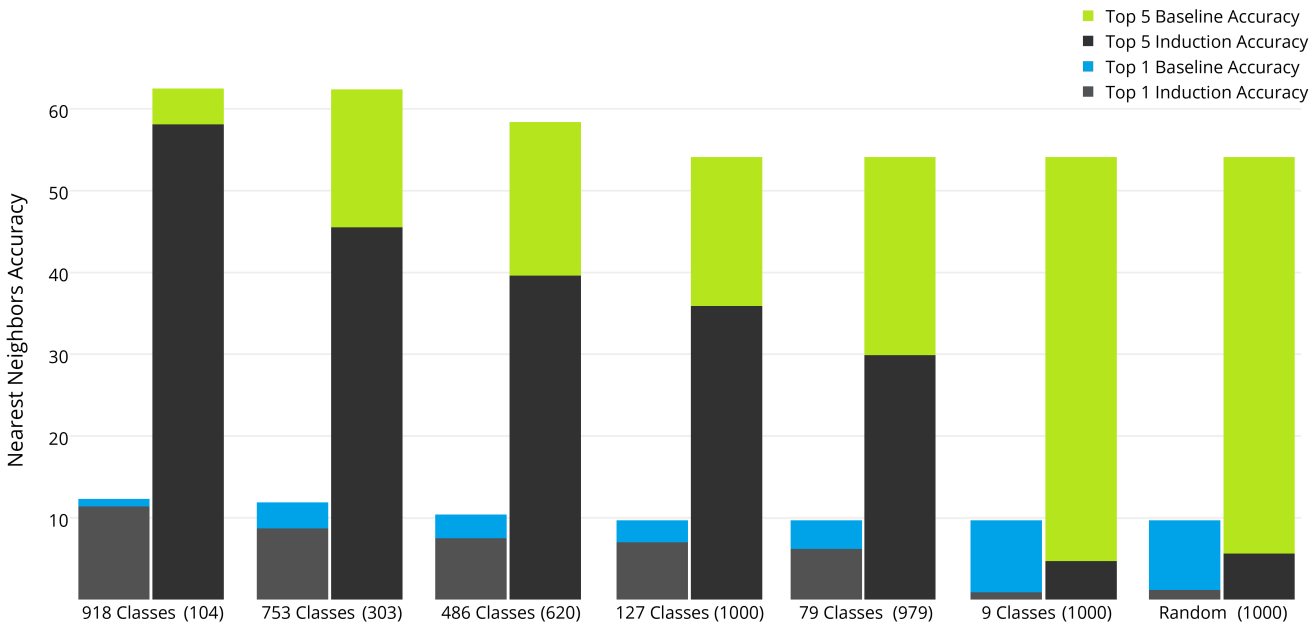

Figure 4: Does a CNN trained for discriminating between coarse classes learns a feature embedding capable of distinguishing between fine classes? We quantified this by measuring the induction accuracy defined as following: after training a feature embedding for a particular set of classes (set A), the induction accuracy is the nearest neighbor (top-1 and top-5) classification accuracy measured in the FC8 feature space of the subset of 1000 ImageNet classes not present in set A. The syntax on the x-axis `A Classes(B)` indicates that the network was trained with A classes and the induction accuracy was measured on B classes. The baseline accuracy is the accuracy on B classes when the CNN was trained for all 1000 classes. The margin between the baseline and the induction accuracy indicates a drop in the network's ability to distinguish fine classes when being trained on coarse classes. The results show that features learnt by pre-training on just 127 classes still lead to fairly good induction.

Pre-trained Dataset	PASCAL	SUN
Original	58.3	52.2
127 Classes	55.5	48.7
Random	41.3 [24]	30.0

Table 1: The transfer performance of a network pre-trained using 127 (coarse) classes obtained after top-down clustering of the WordNet tree is comparable to a transfer performance after fine-tuning on all 1000 ImageNet classes. This indicates that fine-grained recognition is not necessary for learning good transferable features.

recognition necessary for CNN models to learn good feature representations, or is coarse-grained object recognition (e.g. just distinguishing cats from dogs) is sufficient?

Note that the label set of 127 classes from the previous experiment contains 65 classes that are present in the original set of 1000 classes and the remainder are inner nodes of the WordNet tree. However, all these 127 classes (see supplementary materials) represent coarse semantic concepts. As discussed earlier, pre-training with these classes results in only a small drop in transfer performance (see Table 1). This suggests that performing fine-grained recognition is only marginally helpful and does not appear to be critical for learning good transferable features.

5.3. Does training with coarse classes induce features relevant for fine-grained recognition?

Earlier, we have shown that the features learned on the 127 coarse classes perform almost as well on our transfer tasks as the full set of 1000 ImageNet classes. Here we will probe this further by asking a different question: is the feature embedding induced by the coarse class classification task capable of separating the fine labels of ImageNet (which it never saw at training)? To investigate this, we used top-1 and top-5 nearest neighbors in the FC7 feature space to measure the accuracy of identifying fine-grained ImageNet classes after training only on a set of coarse classes. We call this measure, "induction accuracy". As a qualitative example, Figure 5 shows nearest neighbors for a macaque (left) and a schnauzer (right) for feature embeddings trained on ImageNet but with different number of classes. All green-border images below the dotted line indicate instances of correct fine-grain nearest neighbor retrieval for features that were never trained on that class.

Quantitative results are shown in Figure 4. The results show that when 127 classes are used, fine-grained recognition k-NN performance is only about 15% lower compared to training directly for these fine-grained classes (i.e. baseline accuracy). This is rather surprising and suggests that

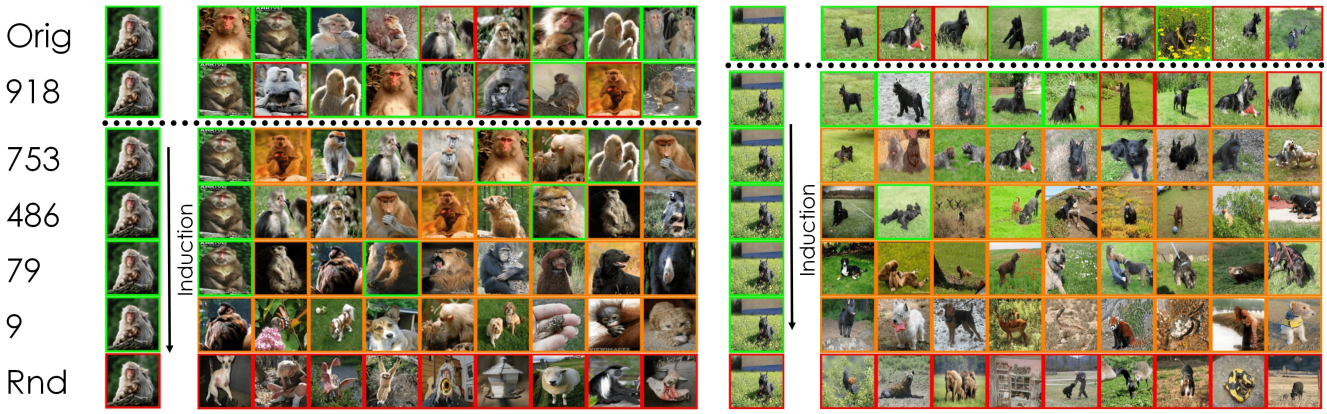

Figure 5: Can feature embeddings obtained by training on coarse classes be able to distinguish fine classes they were never trained on? E.g. by training on monkeys, can the network pick out macaques? Here we look at the FC7 nearest neighbors (NN) of two randomly sampled images: a macaque (left column) and a giant schnauzer (right column), with each row showing feature embeddings trained with different number of classes (from fine to coarse). The row(s) above the dotted line indicate that the image class (i.e. macaque/giant schnauzer) was one of the training classes, whereas in rows below the image class was not present in the training set. Images in green indicate that the NN image belongs to the correct fine class (one of 1000); orange indicates the correct coarse class (based on the WordNet hierarchy) but incorrect fine class; red indicated incorrect coarse class. All green images below the dotted line indicate instances of correct fine-grain nearest neighbor retrieval for features that were never trained on that class.

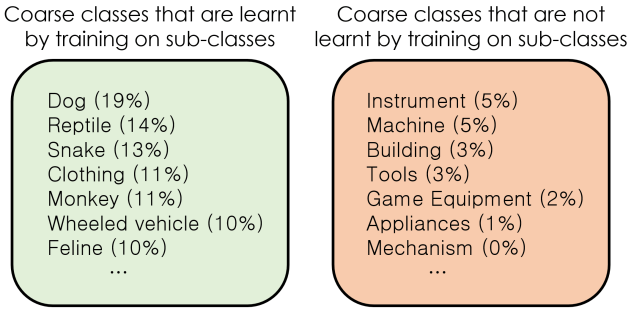

Figure 6: Does the network learn to discriminate coarse semantic concepts by training only on finer sub-classes? The degree to which the concept of coarse class is learnt was quantified by measuring the difference (in percentage points) between the accuracy of classifying the coarse class and the average accuracy of individually classifying all the sub-classes of this coarse class. Here, the top and bottom classes sorted by this metric are shown. We observe that classes whose subclasses are visually consistent (e.g. dogs, monkey) are better represented than these that are visually dissimilar (e.g. instruments, machines).

CNNs implicitly discover features capable of distinguishing between finer classes while attempting to distinguish between relatively coarse classes.

5.4. Does training with fine-grained classes induce features relevant for coarse recognition?

Investigating whether the network learns features relevant for fine-grained recognition by training on coarse classes begs the reverse question: does training with fine-grained classes induce features relevant for coarse recognition? If this is indeed the case, then we would expect

Dataset	PASCAL			SUN		
Data size	500K	250K	125K	500K	250K	125K
More examples/class	57.1	54.8	50.6	50.6	45.7	42.2
More classes	57.0	52.5	49.8	49.7	46.7	42.3

Table 2: For a fixed budget of pre-training data, is it better to have more examples per class and fewer classes or vice-versa? The row 'more examples/class' was pretrained with subsets of ImageNet containing 500, 250 and 125 classes with 1000 examples each. The row 'more classes' was pretrained with 1000 classes, but 500, 250 and 125 examples each. Interestingly, the transfer performance on both PASCAL and SUN appears to be broadly similar under both scenarios.

that when a CNN makes an error, it is more likely to confuse a sub-class (i.e. error in fine-grained recognition) with other sub-classes of the same coarse class. This effect can be measured by computing the difference between the accuracy of classifying the coarse class and the average accuracy of individually classifying all the sub-classes of this coarse class (please see supplementary materials for details).

Figure 6 shows the results. We find that coarse semantic classes such as dogs, reptiles, clothing etc that contain visually similar sub-classes show the hypothesized effect, whereas classes such as instruments and buildings that contain visually dissimilar subclasses do not exhibit this effect. These results indicate that subclasses that share a common visual structure allow the CNN to learn features that are more generalizable. This might suggest a way to improve feature generalization by making class labels respect visual commonality rather than simply WordNet semantics.

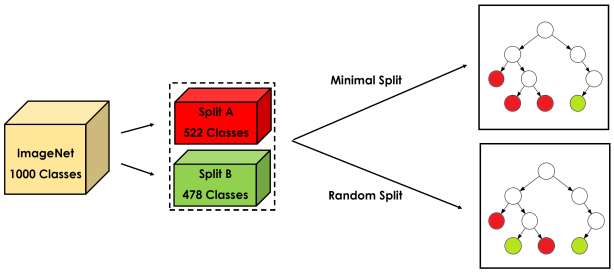

Figure 7: An illustration of the procedure used to split the ImageNet dataset. Splits were constructed in 2 different ways. The random split selects classes at random from the 1000 ImageNet classes. The minimal split is made in a manner that ensures no two classes in the same split have a common ancestor up to depth four of WordNet tree. Collage in Figure 8 visualizes the random and minimal splits.

Pre-trained Dataset	PASCAL
ImageNet	58.3 ± 0.3
Pascal removed ImageNet	57.8 ± 0.1
Places	53.8 ± 0.1

Table 3: PASCAL-DET results after pre-training on entire ImageNet, PASCAL-removed-ImageNet and Places data sets. Removing PASCAL classes from ImageNet leads to an insignificant reduction in performance.

5.5. More Classes or More Examples Per Class?

Results in previous sections show that it is possible to achieve good performance on transfer tasks using significantly less pre-training data and fewer pre-training classes. However it is unclear what is more important – the number of classes or the number or examples per class. One extreme is to only have 1 class and all 1.2M images from this class and the other extreme is to have 1.2M classes and 1 image per class. It is clear that both ways of splitting the data will result in poor generalization, so the answer must lie somewhere in-between.

To investigate this, we split the same amount of pre-training data in two ways: (1) more classes with fewer images per class, and (2) fewer classes with more images per class. We use datasets of size 500K, 250K and 125K images for this experiment. For 500K images, we considered two ways of constructing the training set – (1) 1000 classes with 500 images/class, and (2) 500 classes with 1000 images/class. Similar splits were made for data budgets of 250K and 125K images. The 500, 250 and 125 classes for these experiments were drawn from a uniform distribution among the 1000 ImageNet classes. Similarly, the image subsets containing 500, 250 and 125 images were drawn from a uniform distribution among the images that belong to the class.

The results presented in Table 2 show that having more images per class with fewer number of classes results in features that perform very slightly better on PASCAL-DET, whereas for SUN-CLS, the performance is comparable across the two settings.

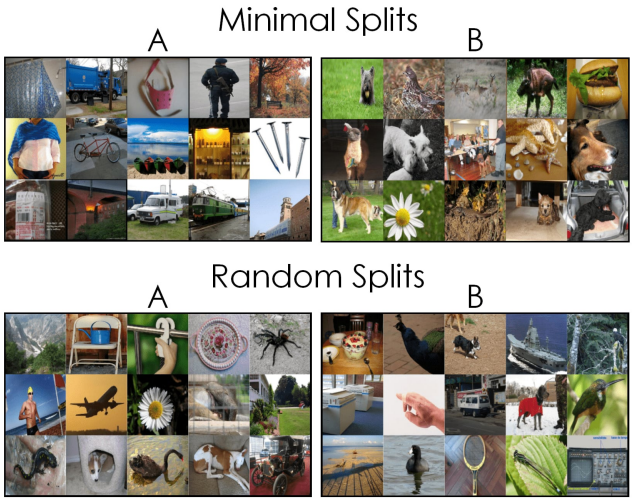

Figure 8: Visualization of the random and minimal splits used for testing - is adding more pre-training data always useful? The two minimal sets contain disparate sets of objects. The minimal split A and B consists mostly of inanimate objects and living things respectively. On the other hand, random splits contain semantically similar objects.

5.6. How important is to pre-train on classes that are also present in a target task?

It is natural to expect that higher correlation between pre-training and transfer tasks leads to better performance on a transfer task. This indeed has been shown to be true in [49]. One possible source of correlation between pre-training and transfer tasks are classes common to both tasks. In order to investigate how strong is the influence of these common classes, we ran an experiment where we removed all the classes from ImageNet that are contained in the PASCAL challenge. PASCAL has 20 classes, some of which map to more than one ImageNet class and thus, after applying this exclusion criterion we are only left with 771 ImageNet classes.

Table 3 compares the results on PASCAL-DET when the PASCAL-removed-ImageNet is used for pre-training against the original ImageNet and a baseline of pre-training on the Places [53] dataset. The PASCAL-removed-ImageNet achieves mAP of 57.8 (compared to 58.3 with the full ImageNet) indicating that training on ImageNet classes that are not present in PASCAL is sufficient to learn features that are also good for PASCAL classes.

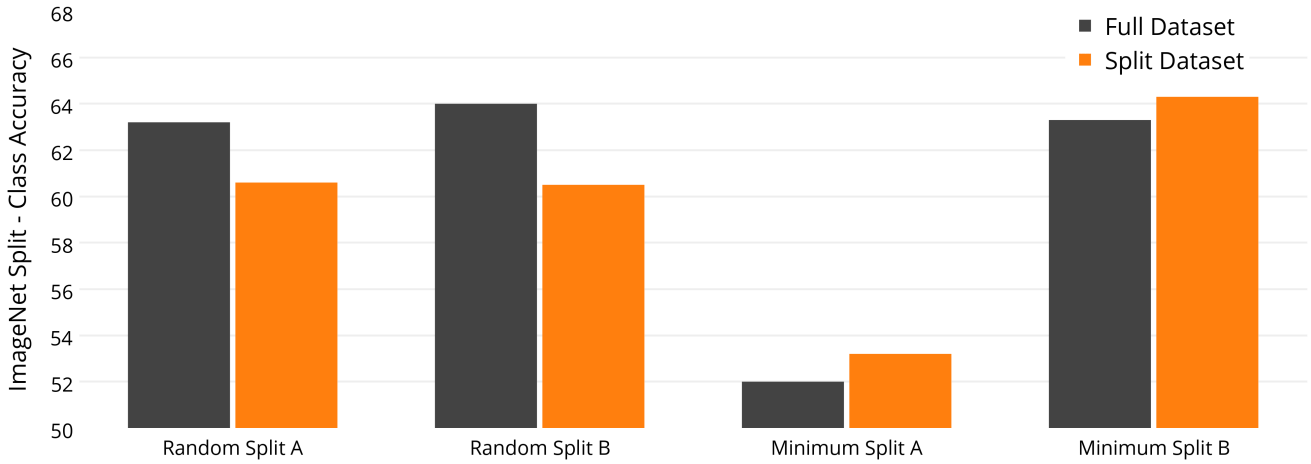

Figure 9: Does adding arbitrary classes to pre-training data always improve transfer performance? This question was tested by training two CNNs, one for classifying classes in split A and other for classifying classes in split A and B both. We then finetuned the CNN trained on both the splits on split A. If it is the case that adding more pre-training data helps, then performance of the CNN pre-trained on both the splits (black) should be higher than a CNN pre-trained on a single split (orange). For random splits, this indeed is the case, whereas for minimal splits adding more pre-training data hurts performance. This suggests, that additional pre-training data is useful only if it is correlated to the target task.

6. Does data augmentation from non-target classes always improve performance?

The analysis using PASCAL-removed ImageNet indicates that pre-training on non-PASCAL classes aids performance on PASCAL. This raises the question: is it always better to add pre-training data from additional classes that are not part of the target task? We investigated this question by first splitting ImageNet classes into two sets randomly: ran-split A and ran-split B (Figure 7). In order to determine if additional data helps performance for classes in split A, we pre-trained two CNNs – one for classifying all classes in split A and the other for classifying all classes in both split A and B (i.e. full dataset). We then finetuned the last layer of the network trained on the full dataset on split A only. If it is the case that additional data from split B helps performance on split A, then the CNN pre-trained with the full dataset should perform better than CNN pre-trained only on split A. Figure 9 shows the results of this experiment for both splits and confirms the intuition that additional data is indeed useful.

However, under a random class split within ImageNet, we are almost certain to have extremely similar classes (e.g. two different breeds of dogs) ending up on the different sides of the split. So, what we have shown so far is that we can improve performance on, say, husky classification by also training on poodles. But we can also ask: does adding arbitrary, unrelated classes, such as fire trucks, help dog classification?

To test this, we constructed the "minimal" ImageNet split. The classes in minimal split A do not share any common ancestor with minimal split B up until the nodes at depth 4 of the WordNet hierarchy (Figure 7). This ensures that any class in split A is sufficiently disjoint from split B. Split A has 522 classes and split B has 478 classes (N.B.: for consistency, random splits A and B also had the same number of classes). In order to intuitively understand the difference between min splits A and B, we have visualized a random sample of images in these splits in Figure 8. Min split A consists of mostly static images and min split B consists of living objects.

Contrary to the earlier observation, Figure 9 shows that both min split A and B performs better than the full dataset irrespective of whether we finetune only the last layer or all the layers. This result is quite surprising because it shows that, even when all layers are finetuned from a network pre-trained on the full dataset, it is not possible to match the performance of a network trained on just one split.

While it might be possible to recover performance with very clever adjustments of learning rates, current results suggest that training with data from unrelated classes may push the network into a local minimum from which it might be hard to find a better optima that can be obtained by training the network from scratch.

7. Discussion

In this work we analyzed factors that affect the quality of ImageNet pre-trained features for transfer learning. The results were quite surprising. For example, we have found that a significant reduction in the number classes or the number of images used in pre-training has only a modest effect

on transfer task performance. Indeed, we observed that the pre-trained features were surprisingly resilient to the various changes in the training data. While we do not have an explanation as to the cause of this resilience, we list some speculative possibilities that should inform further study of this topic:

- In our experiments, we investigated only one CNN architecture – AlexNet. While ImageNet-trained AlexNet features are currently the most popular starting point for fine-tuning on transfer tasks, there exist deeper architectures such as VGG [42] and ResNet [19]. It would be interesting to see if any of our findings are consistent when applied to these deeper networks. If so, it might suggest that AlexNet capacity is less than previously though.
- Our results might indicate that researchers have been overestimating the amount of data required for learning good general CNN features. If that is the case, it might suggest that CNN training is not as data-hungry as previously thought. It would also dash the hopes of beating ImageNet-trained features with unsupervised feature learning on a much bigger data corpus.
- Finally, it might be that the currently popular target tasks, such as PASCAL and SUN, are all too similar to the original ImageNet task to really test the generalisation ability of the learned features. Or that generalization should be tested with much less fine-tuning (e.g. one-shot-learning) or no fine-tuning at all (e.g. nearest neighbour in the learned feature space).

In conclusion: the answer to the titular question "what makes ImageNet good for transfer learning?" remains elusive for the time being. But the results of the experiments performed in the present study give rise to even more tantalizing questions. We hope that this work will pique our colleagues' curiosity and facilitate further research on this fascinating topic.

8. Acknowledgements

This work was supported in part by ONR MURI N00014-14-1-0671. We gratefully acknowledge NVIDIA corporation for the donation of K40 GPUs and access to the NVIDIA PSG cluster for this research. We would like to acknowledge the support from the Berkeley Vision and Learning Center (BVLC) and Berkeley DeepDrive (BDD). Minyoung Huh was partially supported by the Rose Hill Foundation.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 37–45, 2015.

[2] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *Computer Vision–ECCV 2014*, pages 329–344. Springer, 2014.

[3] M. Aubry and B. C. Russell. Understanding deep features with computer-generated imagery. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2875–2883, 2015.

[4] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 36–45, 2015.

[5] A. Bakry, M. Elhoseiny, T. El-Gaaly, and A. Elgammal. Digging deep into the layers of cnns: In search of how cnns achieve view invariance. *arXiv preprint arXiv:1508.01983*, 2015.

[6] Y. Bengio, A. C. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR, abs/1206.5538*, 1, 2012.

[7] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

[8] C. Buchheim, M. Jünger, and S. Leipert. Improving walker's algorithm to run in linear time. 2002.

[9] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. *arXiv preprint arXiv:1507.06550*, 2015.

[10] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. *arXiv preprint arXiv:1512.04412*, 2015.

[11] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1422–1430, 2015.

[12] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[13] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[14] A. Dosovitskiy and T. Brox. Inverting convolutional networks with convolutional networks. *arXiv preprint arXiv:1506.02753*, 2015.

[15] C. Fellbaum. Wordnet: An electronic lexical database. 1998.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[17] G. Gkioxari, R. Girshick, and J. Malik. Contextual action recognition with rcnn. 2015.

[18] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised feature learning from temporal data. *arXiv preprint arXiv:1504.02518*, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[20] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015.

[21] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

[22] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[23] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[24] P. Krähenbühl, C. Doersch, J. Donahue, and T. Darrell. Data-dependent initializations of convolutional neural networks. In *ICLR*, 2016.

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

[26] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[27] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[28] S. Mallat. Understanding deep convolutional networks. *arXiv preprint arXiv:1601.04920*, 2016.

[29] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009.

[30] M. Noroozi and F. Paolo. Unsupervised learning of visual representations by solving jigsaw puzzles. *arXiv preprint arXiv:1603.09246v2*, 2016.

[31] B. A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[32] A. Owens, P. Isola, J. McDermott, A. Torralba, E. Adelson, and F. William. Visually indicated sounds. In *CVPR*, 2016.

[33] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[34] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[35] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813, 2014.

[36] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[37] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[38] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009.

[39] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[40] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[41] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014.

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[43] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[44] J. Wang, Z. Zhang, V. Premachandran, and A. Yuille. Discovering internal representations from object-cnns using population encoding. *arXiv preprint arXiv:1511.06855*, 2015.

[45] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2015.

[46] D. Wei, B. Zhou, A. Torrabla, and W. Freeman. Understanding intra-class knowledge inside cnn. *arXiv preprint arXiv:1507.02379*, 2015.

[47] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[48] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002.

[49] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[50] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

[51] R. Zhang, P. Isola, and A. Efros. Colorful image colorization. In *ECCV*, 2016.

[52] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[53] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. *NIPS*, 2014.

Supplementary Material

1. Does additional pre-training data always improve performance? - Finetuning all layers

In Section 4 of the main paper, we investigated if additional pre-training data always improves performance. We presented results under the experimental paradigm where only the last layer of the network was finetuned. Another common practice while using pre-trained network is to finetune all the layers. We also finetuned in this way and the results are reported in Table 1. The obtained results follow the same trend as in the main paper. The most interesting bit is that even when all layers of a network pre-trained on minimal split A+B are finetuned on a single split, it performs worse than directly training on that split. This suggests that it might be the case, pre-training with A+B both, pushed the network parameters in a local minima from which it is not possible to recover a solution that is as good as starting from scratch.

2. Visualizing how training evolves over time

Does a CNN learn to distinguish coarse semantic classes prior to learning fine grain classes? We investigated this by visualizing FC7 nearest neighbors of randomly chosen images at different time steps during the training (see Figure 3). The visualization shows that even a randomly initialized network preserves some semantic structure. During training, the network learns to first perform matching based on color and texture and slowly hones into features that are required for more fine-grained recognition that is required to solve the ImageNet challenge. Figure 5 conveys similar information through t-SNE visualization. One thing worth noticing is the shift in the t-SNE embedding space from a almost connected sphere to disjoint connected components in the early phases of training.

3. Visualizing the effect of number of pre-training classes

Additional visualizations for results presented in Figure 5 and Section 5.3 of the main paper are shown in Figure 4.

4. Visualizing ImageNet

When we train for the leaf nodes of WordNet, how well do the learnt features perform on classifying the intermediate nodes? Ideally, we would like to visualize this by color coding all nodes of the WordNet tree by their accuracy. However, as the WordNet tree is quite imbalanced, visualizing the entire tree is tough. However, there are several graph visualization algorithms that have been developed to solve this problem. We used a linear time variant of the Reingold-Tilford algorithm [8] to visualize the accuracy of all WordNet nodes in Figure 2. A higher resolution figure is available on the project website.

Our work can be found on our website
http://minyounghuh.com/papers/analysis

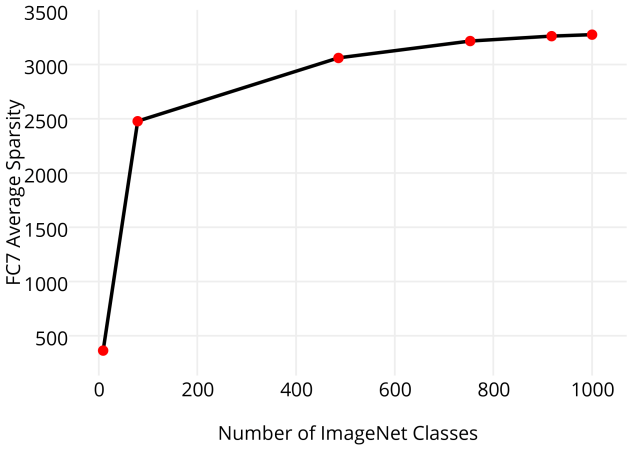

Figure 1: The average sparsity of FC7 features computed on the validation set. The average sparsity number represents the average number of zeros in the FC7 features. We have computed the average sparsity on various label sets constructed from the hierarchy experiment.

Finetune Data	Split	Full (All)	Full (Last)
Ran Split A	60.6	61.3	63.2
Ran Split B	60.5	62.7	64.0
Min Split A	53.2	52.1	52.0
Min Split B	64.3	64.1	63.3

Table 1: Results comparing the accuracy of finetuning all layers vs finetuing the last layer. The accuracy is computed on the experiment investigating - "Does adding additional pre-training data improve task performance?"

5. Sparsity

Around 80% sparsity in FC7 features of AlexNet trained on ImageNet provides an attractive compression attribute for data storage and fast image querying. Figure 1 shows the average FC7 feature sparsity when different number of pre-training classes are used. The results indicate that with the need of distinguising between finer classes, the features become more sparse.

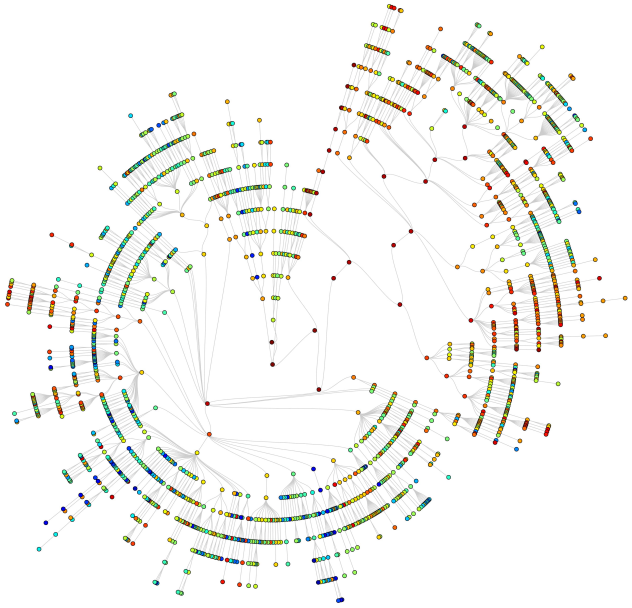

Figure 2: A visualization of the accuracy of classifying all nodes in the wordnet tree by features learnt by training only on the leaf nodes. A linear time variant of Reingold-Tilford algorithm [8] is used for visualization. Each node's accuracy is color coded by linearly mapping the accuracy to the jet colormap, where red and blue indicate high and low accuracies respectively. This figure is best viewed in color. For a higher resolution figure including the node labels refer to our website.

6. Computing coarse class classification accuracy

In Section 5.3 of the main paper, we discussed, "Does training with fine-grained classes induce features relevant for coarse recognition". In this section, we computed the difference between the accuracy of the coarse class and the average of sub-class accuracies. In order to account for the different number of examples in each coarse class, we normalized the accuracy in the following way: Let $\Phi(A)$ be the normalized accuracy for class A. Let x denote the ground truth class of an image and $\hat{x}$ denote the predicted class. Then $\Phi(A)$ is defined as below:

$$\Phi(A) = \frac{P(\hat{x} \in A | x \in A) + P(\hat{x} \notin A | x \notin A)}{2}$$

7. List of classes 127 classes

In Section 5.1, we generated different label sets based on the hierarchy of the WordNet tree. The list below shows all the classes used in the 127 label set.

n02856463 board
n02638596 ganoid
n04235291 sled
n01428580 soft-finned fish
n10401829 participant
n04500060 turner
n02512938 food fish
n03664943 ligament
n03446832 golf equipment
n03764276 military vehicle
n03122748 covering
n06874019 light
n03619396 kit
n04128837 sailing vessel
n03528263 home appliance
n04100174 rod
n03880531 pan
n02924116 bus
n01693783 chameleon
n04015204 protective garment
n01482330 shark
n06793231 sign
n01629276 salamander
n13134947 fruit
n02954340 cap
n09820263 athlete
n09214060 bar
n01687665 agamid
n03476083 hairpiece
n01698434 alligator
n03241093 drill rig
n03151500 cushion
n01703569 ceratopsian
n01861778 mammal
n03450516 gown
n01495701 ray
n03678362 litter
n04571292 weight
n03597469 jewelry
n04077734 rescue equipment
n03294833 eraser
n01909422 coelenterate
n07929519 coffee
n04285622 sports implement
n03497657 hat
n07930554 punch
n01692864 lacertid lizard
n07683786 loaf of bread
n03039947 cleaning implement
n04447443 toiletry
n02642644 scorpaenid
n10019552 diver
n01726692 snake
n02942699 camera
n04317420 stick
n01767661 arthropod
n01674990 gecko
n03309808 fabric
n03257586 duplicator
n07829412 sauce
n01922303 worm
n01940736 mollusk
n07882497 concoction
n02605316 butterfly fish
n07891726 wine
n06595351 magazine
n09287968 geological formation
n07681926 cracker
n03540267 hosiery
n01691951 venomous lizard
n02606384 damselfish
n03513137 helmet
n03510583 heavier-than-air craft
n07557434 dish
n07707451 vegetable
n02858304 boat
n01689411 anguid lizard
n03964744 plaything
n01503061 bird
n15074962 tissue
n12992868 fungus
n04099429 rocket
n07582609 dip
n03472232 gymnastic apparatus
n03414162 game equipment
n03035510 cistern
n03236735 dress
n01662784 turtle
n03419014 garment
n03613592 key
n04509592 uniform
n07612996 pudding
n01697178 crocodile
n03405725 furniture
n11669921 flower
n02316707 echinoderm
n07680932 bun
n03278248 electronic equipment
n03896233 passenger train
n04125853 safety belt
n03906997 pen
n04341686 structure
n07579575 entree
n03825080 nightwear
n01685439 teiid lizard
n03990474 pot
n07560652 fare
n04264914 spacecraft
n01676755 iguanid
n03441112 glove
n07800740 fodder
n02807260 bath linen
n07611358 frozen dessert
n04377057 system
n03101156 cooker
n03666917 lighter-than-air craft
n03094503 container
n03183080 device
n04451818 tool
n09289709 globule
n03837422 oar
n04185071 sharpener
n04194289 ship
n03961939 platform
n01694709 monitor
n02652668 plectognath
n01639765 frog

A complete list of classes used in the experiment studying the effect of the number of pre-training classes can be found on our website.

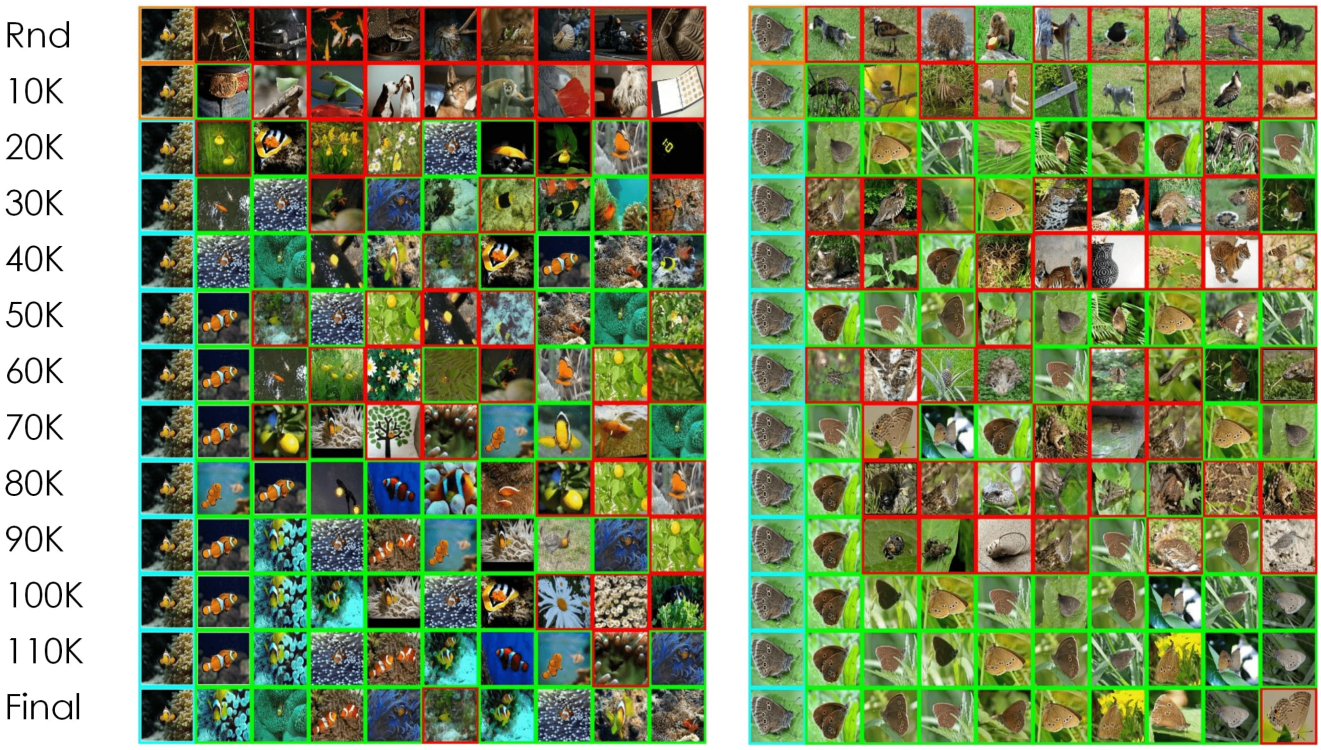

Figure 3: The evolution of the FC7 nearest neighbors as the network is being trained. Each row represents the amount of iterations trained before computing the nearest neighbors. The left most column is the target image and nearest neighbors span from left to right corresponding to closest to furthest. For the left most column, orange indicates that the network has incorrectly classified the image and blue indicates it has classified correctly. For the remaining columns, green indicates that the nearest neighbor is correct and red indicates that the nearest neighbor is incorrect. The figure shows that color and pattern are learned prior to learning the coarse label. The fine labeling is concretely learnt around 100k iterations. The figure is best viewed in color

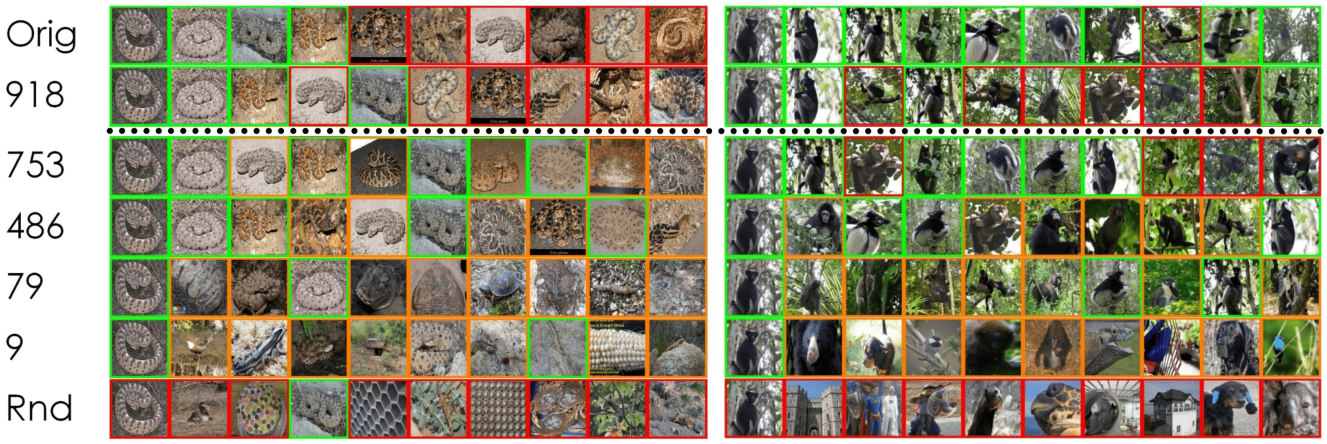

Figure 4: Additional results of nearest neighbors of FC7 features from correctly classified images. The image on the left is a horned rattlesnake and the image on the right is a sloth bear. Each row shows feature embeddings trained with different number of classes (from fine to coarse). The row(s) above the dotted line indicate that the image class was one of the training classes, whereas in rows below the image class was not present in the training set. Images in green indicate that the NN image belongs to the correct fine class (one of 1000); orange indicates the correct coarse class (based on the WordNet hierarchy) but incorrect fine class; red indicated incorrect coarse class. All green images below the dotted line indicate instances of correct fine-grain nearest neighbor retrieval for features that were never trained on that class.

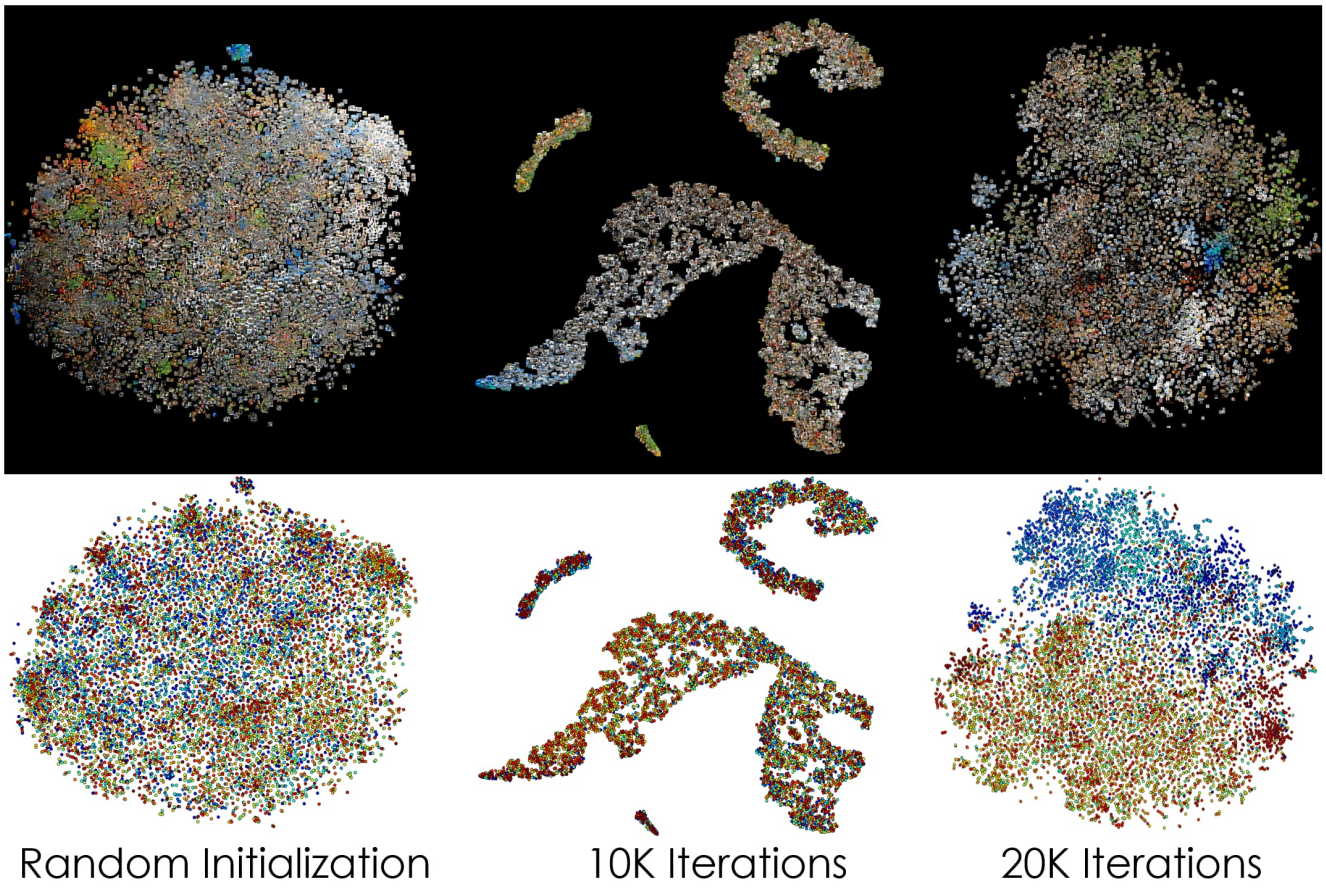

Figure 5: t-SNE visualization on the validation set using weights from random initialization, 10K and 20K iterations. The t-SNE visualization is run using 10 random samples per class. The figure on the top represents images embedded on the actual t-SNE embedding. The figure on the bottom shows the t-SNE embedding color coded by linearly mapping ImageNet class labels, as originally provided by [37], to the jet colormap.