How effective is Transfer Learning method for image classification

Marek Dąbrowski,
Tomasz Michalik
Orange Polska, Centrum
Badawczo-Rozwojowe, ul.
Obrzeżna 7, 02-691 Warszawa
Email: {marek.dabrowski,
tomasz.michalik}@orange.com

Abstract—**This paper deals with re-training neural network-based image classification model, using so-called Transfer Learning approach. This method allows for creating a new image classifier, reusing pre-trained weights from a publicly available model. Our study gives some insight on accuracy of re-trained models and provides guidelines concerning required number of training examples. Presented results may be useful for computer vision practitioners, who would like to adapt results of state-of-the-art research on neural networks for their own customized image recognition models.**

I. Introduction

MACHINE learning and neural networks dominate in recent research on computer vision. Deep learning and convolutional neural networks [1][2] proved to be especially successful in solving various computer vision problems, including image classification, segmentation, object detection etc.

The goal of image classification is to guess the object presented in the picture, from a set of pre-defined labels, or classes. Convolutional neural networks have been used for image classification since pioneering research by Yann LeCun in 1980ies [3]. More recently, since the work of A. Krizhevsky [4] the same basic concepts are applied to classification of colorful images (photos).

The general idea of deep neural network for image classification is depicted in Fig.1. Input data consists of numerical values for RGB color intensities of image pixels. The input is processed by artificial neural units, structured in multiple interconnected layers. The internal units may be of different types: convolutional, pooling and fully-connected, usually intervened with each other according to some complicated model architecture. The last layer is the classification layer, which calculates output probabilities. In the case of image classification problem, these probabilities represent likelihood that the image belongs to given pre-defined class. For example, the output layer may calculate likelihood that the input image depicts a "cat", or a "dog", and so on.

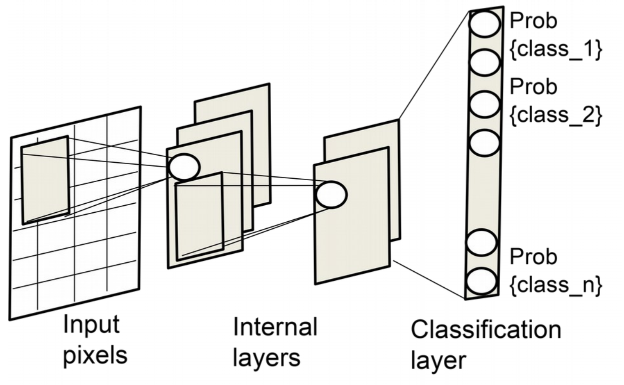


Fig. 1 Illustration of multi-layer convolutional model for image classification

Recent progress in research on image recognition has been enabled, among others, by increased availability of high quality training data. The Imagenet repository [5] is freely available for research and contains about 14 million human-annotated images, grouped in 21 thousand classes. In an annual contest organized by Imagenet, researchers compete to achieve best results in typical computer vision problems. For the purpose of this contest, 1000 classes have been specially selected, corresponding to a wide overview of objects that may typically occur in real-life images. These 1000 classes (each with about 1000 training images) constitute a special "Imagenet1000" corpus, which is often treated in research [6].

It is not uncommon for top research teams to publish publicly not only the research results, but also pre-trained models of neural networks that have been successful in Imagenet contest. These publicly available pre-trained models can be freely used for research and even commercial purposes. Notable example is the "GoogleNet" model (see Fig.2) proposed by Google in 2014 [7][8][9], improved in consecutive years, and made publicly available together with open source computation toolkit Tensorflow [10].

Our research on image classification has been motivated by a concept of photo album service for home users, where submitted photos would be automatically tagged with semantic information about depicted objects.

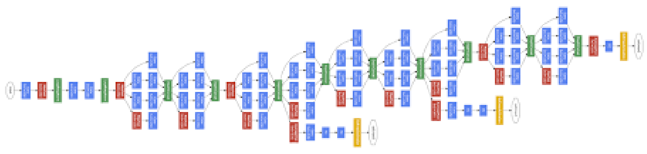


Fig. 2 "GoogleNet": state-of-the-art multi-layer convolutional model for image classification

The simplest way to implement such service is to use open source computation framework, with standard publicly available image classification model pre-trained on Imagenet1000 corpus. Such model would categorize among various kinds of standardized categories, which include among others a type of place where photo was taken (e.g. "lighthouse", "shop"), type of depicted animal ("Egyptian cat", "English foxhound"), or type of clothing ("T-shirt", "blue jeans"). Such approach has some disadvantages for practical deployment. On one hand, the "one-size-fits-all" approach is too generic if the foreseen service should focus on one particular applicative area, e.g. to recognize only geographical places, or types of animals. On the other hand, the Imagenet1000 categories are over-specified in some selected fields, like for example they contain 120 dog breeds to recognize among. Such level of detail is usually not necessary.

Thus, aiming to build a specific service based on image recognition technology, we would rather create our own custom image classifier. This paper focuses on methods how we can do that with minimum effort.

II. Transfer learning

It is known that training image classification from scratch is a very long and difficult process and may take weeks to complete on high-performance hardware [7]. Thus, a Transfer Learning method is often proposed to simplify it [12]. It assumes that a new classification layer is learned, while all weights of internal layers (remind that for example the GoogleNet model has 22 of them) are transferred from a pre-trained model. The process for practical usage of Transfer Learning is the following:

1. Get a previously trained model, e.g. from [11].
2. Split the old model architecture into two parts:
   a. All hidden layers of the neural network, with their structure (connections) and previously learned weights, will be copied into the new model.
   b. The last layer of neural network, which performs actual classification into one of the classes, is strictly related with the old model and will be disregarded in the new model.
3. Prepare a new set of training examples (images labelled with appropriate class name, as required for the new model).
4. For a new set of training images, calculate the output values after passing through the first part of neural network (the one that is transferred into the new model). The numerical value calculated as output of next-to-last layer of original model for a given image, will be called a "bottleneck".
5. Add a new final fully-connected layer, which will now constitute the last layer of new neural network model. This new final layer will calculate the probability of given image belonging to a given class.
6. Train the new final layer with previously calculated "bottlenecks" as input, and a set of new "ground truth" labels that denote true classes of training images.

Thanks to that method, we can create a new image classification model with our own classes and labels, within several hours instead of weeks, on standard hardware.

An example neural network architecture with new re-trained classifier is presented in Fig.3. Remark that the new classifier may contain some classes that overlap with the old classifier (we will call them *"internal"* classes, shortly INT) and some classes that are totally new (*"external"* classes, EXT). Only in the case of EXT classes we can truly claim using the Transfer Learning method, since the training images belonging to them have not been previously used for training internal weights of deep neural network. In the case of INT classes, their training images may have been previously used in training the internal features, so we cannot really speak of Transfer Learning. However, the INT classes will be discussed in our experiments since in practice they may be equally useful in creating a service-oriented custom classifier.

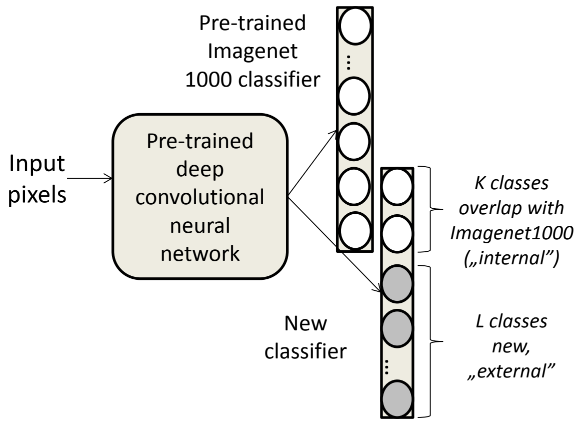

Pre-trained Imagenet 1000 classifier
Input pixels
Pre-trained deep convolutional neural network
K classes overlap with Imagenet1000 („internal")
L classes new, „external"
New classifier

Fig. 3. Transfer learning setup: new classifier on top of pre-trained deep convolutional network

Fig.4 gives additional explanation of the distinction between "external" and "internal" classes. The left column is an example excerpt from list of class labels in standard Imagenet1000 model. The right column presents a fictional classifier with 6 custom classes, among which 3 overlap with Imagenet1000 (INT) and 3 are outside of the Imagenet1000 set (EXT).

No	Imagenet1000 classifier	Custom classifier with 6 classes
1	Bakery ---	--> Bakery (INT)
2	Barn ---	--> Barn (INT)
3	Lighthouse ---	--> Lighthouse(INT)
4	Electric ray	Embankment (EXT)
5	English foxhound	Embassy (EXT)
6	English setter	Farm building (EXT)
7	Eggnog	
8	Egyptian cat	
…	…	

Fig. 4. Example fictional custom classifier with 6 classes. The 3 INT classes in custom classifier are the same as Imagenet1000 model, while 3 EXT classes are new.

The Transfer Learning method has been previously studied in scientific literature, e.g. in [12] and its practical usage is not an original idea. The software scripts for modifying the standard Imagenet1000 model are available with popular neural network computation frameworks, like Tensorflow [11]. In previous work [13] we have studied the Transfer Learning method, focusing on practical guidelines for setting hyperparameters of re-training process, as well as evaluation of re-training speed on several typical hardware platforms. What we found missing in the literature is, however, a comprehensive evaluation of performance of re-trained models. In this paper we would like to share our experience in this regard, in particular trying to answer the following questions:

- What accuracy we can expect from re-trained model? We know that the publicly available GoogleNet model for Imagenet1000 classification can achieve about 80% accuracy (meaning that it gives correct result on 80% of test images). We can expect that the re-trained model could be less accurate, since the internal features were effectively trained on different set of images than the final classifier. But how can we quantify the loss of accuracy?
- How many training examples do we actually need in re-trained classes to achieve acceptable accuracy of end-user service?

III. ACCURACY OF RETRAINED MODEL

First goal of our research was to assess accuracy of a re-trained image recognition model. For our experiments we have used Tensorflow [10], open source software library from Google. It performs machine learning computations, including neural network models, on various types of hardware, with CPU and GPU. Tensorflow implements the Transfer Learning method, with "GoogleNet" model as basis for re-training. We made our experiments on a typical desktop PC with 4 CPU cores and 8GB RAM, equipped with GPU card NVIDIA GeForce GTX 960.

As a target model for Transfer Learning we have defined a custom set of 100 classes, manually and arbitrarily selected from wide Imagenet corpus, which has 21thousand classes in total. For each new class we have verified if it is, or it is not, included in the standard Imagenet1000 model. Referring again to the example presented in Fig 4, the class "bakery" is included in Imagenet 1000 and the original GoogleNet model has been trained with examples belonging to this class. This class is thus considered "internal" (INT) class, in contrary to, for example, "embassy" which is a new class, named "external" (EXT).

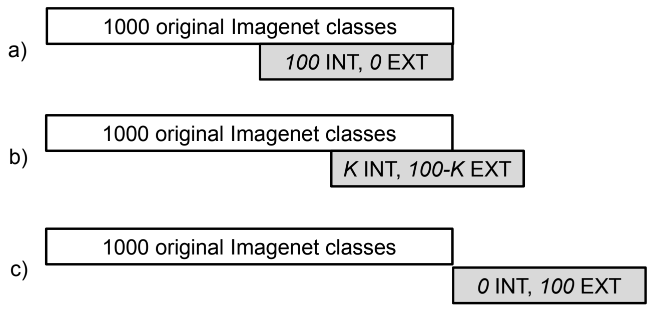

a)
1000 original Imagenet classes
100 INT, *0* EXT
b)
1000 original Imagenet classes
K INT, *100-K* EXT
c)
1000 original Imagenet classes
0 INT, *100* EXT

Fig. 5. Various mix of INT and EXT classes in the custom classifier with 100 classes: a) full overlap, b) partial overlap, c) no overlap

Depending on particular service scenario, the target custom classifier may contain certain number of INT and EXT classes. Thus, in our experiments we have varied the mix of their mutual proportions (see Fig.5).

Intuitively, we may expect that the INT classes may be more accurately recognized by the new re-trained classifier, since they are somehow "known" by the model from the beginning. On the other hand, we may expect that the EXT classes are less accurately recognized by the new classifier, since they were not taken into account in prior training process of internal layers. Thus, we were interested in studying how the overall accuracy depends on the mix of "new" (EXT) and "old" (INT) classes, which reflects somehow the level of similarity between the "old" and the "new" model.

We have used the re-training procedure as described in our previous work [13]. The classifier layer has been trained with Stochastic Gradient Descent algorithm [1], which takes the forward propagation loss (cross-entropy loss), calculates the gradients in backward propagation and then changes the weights of the model trying to minimize the loss. The learning rate value tells how fast the optimizer should converge to minimal loss. Based on [13], the chosen parameter setting was as follows: training steps=10000, the type of optimization algorithm was Adam Optimizer with learning rate α=0.01 and epsilon=0.1 (a small constant for numerical stability), train batch size=100. For each re-trained class we have downloaded about 1000 training images from the Imagenet corpus.

The accuracy metric has been computed on a test set, created by putting-off 10% of images from the overall corpus. We have used two typical test metrics. *Top-1* *accuracy* is defined as ratio of correct classification results in the entire test set (where correct result means: "the label with maximum score is equal to ground truth"). The *Top-5 accuracy* metric is less restrictive and defined as ratio of correct classification results in the test set, but the correct result is now: "the ground truth label is among 5 maximum-score labels assigned by the classification algorithm".

We have evaluated separately the accuracy metrics for the sets of INT and EXT classes. In fact, we have measured "per-class" accuracy, that is a separate metric value for each of 100 classes in the re-trained model. Then, we define the "INT Top-1" accuracy as the average among all "internal" classes. Respectively, the "EXT Top-1" accuracy is defined as average accuracy among the "external" classes. As mentioned earlier, we expect EXT accuracy to be lower than INT accuracy, since it covers classes that had not been known in prior pre-training process.

To mitigate impact of arbitrary choice of set of target classes, we have repeated each experiment (i.e. performed re-training and measured resulting accuracy) five times, assuming different set of target classes, while keeping the same proportion of EXT and INT. The result is reported as an average with 5% confidence intervals.

Fig. 6 presents results of accuracy of re-trained models with different mix of EXT and INT classes. Starting from the left, "100% INT" means that all classes of the re-trained model overlapped with the original Imagenet1000. Then, we have gradually introduced more and more external classes. On the extreme right hand-side, all classes of the re-trained model are "new", meaning that the scenario is a full Transfer Learning.

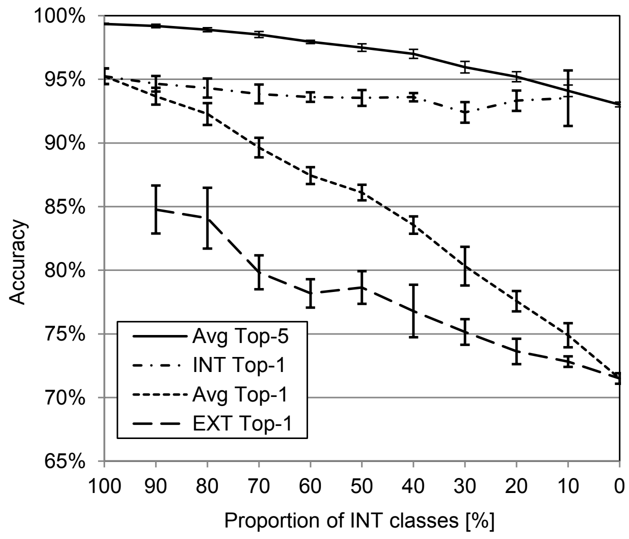


Fig. 6. Accuracy of re-trained 100-classes model with different proportions of INT and EXT classes

First, let us have a look at the "Avg Top-1" line in Fig.6. It depicts overall quality of the re-trained model, not knowing how different types of classes perform. So, in the case of 100% internal classes, the Top-1 accuracy is about 95%, which is very high value (almost all images in test corpus are properly classified). Considering that all the classes overlap with the original Imagent1000 model, we can say that re-training the classification layer does not degrade the performance (which is not so surprising since all the images used for re-training have been previously used to train the internal features of the model).

Going to the right, however, the accuracy drops, since more and more classes and training images are "new", not used before for training the internal features. At the extreme case, where 100% classes are new, we note Top-1 accuracy of 71,5%, which means that about 3 in 4 test images are still properly recognized. The "Top-5" accuracy metric shows that this less restrictive metric drops from 99% to 93% in the Transfer Learning scenario.

A Top-1 accuracy drop from 95% to 71% is noticeable, but perhaps acceptable, taking into account the easiness and low computational cost of obtaining the re-trained model, comparing to training it from scratch.

The lines marked as "INT Top-1" and "EXT Top-1" give us a little more insight into performance of "old" and "new" classes separately. The "INT" classes that were present in original model and remained in the re-trained one, keep the over-90% accuracy. The internal features of original model were actually trained on them, so we could expect that re-training does not significantly impact these classes. For EXT classes, we observe that their Top-1 accuracy is between 85% and 71% throughout tested range of classes mix.

To confirm the outcome of that experiment, we have repeated it in a scenario with 200-classes classifier, instead of 100 classes. The results are depicted in Fig.7.

Since this experiment was just a confirmation of previous one, this time we did not repeat it 5 times with different sets of target classes, as we did for the 100-classes model. For this reason the curves depicted in Fig.7 display more variability, caused by randomness in choice of classes in each of the points. Previously this variability was smoothed out by taking the average of 5 repetitions of each experiment.

The conclusions of the 100-classes scenario hold in 200-classes case, with the restriction that absolute values of accuracy of re-trained models are now a bit lower, reaching 65% in the case of all-EXT classes (comparing to 71% in 100-classes scenario).

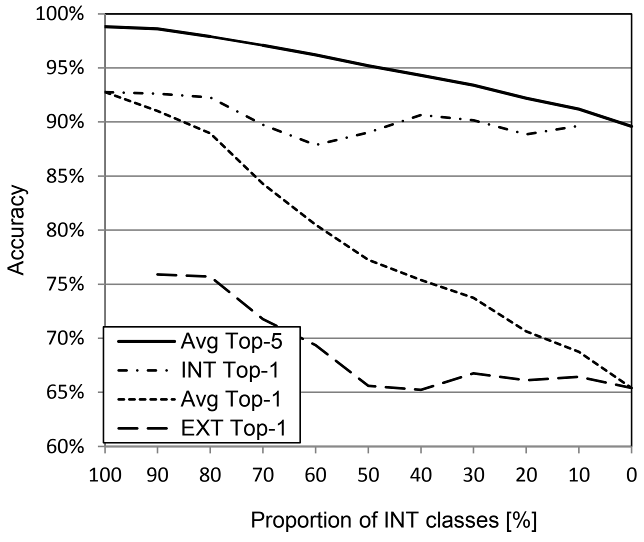


Fig. 7. Accuracy of re-trained 200-classes model with different proportion of INT and EXT classes

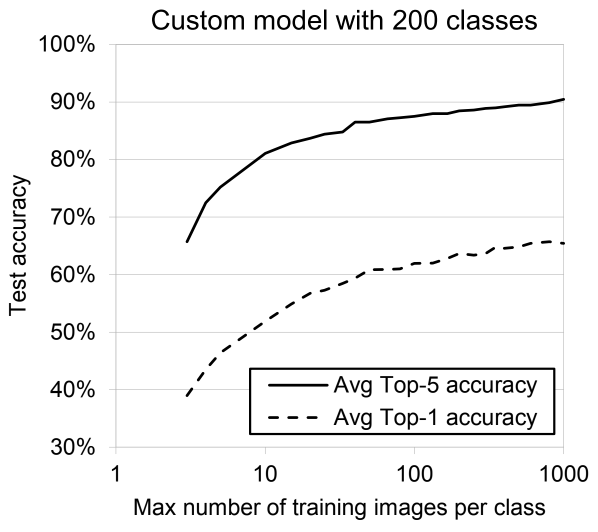
IV. Required number of training examples

An application-oriented researcher or software developer why wants to train his custom image classifier will certainly ask this important question: "How many training examples do I need to re-train the model with sufficient accuracy?". Probably the more examples the better, but from practical perspective, large number of training examples may not be available, or may be very costly to obtain. Thus, training with limited corpus could be of interest, if the quality (accuracy) of resulting model is sufficiently good.

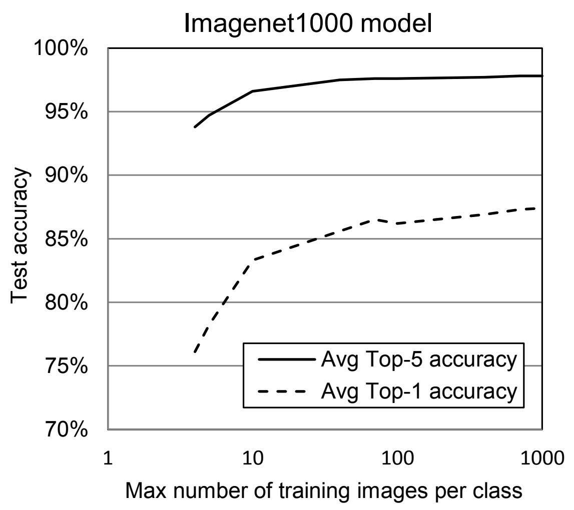
Aiming to answer this question we made another re-training experiment on the previously discussed 200-classes custom model, with all classes being of EXT type. However, at this time we have limited the number of training examples available in each class, starting from as few as only 3 training examples per class. For a model re-trained in such way we have measured the Top-1 and Top-5 accuracy on a test set. Then, we gradually increased the number of available training examples, up to maximum value that was about 1000 for each class (remark that for some classes Imagenet has less than 1000 examples, so in this case we have used the maximum number available).

The results, presented in Fig.8a, are a bit surprising. Clearly, the more training examples we have used, the better is the accuracy. But, apparently, with only 3 training examples per class, the Top-1 metric reaches almost 40%, and Top-5 is above 60%. This result shows that the re-trained neural network is able to extract generic visual features from images, being pre-trained on a standard image corpus. Then, having just a few training examples is sufficient to tell the model how to classify images into classes, even if the set of classes is totally different than in the original standard model.

To confirm this result, we have made similar experiment with the "standard" set of classes as defied in Imagenet1000. The achieved accuracy (see Fig.8b) is now even better, which is explained by the fact that all classes are of INT type, thus the discussed test case is not a real Transfer Learning scenario.



a)



b)

Fig. 8. Accuracy of a) custom 200-classes model, and b) Imagenet1000 model, re-trained with different number of training images per class

The reported accuracy values are averaged over all 200, or 1000 classes of a model. For this reason we don't really know how particular classes perform. Their recognition capability may differ, depending on characteristics of particular class. For example, we may expect that telling the difference between "subway train" and "train" may be more tricky than between "train" and "cat". This intuitively understandable differences should somehow be reflected in results of accuracy metric measured per-class.

In Fig.9 we present histogram of such per-class accuracy, in 10%-wide bins. The test was made for three different numbers of training images available per class: 3, 33 and 1000.

In the case of 1000 training images (maximum what is available) we can see that for the highest number of classes the measured Top-1 accuracy reaches between 80 an 90%. Still, there is a number of classes which seem to be problematic for image recognition model. There is 1 class with accuracy of 0%, 1 in 0-10% bin, and 4 classes in the range 10-20%.

When we limit the number of training images to 3, we can see more classes that perform poorly after re-training. Now, there are 10 classes with accuracy 0%, 26 with 0-10%, and the maximum, 31 classes, fall in the bin of 30-40%. But still, we can find 9 classes where accuracy reaches 90-100%. Clearly, there is a big discrepancy between different classes and reporting only the all-class average accuracy may be quite misleading if we want to look at one particular class.

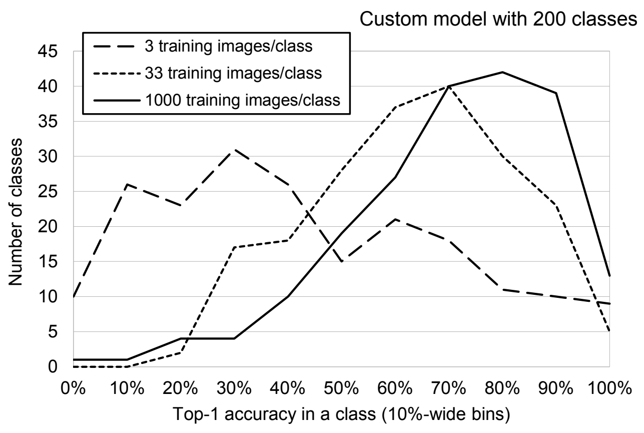


Fig. 9. Histogram of per-class accuracy in re-trained model with 200 custom classes

To look more deeply at this characteristic, we have produced a chart similar to Fig.8, but now we have divided our set of 200 target classes into several sub-sets, each with 40 classes that achieve similar per-class performance. Fig. 9 depicts the average accuracy for 3 sub-sets of classes, labeled as "easy", "moderate" and "difficult" (referring to difficulty of classifying images of given class). Remark that the accuracy values on the figure are normalized to maximum that can be achieved for given class, because we wanted to ephasize relative differences between classes, rather than absolute values of accuracy.

We can see that the "easy" set achieves almost 90% of its maxiumum accuracy already with 3 training examples per class. This "easy" set contains among others such classes like:

- *Motorcycling*
- *Van, caravan*
- *Cruiser, police cruiser, patrol car*
- *Sawmill*
- *Campsite, campground, camping site*

Intuitively, these classes represent quite concrete and well-defined objects – we may easily imagine how a "motorcycle" or "van" or "sawmill" looks like. Images that belong to these classes seem to be quite "easy" for the recognizer to distinguish and so only a few training examples are enough to re-train a high quality model from the standadr Imagnet1000 classifier.

The "moderate" classes achieve 50% of their maximum accuracy with only 3 training examples. The "moderate" classes are, among others:

- *Musical instrument, instrument*
- *Surveillance system*
- *Public toilet, comfort station*
- *Florist, florist shop, flower store*
- *Chairlift, chair lift*

Comparing to the first set of classes, we can intuitively see that the shape and look of "musical instrument" or "surveillance system" is not so obvious. Then, the "difficult" set achieves about 34% of its maximum accuracy with 3 training examples and it includes such classes like:

- *Plant, works, industrial plant*
- *Plaza, mall, center, shopping mall*
- *Outcrop, outcropping, rock outcrop*
- *Display window, shop window*
- *Shop, store*

A typical appearance of "shop" or "indiustrial plant" is not obvious at all, so difficulties in recognizing them are intuitively understandable.

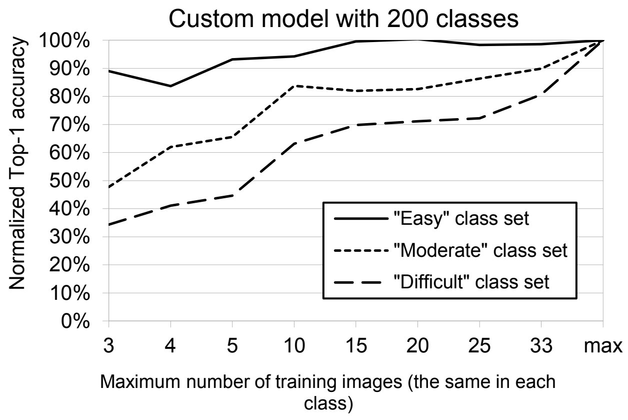


Fig. 10. Normalized accuracy vs. number of training images, for three groups of classes differing by "difficulty" of recognition

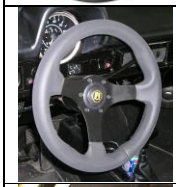
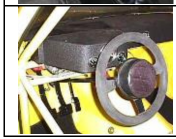
V. Custom classifier with minimum number of training examples

In this experiment we focus in detail on one specific class of retrained classifier. The goal was to verify that it can indeed be trained with just as few as 3 training examples, which seemed to us unbelievable at the first time. We have added a new class, named *"steering wheel"*, to our custom 200-classes classifier. Such class exists in the wide Imagenet corpus, but it is not part of Imagenet1000 (so it is of EXT type).

To train the new “steering wheel” class we have decided not to use the images from Imagenet website. To make sure that we have complete control over what kind of images are used for training, we have made ourselves 3 photos of a car steering wheels, as presented in Fig.11a. These three images were used as training examples for re-training the classifier. Then, we have used 10 random images from Imagenet “steering wheel” category as a mini-test set. Fig.11b shows 4 of these test images, with final classification results. We can see that in 3 cases the class “steering wheel” indeed has the highest probability, and in 1 case it is on 3rd place in classification results.

In this simple experiment we were able to create a custom classifier to recognize “steering wheel”, and train it with just a few images. Of course, the “steering wheel” class may be considered as rather “easy” test case, since the rounded shape is visually quite characteristic. Nevertheless, the fact that in some cases just a few examples are sufficient to re-train and use a reasonably “good” model, can be of great value for potential practical deployments.

a)

	1.steering wheel (0.06)
	1. surveillance system (0.09) 2. siren (0.04) **3. steering wheel, wheel (0.04)**
	**1. steering wheel (0,88)**
	**1. steering wheel (0,31)**

b)

Fig. 11. Experiment with new custom class “steering wheel”: a) training examples prepared by authors, b) testing images from Imagenet

VI. Summary and Next Steps

This paper has discussed Transfer Learning method for re-training image classification models based on neural networks. Following our previous work [13], which focused on guidelines for tuning the re-training process, now we have experimentally studied performance of re-trained models, and thus the boundaries for their practical applicability.

We have shown that the re-trained model may achieve accuracy of about 70%, comparing to 90% of a fully-trained model. We think, however, that such degradation may be acceptable in some service deployments, where cost of full-scale training could be excessive. Furthermore, we have shown that the number of images required for re-training is not so big, and in the case of some “easy” classes, it could be as few as less than 10 images.

We identify several research directions as next steps:

- Continue experiments with limited number of training examples, possibly with more “difficult” classes.
- Put more attention to theoretical understanding of causes of obtained experimental results.
- Systematically compare computational cost of re-training vs. full-scale training.

We may conclude that the Transfer Learning method may be effectively used to create custom-built image classification models on top of publicly available standard ones, in a short time and with moderate cost.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, “Deep Learning”, Book in preparation for MIT Press, 2016, on-line version available at:http://www.deeplearningbook.org

[2] Michael A.Nielsen, “Neural Networks and Deep Learning”, Determination Press, 2015, on-line version of the book available at: http://neuralnetworksanddeeplearning.com/index.html

[3] LeCun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D.,Howard, R. E., and Hubbard, W.. Handwritten digit recognition: Applications of neural network chips and automatic learning. IEEE Communications Magazine, 27(11), 1989

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In NIPS 2012, Neural Information Processing Systems, Nevada, 2012

[5] ImageNet database of computer images: http://image-net.org/

[6] Li Fei-Fei et al. ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision, 2015.

[7] Ch.Szegedy et al, “Going deeper with convolutions”, http://arxiv.org/abs/1409.4842

[8] Ch.Szegedy et al, Rethinking the Inception Architecture for Computer Vision, https://arxiv.org/abs/1512.00567

[9] Ch.Szegedy et al, Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, https://arxiv.org/abs/1602.07261

[10] M.Abadi et al, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[11] Publicly available pre-trained GoogleNet model: https://github.com/tensorflow/models/tree/master/inception

[12] Yosinski J, Clune J, Bengio Y, and Lipson H. How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems 27 (NIPS ’14), NIPS Foundation, 2014

[13] M.Dąbrowski, J.Gromada, T.Michalik, A practical study of neural network-based image classification model trained with transfer learning method, Position Paper of FedCSIS AIMaViG 2016, Gdańsk, September 2016, DOI: http://dx.doi.org/10.15439/2016F211