Fine-tuning Deep Neural Networks in Continuous Learning Scenarios

Christoph Käding[1,2], Erik Rodner[1,2], Alexander Freytag[1,2], and Joachim Denzler[1,2]

[1]Computer Vision Group, Friedrich Schiller University Jena, Germany
[2]Michael Stifel Center Jena, Germany

Abstract. The revival of deep neural networks and the availability of ImageNet laid the foundation for recent success in highly complex recognition tasks. However, ImageNet does not cover *all* visual concepts of *all* possible application scenarios. Hence, application experts still record new data constantly and expect the data to be used upon its availability. In this paper, we follow this observation and apply the classical concept of fine-tuning deep neural networks to scenarios where data from known or completely new classes is continuously added. Besides a straightforward realization of continuous fine-tuning, we empirically analyze how computational burdens of training can be further reduced. Finally, we visualize how the network's attention maps evolve over time which allows for visually investigating what the network learned during continuous fine-tuning.

1 Introduction

"How would you train a deep neural network when new data from potentially new categories is continuously added to the training set?"

Machine learning and vision have significantly benefited from benchmarking on fixed datasets, since they allowed for comparison between algorithms and developed models [1–5]. However, our world is an environment which undergoes ongoing change. Instead, both the semantic space of object categories as well as the visual appearance of known categories are not fixed. To handle this, we humans are able to continuously learn and adapt our knowledge. Both aspects, fixed models and changing environments, are in contrast with each other. Therefore, incremental learning is an important field of research aiming at developing visual recognition systems that are able to deal with new data from known or even completely new classes by performing learning in a continuous fashion. Furthermore, it is an essential element for active learning [6] and active discovery [7] approaches, which strictly require continuously changing models. Incremental learning aspects have been studied for a great variety of different models *e.g.,* [8–10], but not so far for deep neural networks (DNN).

In addition to large and fixed datasets, the resurrection of DNNs lead to the latest innovation pitch in computer vision research. Besides training deep models from scratch, additional benefits have become apparent when looking at the common use of

This research was supported by grant DE 735/10-1 of the German Research Foundation (DFG).

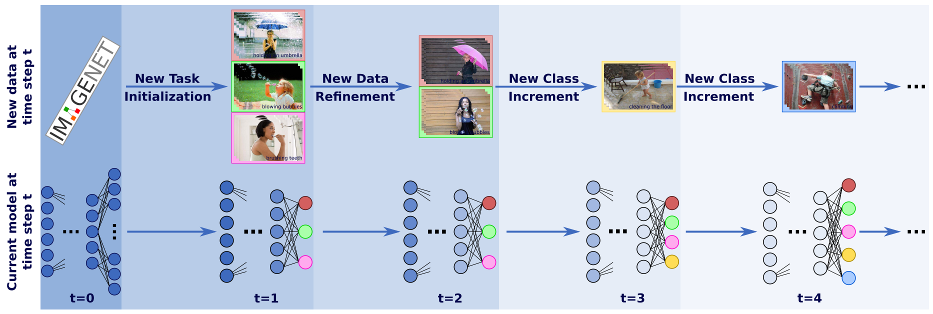


Fig. 1: At the beginning of the learning process, we have a pre-trained network as initialization for parameters. The first step is the classical fine-tuning where the final layer is replaced to fit the novel task (shown as colored dots). Upon the availability of more data, the relative importance of initial weight estimates is further reduced (indicated by reduced blue color). When even novel categories are discovered, also the network architecture needs to be adapted, *e.g.,* by adding new output variables.

fine-tuning to small datasets. Originally, fine-tuning has been referred to the process of pre-training neural networks with a generative objective followed by an additional training phase with a discriminative objective on the same dataset [11]. More recently, fine-tuning refers to re-using parameter values estimated on potentially large datasets as initialization in applications with limited access to labeled data. This approach has paved the way to significant performance gains in many applications, *e.g.,* [12–16].

From the perspective of continuous learning, the latter fine-tuning scenario can be seen as the extreme case of continuous learning which is restricted to only two time steps: pre-training and update.

More general forms of continuous learning for deep convolutional neural networks (CNN) have hardly been studied before. The question remains how continuous learning with a series of update steps can be performed robustly and efficiently. To study this question, we continuously fine-tune convolutional neural networks and empirically evaluate the effect of individual hyperparameters on the robustness of learning. A visualization of the process of continuous learning is given in Fig. 1.

The concept of continuous fine-tuning is general and applies to any form of a deep neural network. In this paper, we especially focus on image understanding scenarios and hence apply deep convolutional neural networks. A possible application scenario is active learning, where the updated model is immediately required for the sub-sequent selection [6]. Similarly, automated visual monitoring scenarios require efficient concepts for continuous learning when labeled data is incrementally provided by expensive but rather slow experts. In consequence, we specifically focus on scenarios where only little novel data is available in each update step but an updated model should be immediately available.

We provide empirical evidence for guidelines which show that the computation time for parameter updates in continuous learning scenarios can be significantly reduced. Although our empirical findings are intuitive, we believe that sharing our obtained insights is beneficial for a broader audience in several application areas. Furthermore, we investigate how CNNs evolve over time by visualizing how the attention of the nets shifts towards discriminative parts of newly added categories. This visualization allows for controlling and analyzing the continuous learning process.

2 Related Work

Since the necessity for dealing with continuous data streams is vast, update techniques for a variety of classification techniques have been proposed. While a complete overview is far beyond the scope of this paper, we briefly present examples for some well-established classification techniques.

Continuous Learning for Various Model Types The frequently used approach of *online learning* can be viewed as a special case of continuous learning. Here, the training dataset is not presented at once (as in batch learning) but with only a single example in every time step. Support vector machines (SVMs) have been the presumably most frequently used model in the last two decades. For SVMs, Cauwenberghs et al. presented online learning with fixed categories [17]. Based on these results, Tax et al. [9] introduced online learning of Support Vector Data Description (SVDD) models for novelty detection scenarios. Later on, incremental learning for the frequently used SVM solver SMO (sequential minimal optimization) with varying numbers of categories has been presented by Yeh and Darrell [18]. Gaussian Process (GP) models, which are closely related to SVMs, have been used for continuous learning of object categories by Freytag et al. [8]. Mensink et al. propose update techniques for nearest class mean (NCM) classifiers in [19] followed by Ristin et al., who present continuous extensions of a hybrid model consisting of NCM and RDF [20]. Sillito and Fisher derive update rules for GMM models in [10] which laid the foundation for the work of Hospedales et al. for the task of continuous class discovery [21]. Despite all reported benefits, these techniques have been rarely used recently due to their rather low model complexity compared to the one provided by deep neural networks. Let us therefore briefly investigate continuous learning approaches for deep nets.

Continuous Learning of Deep Neural Networks A decade ago, Wilson and Martinez compared batch learning and online learning of neural networks in [22]. According to their argumentation, online learning should be always preferred over batch learning, especially for large datasets. While the currently preferred mini-batch learning [23] is somewhat in-between, the authors of [22] found no practical advantage of learning with mini-batches over online learning. Similar arguments have been put forward by LeCun et al. [24]. However, online learning assumes the number of categories within the data stream to be known in advance. Furthermore, multiple cyclic passes (epochs) through the dataset are required for robust estimation of parameters.

For the more general problem of continuous learning, also novel categories have to be included into the existing network appropriately. One example is the work in [25] for training of deep networks with incrementally added categories. The authors approach the necessity of increased network capacity by duplicating the existing network, assigning available classes equally to one of both networks, fine-tune each net individually, and adding a third network on top to predict which net to use. Thus, this strategy can be seen as training a decision tree with a deep network in every node. Besides this relatively complex, memory-intensive, and computationally demanding approach, little is known so far how to continuously train deep networks efficiently.

Fine-tuning of Deep Neural Networks As mentioned before, fine-tuning of pre-trained networks to new tasks can be viewed as a special case of continuous learning with only two time steps: one initial learning step and one update step. A variety of publications underlines the benefits which arise from pre-training deep networks on large datasets. As an example, Agarwal et al. stated that *"pre-training significantly improves performance"* for the task of object recognition [12]. Similarly, Girshick et al. draw the conclusion that *"We conjecture that the 'supervised pre-training/domain-specific fine-tuning' paradigm will be highly effective for a variety of data-scarce vision problems."* [16]. Further benefits have been reported for image retrieval [13], semantic segmentation [14], fine-grained recognition [15], or object localization [26]. However, fine-tuning is only used with a fixed dataset of a new task. In this paper, we empirically investigate the ongoing process of "continuous fine-tuning" with increasing data and number of categories for improving the model of a single task.

Further Related Topics Among the previously shown works, there is a growing set of approaches which aim at replacing fixed models with systems that are able to adapt themselves continuously. A closely related research area is domain adaptation (*e.g.,* [27–29]). Different domains can either occur from different recording techniques (*e.g.,* a change of camera technology) or from a change of data distributions between two data collections. In contrast to our application scenario, domain adaptation techniques aim at estimating the differences between domain distributions to optimally leverage information between data collections. Instead, we take incoming data as-is and allow the network to adapt itself smoothly over time in case of occurring domain shifts.

Similarly related is the area of transfer learning [30, 29], which is also known as learning to learn [31]. In transfer learning scenarios, new categories are incrementally added to previously known data. The underlying assumption is that transferring model parameters from known but semantically related categories is beneficial to represent the novel category. Instead of explicitly finding support tasks or categories to transfer parameters, we rely on the learned representational power of a single network shared by all categories.

3 Deep Neural Network Learning in a Nutshell

Before we investigate continuous learning of deep neural networks, we briefly review the learning of these models and define hyperparameters used in our experiments.

Batch Learning of Deep Networks Let $f(\boldsymbol{x};\boldsymbol{\theta})$ be the output of a neural network with parameters $\boldsymbol{\theta}$ for a given image $\boldsymbol{x} \in \Omega$. Learning a network from a given labeled training set $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y}) = (\boldsymbol{x}_i, y_i)_{i=1}^{N} \subset (\Omega \times \mathcal{Y})$ boils down to minimizing a desired learning objective. Results of computational learning theory tell us that the objective should be comprised by the loss of the training set and a regularization term ω:

$$\bar{\mathcal{L}}(\boldsymbol{\theta};\mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f(\boldsymbol{x}_i;\boldsymbol{\theta}),y_i) + \omega(\boldsymbol{\theta}) \ . \tag{1}$$

In contrast to the overall training objective $\bar{\mathcal{L}}$, the loss term $\mathcal{L}$ operates on individual examples and compares the obtained with the desired model outputs. Common choices for $\mathcal{L}$ are the quadratic loss for regression tasks or the softmax loss for multi-class classification scenarios. The term ω is usually an elastic-net regularization [32] that combines L_2 and L_1-regularization of the parameters $\boldsymbol{\theta}$.

As in many other application domains, vision tasks require model functions of high complexity. Therefore, $f(\boldsymbol{x}_i;\boldsymbol{\theta})$ is commonly a composition of functions, usually referred to as layers, with their individual parameters:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = f_L\left(\ldots\left(f_2\left(f_1\left(\boldsymbol{x};\boldsymbol{\theta}_1\right);\boldsymbol{\theta}_2\right)\ldots\right);\boldsymbol{\theta}_L\right) \ . \tag{2}$$

The underlying idea of "deep" learning is to train these layered models $f(\cdot;\boldsymbol{\theta})$ directly from input data by optimizing all involved parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_L)$ jointly with respect to the single loss function specified in Eq. (1). Unfortunately, this learning objective is non-convex and highly non-linear, except for trivial architectures. Thus, closed-form solutions for minimization, such as the ones existing for simpler models like least-squares regression [33, Sect. 3.1.4] can by no means be expected. Instead, optimization techniques such as gradient descent need to be applied which iteratively refine initial parameter guesses and ideally converge to suitable (local) optima. For layered models in Eq. (2), applying the chain rule allows for calculating partial derivatives for parameters of hidden layers which lead to the backpropagation algorithm [34].

Learning with Mini-Batches Since gradient descent requires the entire "batch" of training examples at once, it is rarely feasible for large-scale datasets. In consequence, today's standard optimization techniques are stochastic gradient descent (SGD) [35] and mini-batch gradient descent [36]. Both techniques approximate the true gradient $\nabla_{\boldsymbol{\theta}}\bar{\mathcal{L}}(\boldsymbol{\theta};\mathcal{D})$ of the objective function using a randomly drawn subset $\mathcal{S}^k \subseteq \mathcal{D}$ in every iteration k. SGD originally referred to the special case of a single example in each iteration (*i.e.,* $|\mathcal{S}| = 1$), whereas mini-batch gradient descent uses larger sets ($1 < |\mathcal{S}| \ll N$). Since both techniques rely on randomized approximations of the underlying gradient, both are commonly called stochastic gradient descent. In the remainder of this paper, we follow this naming convention and refer with SGD to every gradient descent which is not using the entire training set for gradient calculations.

In every iteration, SGD computes an approximated gradient $\tilde{\nabla}_{\boldsymbol{\theta}}$ based on the currently drawn subset:

$$\tilde{\nabla}_{\boldsymbol{\theta}}\bar{\mathcal{L}}(\boldsymbol{\theta};\mathcal{S}^k) = \frac{1}{|\mathcal{S}^k|}\sum_{i\in\mathcal{S}^k}\nabla_{\boldsymbol{\theta}}\mathcal{L}(f(\boldsymbol{x}_i;\boldsymbol{\theta}),y_i) + \nabla_{\boldsymbol{\theta}}\omega\ (\boldsymbol{\theta}). \tag{3}$$

Based on that, parameter estimates are updated using a gradient descent step of length $\gamma > 0$ which is often referred to as learning rate:

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \gamma\cdot\tilde{\nabla}_{\boldsymbol{\theta}}\bar{\mathcal{L}}(\boldsymbol{\theta}^k;\mathcal{S}^k) \ . \tag{4}$$

There are plenty of further optimization modifications including momentum [37], weight decay [34], dropout [38], as well as annealing schemes for the learning rate [39]. Since we are not empirically analyzing the training deep neural networks per se or the influence of such modifications on continuous learning, we refer to [36, 40, 37] for an evaluation of the different strategies.

4 Continuous Learning of Deep Neural Networks

In the following, we first define continuous learning scenarios and then propose different strategies how to cope with them with straightforward fine-tuning when using deep neural networks.

Continuous Learning Scenarios In contrast to standard batch learning applications, we are interested in continuous learning of deep neural networks in this paper. Thus, we are given a *series* of learning datasets $\mathcal{D}^t \subset (\Omega \times \mathcal{Y}^t)$. The goal is to learn the network $f(\cdot; \hat{\boldsymbol{\theta}}^{(t)})$ for each time step $t \geq 0$ with dataset $\mathcal{D}^t$ continuously over time. Depending on the overlap between the sets $\mathcal{D}^t$ for different t, we can differentiate between three scenarios:

(F) *Classical fine-tuning for new tasks*: Only two disjoint datasets are given and the task changed, *i.e.,* $\mathcal{D}^0 \cap \mathcal{D}^1 = \emptyset$ and $\mathcal{Y}^0 \cap \mathcal{Y}^1 = \emptyset$. This is for example used in [12–16] when a convolutional neural network pre-trained on ImageNet is fine-tuned in one step by either re-learning the last layer only or by performing optimization steps for all layers. For simplicity in notation, we ignore that fractions of $\mathcal{D}^1$ can already be included in $\mathcal{D}^0$. This can happen in special cases when the one-step fine-tuning is performed for new datasets covering a part of the original one.

(C1) *Continuous learning of known classes*: In this case, we get additional training examples for the classes we already know from the initial learning set $\mathcal{D}_0$, *i.e.,* $\forall t: \quad \mathcal{Y}^0 = \mathcal{Y}^t$ and $\mathcal{D}^{t-1} \subset \mathcal{D}^t$. This case describes a continuously growing training dataset and resembles the classical online learning setup.

(C2) *Continuous learning of known and new classes*: In addition to the previous scenario, we might also get examples of completely new classes, *i.e.,* $\forall t: \quad \mathcal{Y}^{t-1} \subseteq \mathcal{Y}^t$ with both sets being not necessarily equal and $\mathcal{D}^{t-1} \subset \mathcal{D}^t$.

In the following, we refer to $\mathcal{U}_t = \mathcal{D}^t \backslash \mathcal{D}^{t-1}$ as the *update set* of time step t.

Continuous Learning with Incremental Fine-tuning As stated before, the learning objective in Eq. (1) is in general non-convex and highly non-linear. In consequence, there are no closed-form update rules available like in the case of Gaussian process regression [41]. However, we can make use of the technique of warm-start optimization [42, 43], where we use the parameters $\boldsymbol{\theta}^{t-1}$ of the previous time step as initialization for the optimization of the current parameters $\boldsymbol{\theta}^t$. This strategy is also applied for standard use of fine-tuning and assumes that the network's parameters $\boldsymbol{\theta}^t$ vary smoothly with the extensions of the training dataset. Furthermore, it expresses the expectation that the optimization is more robust against bad local minima when started from a parameter vector which is likely close to an appropriate solution. Since case (F) is already well studied, we focus on (C1) and (C2) in the remainder of this paper.

Whereas (C1) can be directly tackled with fine-tuning without any modification of the network, scenario (C2) requires extending the last layer of a deep neural network for new categories. In particular, we need to add an additional output node in the last layer along with parameters in the corresponding previous fully-connected layer. We initialize those new parameters randomly using the normalization technique by [44].

In both settings, (C1) and (C2), a number of questions remains in practice. For example, it is unclear whether the solver needs to be run until convergence in each update step. It might be even beneficial to perform an early stopping of the optimization to increase generalization performance [45]. Furthermore, there is no clear guidance for how many layers we can adapt robustly when small sets of novel data arrives consecutively. Finally, the influence of label noise on the success of model updates is important to know for real-world applications. In Section 5, we empirically investigate these questions. In addition, we visualize in Section 6 the network's region of attention which changes most strongly when new data is continuously added.

5 Empirical Evaluation of Continuous Learning of DNNs

We conducted experiments in continuous learning scenarios with known and unknown classes. For different update strategies, we evaluated the classification accuracy of the continuously learned models. Our main findings can be summarized as follows:

1. The number of required SGD iterations can be significantly reduced in comparison to standard batch learning and set to small constants during training without a notable loss in test accuracy (Section 5.3).
2. Continuous fine-tuning by neglecting already known data leads to overfitting towards the update samples (Section 5.4).
3. Continuous fine-tuning is robust with respect to small amounts of label noise (Section 5.5).

We also studied the influence of the SGD batch size $|S|$ and the learning rate γ on the quality of continuous fine-tuning but found no surprising behavior. The results can be found in the supplementary material.In the following, we briefly explain the experimental setup before inspecting each of the previous three aspects in detail. Furthermore, we investigate the principal applicability of continuous learning for CNNs in Section 5.2 before we show more efficient realizations in the following sections.

5.1 Experimental Setup, Datasets, and Implementation Details

Network Architectures and Learning Since our main interest is in image categorization and understanding, we restrict our experiments to CNNs. Thereby, the desired invariance with respect to translation is explicitly encoded into the network layout. In all our experiments, we use the BVLC version of the classical AlexNet architecture [23] with network weights which have been pre-trained using the ImageNet ILSVRC-2010 challenge dataset [46]. Experiments were conducted with MatConvNet [47].

It is well known and extensively studied that a careful choice of the learning rate is among the most crucial aspects for learning [48]. Hence, a intensive evaluation of different learning rates is beyond the scope of this paper. Instead, we use results of preliminary experiments and fix the learning rate to 0.001 in all scenarios. This choice is consistent with reported observations [22] and allows for comparable results in all experiments. Furthermore, we use small update sets $\mathcal{U}_t$ of size $|\mathcal{U}_t| = 25$ which contain only samples from a single category. We further follow the default parameter settings of MatConvNet [47] and apply a weight decay of 0.0005 and a momentum of 0.9 .

Datasets for Evaluations For the evaluation of our proposed learning strategies, we use two state-of-the-art datasets. First, we use MS-COCO-full-v0.9 of Lin et al. [4] which provides a challenging setup similar to real-world applications. Since we are interested in classification tasks, we consider all ground truth bounding boxes with at least 256 pixel height and width. Furthermore, we only use categories that consist of 500 to 1,000 examples. Thereby, we obtain 15 categories which provide enough training examples for initial training and several update sets. The total number of images of this subset is approximately 11,000 for training and test set each.

As second evaluation dataset, we chose the Stanford40Actions dataset introduced in [49]. The dataset contains still images of different activities and provides a split into 4,000 images for training and 5,532 hold-out images for testing. Hence, adapting pre-trained models to the new task is clearly required.

In comparison with large-scale datasets like ImageNet, both datasets contain only small numbers of examples. Thereby, they are well suited for evaluating continuous learning with little data and allow for insights in a rather uncommon application scenario for deep neural networks.

Experimental Setup To evaluate continuous learning, we randomly pick 10 of the given classes to initially fine-tune the CNN. From the remaining data, we chose 5 more classes randomly which serve as novel data to be added during the process of continuous learning. For every category, we randomly select 100 examples. Those samples are either used as initial known data and or as update sets depending on if the category is selected to be added during the experiments or to be known initially. Classification accuracy after every update step is measured on the corresponding test data of the dataset. To reduce randomization effects during evaluation, we repeat the process of random sample selection three times. Furthermore, the entire setup is conducted for three different class splits, which yields 9 different setups to average results over.

5.2 Comparison of Continuous and One-step Fine-tuning

In a first experiment, we analyze whether continuous fine-tuning of deep neural networks is possible without a loss in accuracy. As baseline serves the classical one-step fine-tuning paradigm, where we perform only a single update step when all data $\mathcal{D}_T$ is available. This is similar to case (F) described in Section 4. We compare this baseline against continuous fine-tuning, for which we conduct an fine-tuning cycle as soon as an update set $\mathcal{U}_t$ becomes available which corresponds to case (C1) and (C2). Due to the design of our experiments, we evaluate both cases in a mixed manner because not every

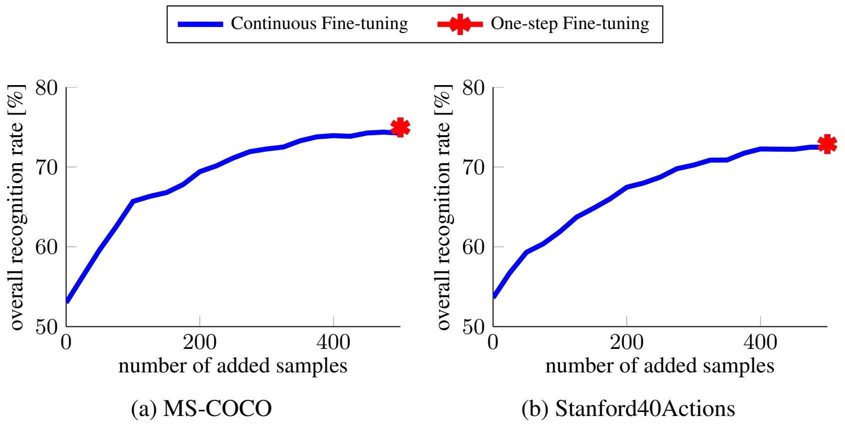


Fig. 2: Comparison of classical one-step fine-tuning (*i.e.,* waiting for all data to be available) and continuous fine-tuning (*i.e.,* fine-tuning as soon as new datasets are available).

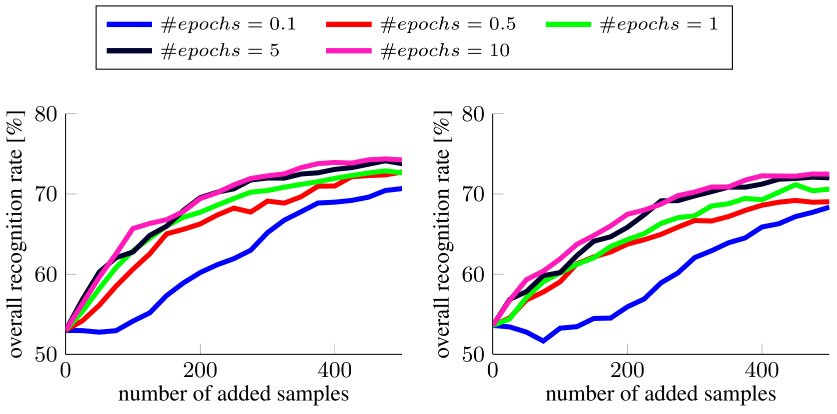
update set contains new classes. An additional evaluation of case (C1) can be found in the supplementary material.

Arguably, if one-step fine-tuning and continuous fine-tuning have been finally conducted with exactly the same amount of data, it sounds intuitively plausible that estimated parameters lead to the same classification accuracy. However, the path $(\boldsymbol{\theta}^t)$ of parameters the continuous learning follows are end in a completely different initialization for $\boldsymbol{\theta}^T$ in the last step compared to one-step fine-tuning. For the importance of initialization, see also the initialization-as-regularization discussion in [11]. We follow the previously described experimental setup and perform 10 epochs with and a SGD batch size of $|\mathcal{S}| = 64$ per update to fine-tune all layers. Although we also performed analyses with adapting parameters of an arbitrary number of layers, experimental results lead to similar findings. Results are shown in Fig. 2.

It can clearly be seen that both learning approaches obtain similar accuracy after all training data has been processed. Hence, we conclude that continuous fine-tuning of deep neural networks is possible without significant loss in accuracy. However, conducting an entire fine-tuning cycle whenever a new update set is available is computationally expensive. In the next experiment, we analyze how to reduce this computational burden.

5.3 Speeding Up Continuous Fine-tuning

The simplest solution for reducing the computation time of each fine-tuning step is to reduce the total number of SGD iterations from entire epochs to few steps done at each update. However, this comes at the risk of interrupting gradient descent too early, *i.e.,* far off the local optimum. In the following, we empirically investigate how many iterations we need in each step without reducing the classification accuracy of learned models. Therefore, we keep the previous experimental setup and only vary the number of SDG iterations T_{sgd} conducted in each update step but keep the SGD mini-batch size of $|\mathcal{S}| = 64$.



(a) MS-COCO (b) Stanford40Actions

Fig. 3: Performance comparison for different numbers of epochs and therefore an increasing number of SGD iterations used during fine-tuning.

It is common practice to express the number of SGD iterations relative to the total number of known training examples, *i.e.,* in epochs e. The resulting number of SGD iterations, *i.e.,* gradient descent updates, is then given by $T_{\text{sgd}} = \left\lceil \frac{n_t}{|\mathcal{S}|} \right\rceil \cdot e$. In the first part of this analysis, we investigate how many epochs we require performing continuous fine-tuning in a stable fashion for all layers. Hence, we evaluated how many epochs we need to successfully learn deep neural networks continuously. Results for different epochs in each step are shown in Fig. 3.

As can be seen, the resulting accuracy remains surprisingly unchanged even with extremely few iterations in each update step. In fact, even a half epoch can be sufficient and we only observe a clear drop in accuracy for the extreme case of 0.1 epochs. We attribute this observation to the effect that 0.1 epochs can easily miss selecting a representative subset for SGD optimization.

Based on the previous results, the question arises whether it is indeed necessary to see every training example in each update. To answer this question, we go beyond the previous analysis and further explore the possible range of fast fine-tuning. Whereas the number of SGD steps grew previously due to the increasing number of known examples after each step, we now fix the number of SGD update steps T_{sgd} to a constant number. This leads to a further significant speed-up of DNN learning. Each update step is comprised of a single mini-batch of size 25 which equals the update set size $|\mathcal{U}_t|$. The results can be seen in Fig. 4.

Similarly to our previous analysis with a varying number of epochs (*i.e.,* with an increasing number T_{sgd} of SGD steps), we observe again that we can drastically reduce the required computations without a strong loss in accuracy. As previously, we attribute the significant accuracy drop for $T_{\text{sgd}} = 1$ to the fact that no representative subset could be randomly sampled. In the next analysis, we investigate whether alternative sampling techniques can improve the accuracy of continuous fine-tuning.

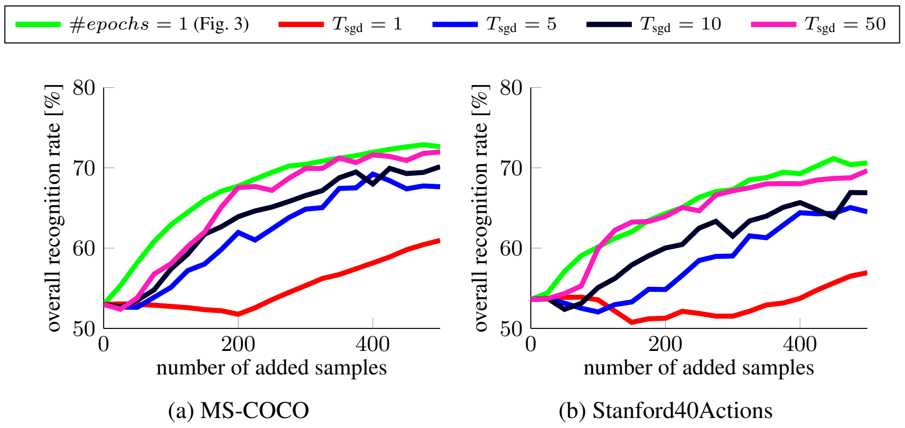


Fig. 4: Performance comparison for continuous fine-tuning with a fixed number T_{sgd} of SDG iterations compared to an increasing number of SGD iterations depending on the number of training examples (fixed number of epochs).

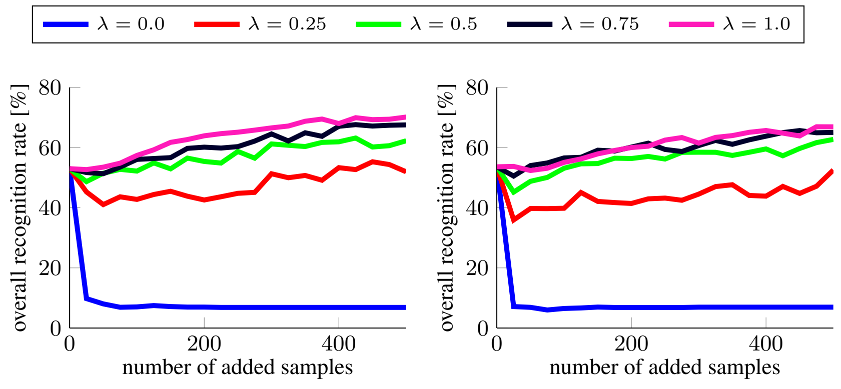
5.4 Continuous Learning with Varying Update Influence

A majority of techniques that incrementally update classifier parameters involves a hyperparameter λ that weights the influence of the new data of the current update set. In the following, we empirically answer the natural question whether different weights for old and new data also lead to an improved continuous fine-tuning accuracy for DNNs. Therefore, we sample examples during SGD iterations non-uniformly and dependent on an example-specific probability p_i:

$$p_i = \frac{1-\lambda}{|\mathcal{U}_t|} \quad \text{if} \quad \boldsymbol{x}_i \in \mathcal{U}_t \quad \text{and} \quad \frac{\lambda}{|\mathcal{D}^{t-1}|} \quad \text{otherwise} \;, \tag{5}$$

with $0 \leq \lambda \leq 1$. For the extreme case of $\lambda = 0$, training examples of previous time steps are completely ignored during sampling and their information is only indirectly used through parameter initializations. This could be seen as a naive realization of continuous learning since it is similar to online updates of common classification models. Furthermore, $\lambda = 1$ is equivalent to the previous continuous fine-tuning analyses but postpones the usage of examples from the current update set to the next update step. For values of λ within that range, we can control the relative importance of update set data by their probability of being sampled. To investigate the influence of λ, we performed $T_{\text{sgd}} = 10$ iterations with a fixed SGD size of $|\mathcal{S}| = 25$ which corresponds to the number of added samples $|\mathcal{U}_t|$. The results for different values of λ can be seen in Fig. 5 where parameters of all layers are adapted. Note that we also investigated to learn only upper layers, which lead to similar results (can be found in the supplementary material).

We conclude that a CNN can only be fine-tuned robustly in an continuous fashion if a fraction of new and old data is considered during SGD iterations. The balance of the data has to be chosen carefully because already known data prevents the net from overfitting to new data and too few new data is not sufficient to prevent this behavior.



(a) MS-COCO (b) Stanford40Actions

Fig. 5: Incremental fine-tuning of all layers with different choices of λ.

In our case, new samples are considered as known, one step after they are added. This explains the similar results $\lambda = 0.75$ and $\lambda = 1.0$. For the extreme case $\lambda = 0$, iterative fine-tuning completely fails which can be attributed to complete overfitting to the new update set $\mathcal{U}_t$.

5.5 Continuous Learning with Label Noise

In contrast to the previous analyses, object labels are hardly perfect in real-world applications, especially with non-experts as annotators in the loop. We were thus interested in how robust the process of continuous learning is against wrongly provided label information. To investigate this question, we perform 10 epochs in each update step, and optimize parameters of all layers. We work with a mini-batch size of $|\mathcal{S}| = 64$. To analyze the influence of label noise, a specific fraction of the newly added data is replaced with examples from remaining categories while keeping the label as in the previous experiments. The results for different amounts of the induced label noise are shown in Fig. 6.

As expected, the resulting accuracy is constantly degraded with an increasing level of wrongly labeled examples. However, 10% noise results in only 2% loss in accuracy which is likely acceptable for real-world applications.

6 What Was New? Visualizing Network Changes

Especially when new categories are added to the training set, we can expect that filters in convolutional layers become specifically tuned to characteristic patterns of the new category. To visually investigate where and how our network changed over time, we first pick the filter that underwent the most drastic change and then visualize it's corresponding attention region in the image.

We expect that the chosen filter is likely to pay attention to a pattern which is discriminative for the category that has been added during the learning step. An intuitive

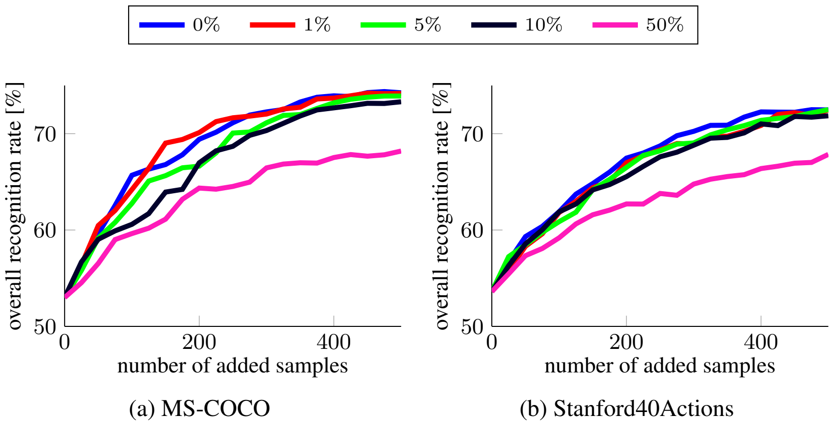


Fig. 6: Impact of label noise in update sets on continuous learning performance.

possibility to measure the “change” of a filter is to calculate differences of filter weights before and after the update. However, this strategy might easily lead to wrong interpretations, since large weight differences in one layer could be partially compensated by changes in previous layers. Hence, large changes of filter entries do not need to correspond to changes in the resulting learned representation. Due to this reason, we follow an alternative selection strategy: we determine the filter with the largest overall change of its outputs rather than in its parameters. The output change is measured by L_2-norm of differences of outputs obtained from single filters before and after an update with new samples. We only consider the `conv5` layer of AlexNet, since it is known to be related to semantic part information [50, 51].

For visualizing the corresponding attention region of the filter in the image, we could use directly the `conv5` activations. However, the resolution of the `conv5` channel is rather low. For a spatially precise localization analysis of the filter's attention, we use the gradient technique of [51, 52] to calculate attention maps. We then compute bounding boxes from these maps, by considering the maximum response location as center and twice the standard deviation in x and y direction as width and height. This allows us to visualize the main attention region of the chosen `conv5` filter.

Visualization Results We apply our visualization technique to the previously investigated scenarios of continuous learning and visualize the attention maps after a single category has been added. We use update sets of sizes $|\mathcal{U}_t| = 100$ and learn parameters of all layers. The output change of `conv5` filters is evaluated on all examples of the novel category. We show the result of our visualization in Fig. 7 for different categories.

As can be seen on the left hand side of the figure, the `conv5` channel with the highest output change often shifts focus to parts of the newly added class, although there was no localization information provided during training. Furthermore, it is interesting to see that the representation within the convolutional neural network adapts already for a small update set size. More visualizations can be found in the supplementary material.

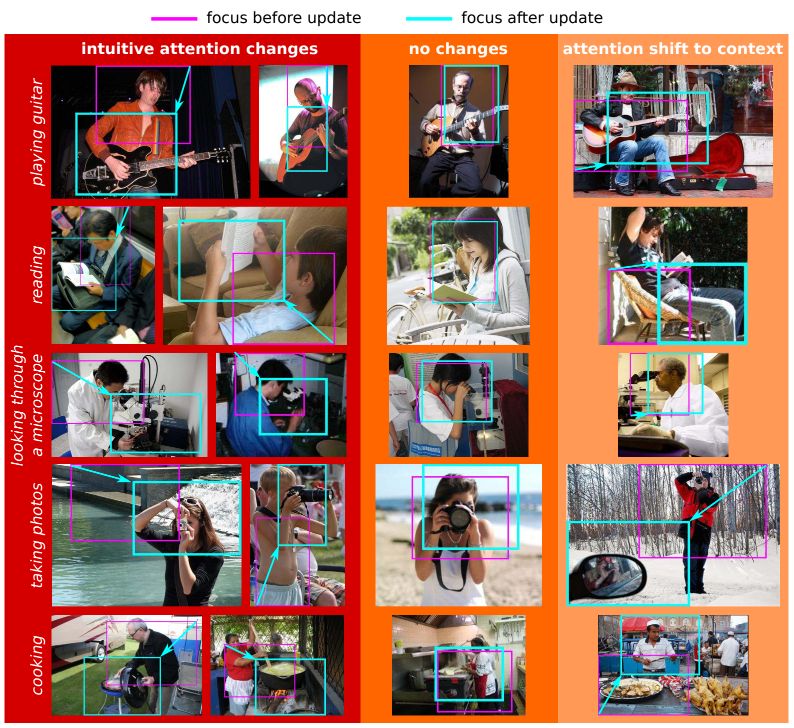


Fig. 7: Visualization of network changes during continuously learning categories from the Stanford40Actions dataset. We visualized attention regions of the single filter in `conv5` which changed most strongly during a single learning step. Regions are shown before and after the respective category became known (*magenta box* and *cyan box*). *Left*: the attention shifts towards the action-related objects. *Middle*: no visible changes in attention. *Right*: the attention region shifted to contextual related areas.

7 Conclusions

In this paper, we studied continuous learning of deep neural networks with incoming data streams. While previous work focused on one-step fine-tuning pre-trained networks to novel tasks, we empirically studied more general scenarios of frequent and small update steps. Based on our analyses, we conclude that continuous learning can be directly achieved by continuous fine-tuning. We further investigated how continuous fine-tuning can be speeded up by reducing the number of SGD iterations. In fact, not even a single epoch is required during each update step to train models robustly. Furthermore, we also analyzed the influence of label noise as it is typically present in real-world applications. While heavily noisy data severely degrades the accuracy of learned models, we found that even with 10% incorrectly labeled data continuous learning can still be successful. Finally, we visualized the change of the network's attention during continuous learning. Thereby, we were able to visualize how well the model shifts its focus to semantically plausible regions of recently learned categories.

References

1. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)
2. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html (2008)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR. (2009) 248–255
4. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV. (2014) 740–755
5. Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset. In: CVPR Workshop on The Future of Datasets in Vision (CVPR-WS). (2015)
6. Freytag, A., Rodner, E., Denzler, J.: Selecting influential examples: Active learning with expected model output changes. In: ECCV. Volume 8692. (2014) 562–577
7. Käding, C., Freytag, A., Rodner, E., Bodesheim, P., Denzler, J.: Active learning and discovery of object categories in the presence of unnameable instances. In: CVPR. (2015) 4343–4352
8. Freytag, A., Rodner, E., Bodesheim, P., Denzler, J.: Rapid uncertainty computation with gaussian processes and histogram intersection kernels. In: ACCV. (2012) 511–524
9. Tax, D., Laskov, P.: Online svm learning: from classification to data description and back. In: Workshop on Neural Networks for Signal Processing (NNSP). (2003) 499–508
10. Sillito, R.R., Fisher, R.B.: Incremental one-class learning with bounded computational complexity. In: ICANN, Springer (2007) 58–67
11. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? JMLR **11** (2010) 625–660
12. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: ECCV. (2014)
13. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: CVPR. (2014) 584–599
14. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Simultaneous detection and segmentation. In: ECCV. (2014)
15. Branson, S., Van Horn, G., Belongie, S., Perona, P.: Improved bird species categorization using pose normalized deep convolutional nets. In: BMVC. (2014)
16. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR. (2014) 580–587
17. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: NIPS. (2001) 409–415
18. Yeh, T., Darrell, T.: Dynamic visual category learning. In: CVPR. (2008) 1–8
19. Mensink, T., Verbeek, J., Perronnin, F., Csurka, G.: Distance-based image classification: Generalizing to new classes at near-zero cost. TPAMI **35** (2013) 2624–2637
20. Ristin, M., Guillaumin, M., Gall, J., Gool, L.V.: Incremental learning of ncm forests for large-scale image classification. In: CVPR. (2014) 3654–3661
21. Hospedales, T.M., Gong, S., Xiang, T.: Finding rare classes: Active learning with generative and discriminative models. TKDE **25** (2013) 374–386
22. Wilson, D.R., Martinez, T.R.: The general inefficiency of batch training for gradient descent learning. Neural Networks **16** (2003) 1429–1451
23. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012) 1097–1105

24. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop. In: Neural networks: Tricks of the trade. Springer-Verlag Berlin Heidelberg (1998) 9–48
25. Xiao, T., Zhang, J., Yang, K., Peng, Y., Zhang, Z.: Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In: International Conference on Multimedia. (2014) 177–186
26. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: CVPR. (2014)
27. Hoffman, J., Darrell, T., Saenko, K.: Continuous manifold based adaptation for evolving visual domains. In: CVPR. (2014) 867–874
28. Tzeng, E., Hoffman, J., Darrell, T., Saenko, K.: Simultaneous deep transfer across domains and tasks. In: ICCV. (2015) 4068–4076
29. Pan, S.J., Yang, Q.: A survey on transfer learning. TKDE **22** (2010) 1345–1359
30. Jie, L., Tommasi, T., Caputo, B.: Multiclass transfer learning from unconstrained priors. In: ICCV. (2011) 1863–1870
31. Thrun, S.: Lifelong learning: A case study. Technical report, DTIC Document (1995)
32. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67** (2005) 301–320
33. Bishop, C.M.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer (2006)
34. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature (1986) 323–533
35. Bottou, L.: Stochastic gradient tricks. In: Neural networks: Tricks of the trade. Springer-Verlag Berlin Heidelberg (2012) 430–445
36. Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Le, Q.V., Ng, A.Y.: On optimization methods for deep learning. In: ICML. (2011) 265–272
37. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: ICML. (2013) 1139–1147
38. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. JMLR **15** (2014) 1929–1958
39. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
40. Larochelle, H., Bengio, Y., Louradour, J., Lamblin, P.: Exploring strategies for training deep neural networks. JMLR **10** (2009) 1–40
41. Lütz, A., Rodner, E., Denzler, J.: I want to know more: Efficient multi-class incremental learning using gaussian processes. Pattern Recognition and Image Analysis. Advances in Mathematical Theory and Applications (PRIA) **23** (2013) 402–407
42. Tsai, C.H., Lin, C.Y., Lin, C.J.: Incremental and decremental training for linear classification. In: SIGKDD. (2014) 343–352
43. Chu, B.Y., Ho, C.H., Tsai, C.H., Lin, C.Y., Lin, C.J.: Warm start for parameter selection of linear classifiers. In: SIGKDD. (2015) 149–158
44. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. (2010) 249–256
45. Perronnin, F., Akata, Z., Harchaoui, Z., Schmid, C.: Towards good practice in large-scale learning for image classification. In: CVPR. (2012)
46. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV **115** (2015) 211–252
47. Vedaldi, A., Lenc, K.: Matconvnet – convolutional neural networks for matlab. In: International Conference on Multimedia. (2015)
48. Orr, G.B., Müller, K.R.: Neural networks: tricks of the trade. Springer (2003)

49. Yao, B., Jiang, X., Khosla, A., Lin, A.L., Guibas, L., Fei-Fei, L.: Human action recognition by learning bases of action attributes and parts. In: ICCV. (2011) 1331–1338
50. Bolei, Z., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Object detectors emerge in deep scene cnns. In: ICLR. (2015)
51. Simon, M., Rodner, E., Denzler, J.: Part detector discovery in deep convolutional neural networks. In: ACCV. Volume 2. (2014) 162–177
52. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR-WS. (2014)