KI - Künstliche Intelligenz (2022) 36:91–98
https://doi.org/10.1007/s13218-021-00746-2

PROJECT REPORTS

Multi-phase Fine-Tuning: A New Fine-Tuning Approach for Sign Language Recognition

Noha Sarhan[1] · Mikko Lauri[1] · Simone Frintrop[1]

Received: 24 April 2020 / Accepted: 22 November 2021 / Published online: 24 February 2022

Abstract
In this paper, we propose multi-phase fine-tuning for tuning deep networks from typical object recognition to sign language recognition (SLR). It extends the successful idea of transfer learning by fine-tuning the network's weights over several phases. Starting from the top of the network, layers are trained in phases by successively unfreezing layers for training. We apply this novel training approach to SLR, since in this application, training data is scarce and differs considerably from the datasets which are usually used for pre-training. Our experiments show that multi-phase fine-tuning can reach significantly better accuracy in fewer training epochs compared to previous fine-tuning techniques

Keywords Sign language recognition · Transfer learning

1 Introduction

Different hand gestures, facial expressions and body postures form grammatically-complete, highly-structured sign languages. Sign language recognition (SLR) can be categorized into three groups: recognition of alphabets [7, 24, 25], isolated words [13], or continuous sentences. In continuous SLR the input comprises a video containing a sequence of gestures and the desired output is a sequence of words complying to the grammar of that sign language [6, 11, 12].

Research on SLR has shifted to using deep learning methods, rather than hand-crafted features, the most prominent work includes [5, 6, 11, 12]. Recent methods even involve domain adaption [19] and sign language transformers [3]. The majority of the aforementioned approaches rely on transfer learning by including some pre-trained convolutional neural network (CNN), either for classifying individual frames or as a feature extractor. Transfer learning uses the weights learned while solving a different, yet related, task to initialize the weights of the network solving a new task. We refer to the former network as the source network, and to the latter as the target network.

Transfer learning involves making two important decisions: First, how many layers to transfer? Second, whether to freeze the transferred layers or to fine-tune them? [4, 5, 23, 26]. While no clear theory exists on how to make these choices, they depend on both the size of the target dataset and its similarity to the source data [26]. Answering these questions becomes even more complicated when the source and target tasks differ strongly.

Existing work on transfer learning for CNNs can be divided into two groups: (a) using a pre-trained source CNN just as a feature extractor [16, 20]; (b) transferring some layers' weights to a target network, randomly initializing the weights of the non-transferred layers, and fine-tuning the target network on the target domain [23, 26]. Fine-tuning can be done by learning either (1) the weights in the non-transferred layers only [14], or (2) also the weights in some transferred layers, where usually the top-k layers are trained at once. We refer to the second fine-tuning method as *single-phase fine-tuning*. Research shows a clear advantage for fine-tuning weights in transferred layers in contrast to freezing them [15, 26]. However, if the target data is inherently different from the source data, these fine-tuning methods do not always works [1, 2, 20, 26].

In this paper, we propose *multi-phase fine-tuning* for tuning deep networks from everyday object recognition to SLR.

✉ Noha Sarhan
sarhan@informatik.uni-hamburg.de

Mikko Lauri
lauri@informatik.uni-hamburg.de

Simone Frintrop
frintrop@informatik.uni-hamburg.de

[1] Department of Computer Science, Universität Hamburg, Hamburg, Germany

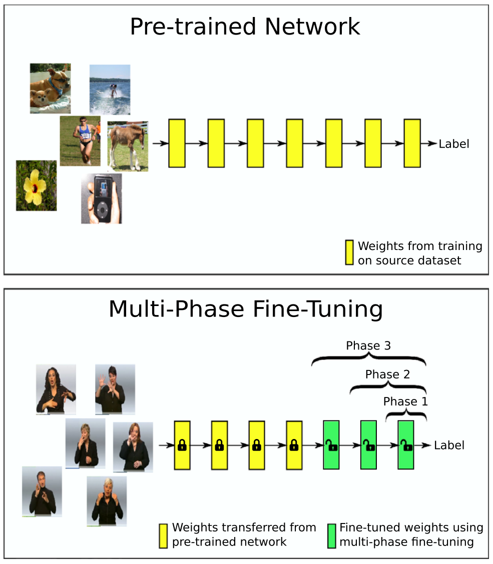
Pre-trained Network
Label
Weights from training on source dataset
Multi-Phase Fine-Tuning
Phase 3
Phase 2
Phase 1
Label
Weights transferred from pre-trained network
Fine-tuned weights using multi-phase fine-tuning

Fig. 1 Top: source network pre-trained on ImageNet. Bottom: proposed multi-phase fine-tuning approach. The layers to be fine-tuned in the target network are adapted over several phases starting from the top layer.

The concept is depicted in Fig. 1. It extends the successful idea of transfer learning by fine-tuning the network's weights in several phases. In the first phase, only a few topmost layers are fine-tuned. In successive phases, more layers are added and jointly fine-tuned with the layers from previous phases. We evaluate our proposed approach using GoogLeNet [21] on frame-level classification in continuous sign language videos, a considerably different domain from ImageNet [18] (see Fig. 1). CNNs applied to individual frames are a key step in several SLR systems [5, 6, 11, 12, 17]. Our results show that multi-phase fine-tuning considerably improves task performance and that it converges faster with fewer learning epochs compared to the earlier single-phase fine-tuning.

Recent SLR methods rely on deep learning methods. Most methods employ some pre-trained network, however little research has been done in investigating what is the best approach to fine-tune these pre-trained networks to the new task, especially since it is usually quite different than the source task (e.g. object recognition when pre-trained on ImageNet). In this work, we focus on investigating how to fine-tune the network from the task of object recognition to sign language recognition. We believe that fine-tuning a pre-trained network phase-wise would allow the top layers to adapt to the new tasks, while keeping the shallower layers unchanged, thereby improving the generalization capabilities of the network.

Our contribution is threefold. (1) We introduce a multi-phase fine-tuning strategy that improves accuracy and additionally allows faster training. We extend [26] by training the unfrozen layers step-wise as opposed to fine-tuning them all at once. (2) We demonstrate the success of multi-phase fine-tuning for transfer learning between two very different domains: from everyday object recognition to SLR. (3) We present a CNN-based approach for frame-based SLR which can be valuable for the sign language community.

The remainder of the paper is structured as follows: in Sect. 2, we thoroughly explain our proposed fine-tuning approach. In Sects. 3 and 4 we demonstrate our experimental setup and results. Section 5 concludes the paper.

2 Methods

In this section, we give an overview of our CNN training, standard fine-tuning methods, and illustrate our proposed multi-phase fine-tuning approach.

2.1 CNN Training

A CNN function maps the input x to a predicted label $\hat{y} = f(x;w)$, given trainable weights w. In supervised learning, CNNs are trained using stochastic gradient descent (SGD), given a training data set $D = \{(x^i, y^i)\}_{i=1}^{N}$ with N inputs x^i and labels y^i. SGD alternates between feedforward and backpropagation steps using mini-batches of m examples from the training set. A minibatch is a subset $\{(x^i, y^i)\}_{i\in I} \subset D$, where $I \subset \{1, \ldots, N\}$ such that $|I| = m$.

In the feedforward step, a prediction $\hat{y}^i$ is computed for each sample x^i in the mini-batch given the current weights w. A scalar loss between the true labels and predictions is calculated by

$$\mathcal{L}(w) = \frac{1}{m}\sum_{i\in I}\mathcal{L}_i(\hat{y}^i, y^i) = \frac{1}{m}\sum_{i\in I}\mathcal{L}_i(f(x^i;w), y^i), \tag{1}$$

where $\mathcal{L}_i$ is the per-sample loss function, e.g., cross entropy for classification.

For backpropagation the gradient of $\mathcal{L}$ with respect to the weights w is first evaluated. We apply SGD with momentum, the initial weights w_0 are drawn randomly. The velocity ϵ_0 representing the past gradients is initialized to zero. At training iteration $t \geq 1$,

$$\begin{aligned} g_t &= \frac{1}{m}\nabla_w \sum_{i\in I}\mathcal{L}_i(f(x^i;w_{t-1}), y^i) \\ \alpha_t &= (1-\psi)\alpha_{t-1} \\ \epsilon_t &= \gamma\epsilon_{t-1} - \alpha_t g_t \\ w_t &= w_{t-1} + \epsilon_t, \end{aligned} \tag{2}$$

where g_t is the current gradient estimate, ϵ_t is the step for modifying the weights, dependent on the former

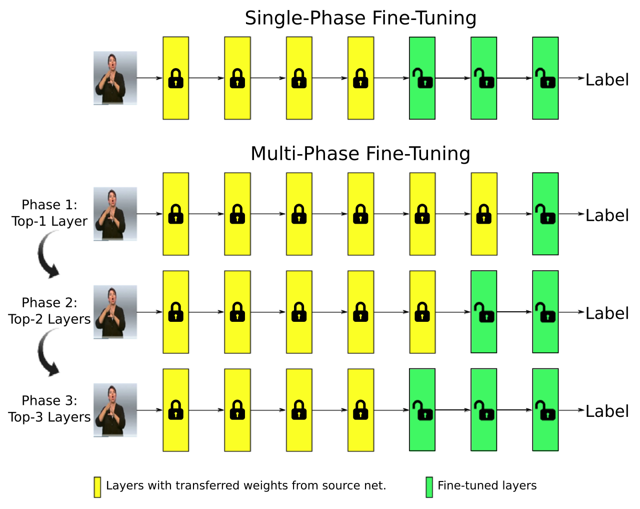

Fig. 2 Top: single-phase fine-tuning unlocks and trains weights in all of the top-k (here $k = 3$) layers of a CNN simultaneously. Our multi-phase fine-tuning (bottom) trains the weights in the top-k layers in several phases, successively adding more layers

gradients weighted by momentum γ, and the current gradients weighted by the learning rate α that decays at a rate of ψ.

2.2 Single-Phase Fine-Tuning

The initial weights w_0 for a target network, apart from the last classifying layer, are initialized to pre-trained values from a source network. The classifying layer is modified to have as many neurons as the number of classes in the target task and is initialized with random weights. Weights of the target network are then fine-tuned, via Eq. (2), using a training dataset from the target domain.

A key question is whether to freeze transferred weights or fine-tune them to the new task. Freezing weights is often referred to as "off-the-shelf" transfer learning [20]; only the weights in the last classifier layer are updated.

If fine-tuning is applied to other layers as well, typically the k topmost layers are fine-tuned while keeping the other layers' weights at their source network values [23, 26]. We refer to this approach as *single-phase fine-tuning*. For a network with a total of L layers, we use the notation *top-k layers* to refer to updated weights in layers $(L-k+1,\dots,L)$. Weights in layers $(1,\dots,L-k)$ remain frozen. Single-phase fine-tuning of the top-3 layers is illustrated in Fig. 2 (top).

2.3 Multi-phase Fine-Tuning

We propose a multi-phase fine-tuning approach where the top-k layers are trained sequentially with a step-size s in (k/s) phases[1] until all of the k layers have been fine-tuned. In the first phase we fine-tune only the top-s layers. In each of the following phases, we add s more layers to be fine-tuned. At each phase, training continues until a pre-specified termination criterion is reached, e.g., the maximum number of training epochs or saturation of the validation loss.

For example, fine-tuning top-k layers with a step-size $s = 1$ for $k = 3$ has three phases; P1, P2, and P3 (see Fig. 2):

P1 Start by fine-tuning one layer, e.g., only the topmost layer of the network.
P2 Include one more layers for a total of 2 and fine-tune the top-2 layers.
P3 Add again one layer for a total of 3 and fine-tune the top-3 layers.

We remark that if $s = k$, multi-phase fine-tuning is equivalent to single-phase fine-tuning of top-k layers.

3 Experimental Setup

In Sect. 3.1 we describe the dataset and the evaluation metrics applied in this work. Section 3.2 covers the implementation details.

3.1 Dataset and Metrics

We use RWTH-PHOENIX-Weather Multisigner 2014 [8, 10], one of the largest, publicly available, annotated datasets in the sign language domain. It has of 6841 videos of continuous signing in German sign language, each video labelled with an output sentence as a sequence of words (Fig. 3). Note that the resulting sequence of words is not a translation to spoken language, rather a literal translation of the signs.

We solve a classification problem where the input is a single frame and as output label we use the frame-to-label alignments provided by [12]. Each word is split into three parts each making up one label as depicted in Fig. 3, resulting in 3693 classes for 500,000 frames. We reserve 10% of the images for validation. Throughout our experiments, we record the top-1 and top-5 classification accuracies.

3.2 Implementation Details

CNN Architecture: We opt for GoogLeNet [21] with inception V3 [22] pretrained on ImageNet as the source network. It is the most commonly used network in recent SLR [6, 11, 12]. GoogLeNet consists of several (precisely 8) inception

[1] We require that k is an integer multiple of s.

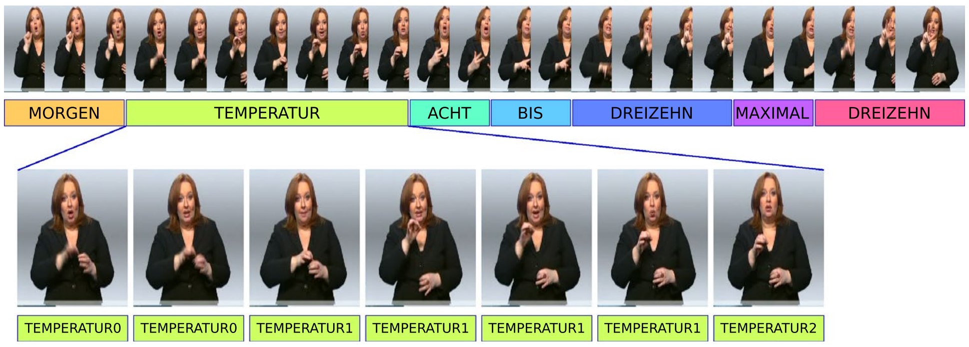

Fig. 3 Sample image sequence from RWTH-PHOENIX-Weather dataset [8]. It contains video sequences from German broadcast news along with their sentence annotations (in German). Authors of [12] have automatically aligned labels to each frame in the video sequence. Each word is further split into three word-part labels; an example is shown for the word "Temperatur" (English: temperature)

modules, so we investigate the effect of fine-tuning a varying number of such modules instead of layers. Thus, we will be referring to layers as modules in our notation (top-k modules instead of top-k layers).

Fine-Tuning Setup: We fine-tune the top-k modules of the network, for $k = 1, 2, 3, \dots, 8$. We compare the accuracy of our proposed multi-phase fine-tuning to traditional single-phase fine-tuning. For multi-phase fine-tuning, we report results for step size $s = 1, 2, 3$ for all values of k. In all cases, the fully-connected layers are always trained from scratch. We note that fine-tuning the top-8 inception modules is equivalent to fine-tuning the entire network.

Training Hyperparameters: We apply SGD with Nesterov momentum $\gamma = 0.9$, and learning rate $\alpha = 0.01$ that decays with rate $\psi = e^{-6}$, and batch size $m = 32$. We apply a categorical cross-entropy loss. We adopt an early-stopping approach, where training is terminated if the validation loss does not improve for 3 consecutive epochs. Random weights are initialized using Xavier normal initializer [9].

4 Results and Analysis

In this section we report results of our baseline, single- and multi-phase fine-tuning experiments, in addition to hyperparameter exploration for mulit-phase fine-tuning.

4.1 Baseline Experiments

Frame-based recognition is a submodule in currently existing SLR systems [5, 6, 11, 12], however, it is not addressed separately. Therefore, we assess the base difficulty of the task with three baseline methods and report top-1 and top-5 accuracies in Table 1.

To see how ImageNet features perform on the new task, we apply GoogleNet pre-trained on ImageNet as a feature extractor and train a fully-connected classifying layer on top.

We also try two non-deep-learning methods to assess the difficulty of the problem. (1) Using SIFT features we extract the image descriptors, normalize and vector-quantize them using k-means to an 800-dimensional feature vector. A random forest classifier with eight trees and a maximum depth of 30 is trained for classification. (2) Using HOG features, we extract a feature vector for each image, and train a logistic regression classifier via SGD.

HOG with logistic regression performs best reaching a top-1 accuracy of 16.9%. The way it outperforms GoogLeNet as a feature extractor suggests that the learned features do not transfer very well to the new target domain.

4.2 Single-Phase vs. Multi-phase Fine-Tuning

We compare the proposed multi-phase fine-tuning with the standard single-phase fine-tuning. Table 2 shows the classification accuracies for both methods with step size $s = 1$. We note that fine-tuning only the topmost module ($k = 1$) already outperforms our baseline results from Table 1. For all values of k modules that are fine-tuned, we observe that multi-phase fine-tuning consistently reaches a higher accuracy than fine-tuning the same modules in a single phase.

Figure 4 (left) visualizes the top-1 accuracy as function of the number of modules fine-tuned. We note that with multi-phase fine-tuning the accuracy constantly improves as more

Table 1 Top-1 and top-5 classification accuracies of baseline methods.

Method	Top-1 accuracy (%)	Top-5 accuracy (%)
GoogLeNet feature extractor	14.7	30.8
SIFT with random forest	4.6	13.5
HOG with logistic regression	**16.9**	**35.9**

The values in bold highlight the best-performing method

Table 2 Top-1 and top-5 accuracies when fine-tuning the top-k modules of GoogLeNet either in a single-phase or multiple phases with a step size $s = 1$

Accuracy	Method	$k = 1$ (%)	$k = 2$ (%)	$k = 3$ (%)	$k = 4$ (%)	$k = 5$ (%)	$k = 6$ (%)	$k = 7$ (%)	$k = 8$ (%)
Top-1	Single-phase	24.5	30.0	22.0	26.3	11.8	10.9	5.0	8.2
	Multi-phase	24.5	31.1	32.5	37.2	38.1	39.3	40.0	41.0
Top-5	Single-phase	46.5	56.9	46.0	52.7	31.8	27.2	16.7	22.7
	Multi-phase	46.5	57.1	58.1	62.8	64.1	64.9	65.8	66.6

Note that for $k = s = 1$, single- and multi-phase fine-tuning are equivalent

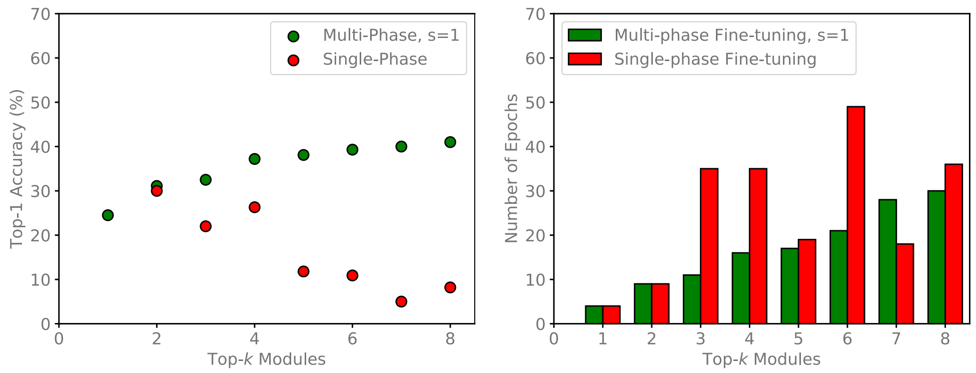

Fig. 4 Left: Top-1 accuracy as a function of the number of modules k fine-tuned for multi-phase fine-tuning with step size $s = 1$ and single-phase fine-tuning. Right: Number of training epochs. Note that for $k = s = 1$, the two fine-tuning approaches are equivalent

modules are included. For single-phase fine-tuning, accuracy starts to degrade for $k > 4$.

Moreover, multi-phase fine-tuning requires less training epochs, see Fig. 4 (right). Training the network in multiple phases gives top layers the chance to adapt to the new task while lower layers remain unchanged. Our results show that this property of multi-phase fine-tuning improves the generalization capability of the network. Fine-tuning pre-trained layers' weights should not be done while random weights of newly added fully-connected layers are yet to be trained. We hypothesize that the pre-trained layers' weights may prematurely start to adapt to the random weights.

4.3 Different Step-Sizes

The step size s controls how many new modules are added for fine-tuning in each phase. We varied the step size to observe how it affects fine-tuning performance. Top-1 accuracies for $k = 1, \dots, 8$ modules fine-tuned with step-sizes $s = 2$ and $s = 3$ are presented in Fig. 5 (left). We note that multi-phase fine-tuning (with $s = 2$ and $s = 3$) still outperforms single-phase fine-tuning. The number of required training epochs shown in Fig. 5 (right), shows that multi-phase fine-tuning converges faster also in this case, although the difference is not as significant as when comparing single-phase fine-tuning to using step-size $s = 1$. Applying a larger step-size $s = 2$ or $s = 3$ does not improve overall performance compared to $s = 1$. Since $k = 6$ is the only value that is comparable for step-sizes $s = 1, 2$ and 3, we compare the top-1 accuracy achieved by fine-tuning top-6 modules using the aforementioned step-sizes in Table 3. The smallest step-size achieves the best performance with the least training epochs.

4.4 Comparison of Training Progress

We examined the training progress by recording the validation loss as a function of the number of training epochs for the best-performing multi-phase fine-tuning with step-size $s = 1$ and single-phase fine-tuning. The results are shown in Fig. 6 for training $k = 3, 4, \dots, 8$ of the topmost modules[2]

[2] There was no significant difference between the loss graphs for single- and multi-phase fine-tuning for $k = 2$; their plots were thus omitted.

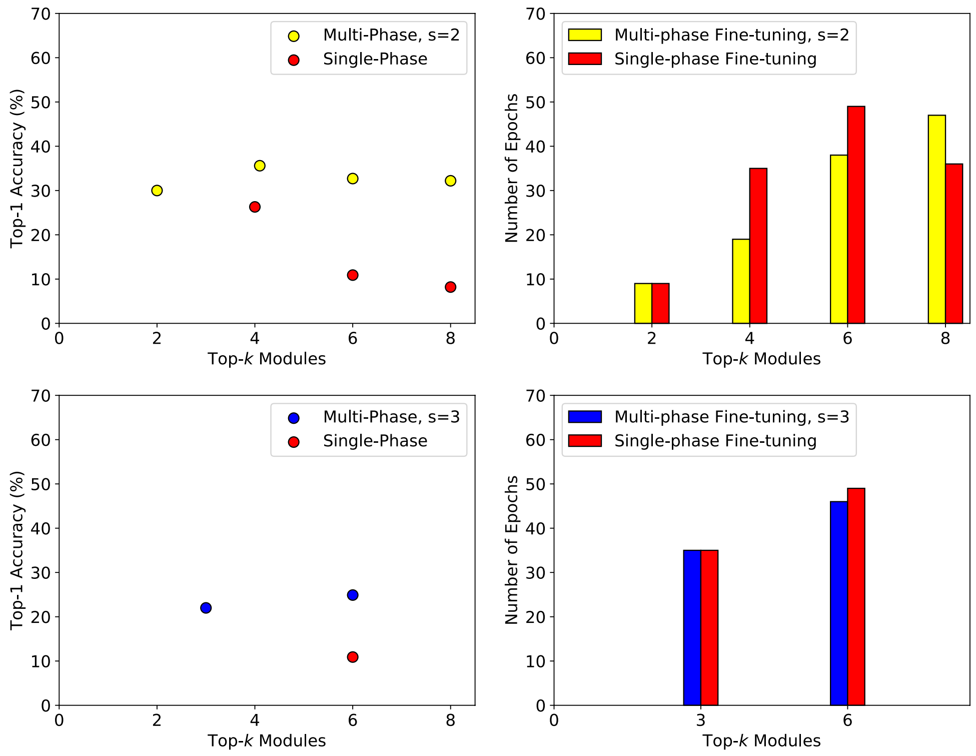

Fig. 5 Top: Top-1 accuracy (left) and total number of fine-tuning epochs (right) for single- and multi-phase fine-tuning with stepsize $s = 2$. Bottom: Top-1 accuracy (left) and total number of fine-tuning epochs (right) for single- and multi-phase fine-tuning with stepsize $s = 3$. Note: for $k = s = 2$ (top) and $k = s = 3$ (bottom), both approaches are equivalent

Table 3 Effect of step-size s for fine-tuning top-6 modules by multi-phase fine-tuning

Step-size s	Top-1 accuracy (%)	Epochs
1	**39.3**	**30**
2	32.7	36
3	24.9	49

The values in bold highlight the best-performing method

We observe that for most values of k, applying single-phase fine tuning results in a sharp increase in the validation loss before it starts to decrease. In contrast, multi-phase fine-tuning results in a consistently decreasing validation loss for all values of k. Although the same parameters are eventually trained by both approaches, we believe that dividing the training into multiple phases is beneficial as it allows smoother changing of the layer weights.

For example, consider the top-3 layers, indexed by $(L-2)$, $(L-1)$, and L, where L is the final layer of the network. By unfreezing all the layers at once, weights in layer $(L-2)$ can start to prematurely adapt to those in layers $(L-1)$ and L, which may still be far from the values they eventually converge to. Including more layers over several phases smooths abrupt changes in layer weights.

Results suggest that multi-phase fine-tuning can also provide an experimental way to decide how many layers should be fine-tuned. We can add more layers in phases and monitor the validation loss. As long as performance improvements are observed, we can continue fine-tuning more layers.

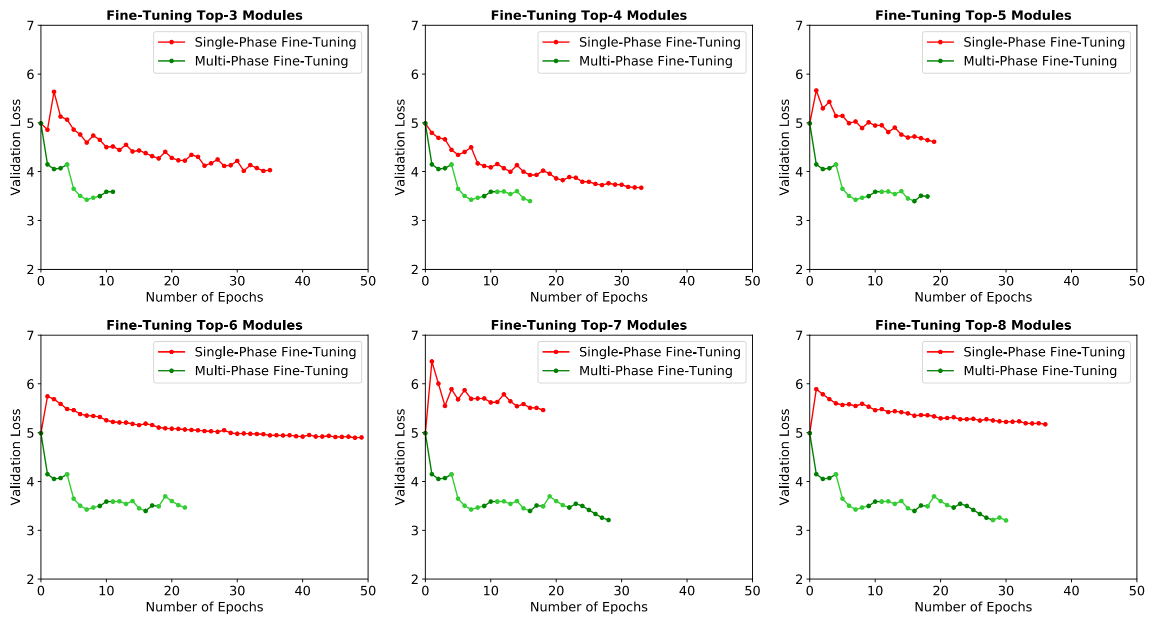

Fig. 6 Validation loss as a function of the number of training epochs when fine-tuning top-k modules of GoogLeNet using single-phase fine-tuning and multi-phase fine-tuning with step-size $s = 1$. Training was terminated using an early-stopping approach. Note that epoch 1 for all experiments is the first training epoch after training the classifying fully-connected layers

5 Conclusion

A key question in transfer learning is how many layers to fine-tune to take advantage of the generality of lower layers' features, while allowing the network to fit to the target task. We proposed multi-phase fine-tuning, starting by only fine-tuning the weights in the last fully-connected layer, and adding more layers in subsequent phases. We applied it to transfer learning from the domain of object recognition to SLR using one of the most commonly used network architectures, GoogLeNet. Results show that compared to earlier fine-tuning approaches, multi-phase fine-tuning has a higher classification accuracy and requires less training time for this pair of domains. In addition, it provides a constructive approach to decide how many layers' weights to fine-tune. Future work includes extending the work presented here into a complete continuous sign language recognition system working on sequences of gestures. We also aim to investigate the applicability of multi-phase fine-tuning in other domains beyond sign language recognition.

References

1. Azizpour H et al (2015) From generic to specific deep representations for visual recognition. In: CVPR DeepVision workshop, pp 36–45
2. Bengio Y (2012) Deep learning of representations for unsupervised and transfer learning. In: ICML workshop on unsupervised and transfer learning, pp 17–36
3. Camgoz NC et al (2020) Sign language transformers: joint end-to-end sign language recognition and translation. In: CVPR, pp 10023–10033
4. Cao C, Zhang Y, Wu Y, Lu H, Cheng J (2017) Egocentric gesture recognition using recurrent 3d convolutional neural networks with spatiotemporal transformer modules. In: ICCV, pp 3763–3771
5. Cihan Camgoz N et al (2017) Subunets: end-to-end hand shape and continuous sign language recognition. In: ICCV, pp 3056–3065
6. Cui R, Liu H, Zhang C (2017) Recurrent convolutional neural networks for continuous sign language recognition by staged optimization. In: CVPR, pp 1–4
7. Daroya R, Peralta D, Naval P (2018) Alphabet sign language image classification using deep learning. In: TENCON 2018-2018 IEEE region 10 conference, IEEE, pp 0646–0650
8. Forster J et al (2014) Extensions of the sign language recognition and translation corpus RWTH-PHOENIX-weather. In: International conference on language resources and evaluation, pp 1911–1916

9. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: International conference on artificial intelligence and statistics, pp 249–256
10. Koller O, Forster J, Ney H (2015) Continuous sign language recognition: towards large vocabulary statistical recognition systems handling multiple signers. Comput Vis Image Underst 141:108–125
11. Koller O et al (2016) Deep sign: hybrid CNN-HMM for continuous sign language recognition. In: BMVC, pp 136.1–136.12
12. Koller O et al (2017) Re-sign: re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMS. In: CVPR, pp 4297–4305
13. Li D et al (2020) Word-level deep sign language recognition from video: a new large-scale dataset and methods comparison. In: IEEE WACV, pp 1459–1469
14. Montone G, O'Regan JK, Terekhov AV (2015) The usefulness of past knowledge when learning a new task in deep neural networks. In: CoCo@ NIPS, pp 10–18
15. Montone G, O'Regan JK, Terekhov AV (2017) Gradual tuning: a better way of fine tuning the parameters of a deep neural network. arXiv preprint arXiv:1711.10177
16. Penatti OA, Nogueira K, dos Santos JA (2015) Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In: CVPR workshops, pp 44–51
17. Pigou L et al (2014) Sign language recognition using convolutional neural networks. In: ECCV workshops ChaLearn looking at people, pp 572–578
18. Russakovsky O et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
19. Sarhan N, Frintrop S (2020) Transfer learning for videos: from action recognition to sign language recognition. In: IEEE ICIP, IEEE, pp 1811–1815
20. Sharif Razavian A et al (2014) CNN features off-the-shelf: an astounding baseline for recognition. In: CVPR workshop, pp 806–813
21. Szegedy C et al (2015) Going deeper with convolutions. In: CVPR, pp 1–9
22. Szegedy C et al (2016) Rethinking the inception architecture for computer vision. In: CVPR, pp 2818–2826
23. Tajbakhsh N et al (2016) Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans Med Imaging 35(5):1299–1312
24. Tolentino LKS et al (2019) Static sign language recognition using deep learning. Int J Mach Learn Comput 9(6):821–827
25. Wadhawan A, Kumar P (2020) Deep learning-based sign language recognition system for static signs. Neural Comput Appl 32:7957–7968
26. Yosinski J, Clune J, Bengio Y, Lipson H (2014) How transferable are features in deep neural networks? In: NIPS, pp 3320–3328