Greedy Layerwise Learning Can Scale to ImageNet

Eugene Belilovsky [1] **Michael Eickenberg** [2] **Edouard Oyallon** [3]

Abstract

Shallow supervised 1-hidden layer neural networks have a number of favorable properties that make them easier to interpret, analyze, and optimize than their deep counterparts, but lack their representational power. Here we use 1-hidden layer learning problems to sequentially build deep networks layer by layer, which can inherit properties from shallow networks. Contrary to previous approaches using shallow networks, we focus on problems where deep learning is reported as critical for success. We thus study CNNs on image classification tasks using the large-scale ImageNet dataset and the CIFAR-10 dataset. Using a simple set of ideas for architecture and training we find that solving sequential 1-hidden-layer auxiliary problems lead to a CNN that exceeds AlexNet performance on ImageNet. Extending this training methodology to construct individual layers by solving 2-and-3-hidden layer auxiliary problems, we obtain an 11-layer network that exceeds several members of the VGG model family on ImageNet, and can train a VGG-11 model to the same accuracy as end-to-end learning. To our knowledge, this is the first competitive alternative to end-to-end training of CNNs that can scale to ImageNet. We illustrate several interesting properties of these models and conduct a range of experiments to study the properties this training induces on the intermediate representations.

1. Introduction

Deep Convolutional Neural Networks (CNNs) trained on large-scale supervised data via the back-propagation algorithm have become the dominant approach in most computer vision tasks (Krizhevsky et al., 2012). This has motivated successful applications of deep learning in other fields such as speech recognition (Chan et al., 2016), natural language processing (Vaswani et al., 2017), and reinforcement learning (Silver et al., 2017). Training procedures and architecture choices for deep CNNs have become more and more entrenched, but which of the standard components of modern pipelines are essential to the success of deep CNNs is not clear. Here we ask: do CNN layers need to be learned jointly to obtain high performance? We will show that even for the challenging ImageNet dataset the answer is *no*.

Supervised end-to-end learning is the standard approach to neural network optimization. However it has potential issues that can be valuable to consider. First, the use of a global objective means that the final functional behavior of individual intermediate layers of a deep network is only indirectly specified: it is unclear how the layers work together to achieve high-accuracy predictions. Several authors have suggested and shown empirically that CNNs learn to implement mechanisms that progressively induce invariance to complex, but irrelevant variability (Mallat, 2016; Yosinski et al., 2015) while increasing linear separability (Zeiler & Fergus, 2014; Oyallon, 2017; Jacobsen et al., 2018) of the data. Progressive linear separability has been shown empirically but it is unclear whether this is merely the consequence of other strategies implemented by CNNs, or if it is a sufficient condition for the high performance of these networks. Secondly, understanding the link between shallow Neural Networks (NNs) and deep NNs is difficult: while generalization, approximation, or optimization results (Barron, 1994; Bach, 2014; Venturi et al., 2018; Neyshabur et al., 2018; Pinkus, 1999) for 1-hidden layer NNs are available, the same studies conclude that multiple-hidden-layer NNs are much more difficult to tackle theoretically. Finally, end-to-end back-propagation can be inefficient (Jaderberg et al., 2016; Salimans et al., 2017) in terms of computation and memory resources and is considered not biologically plausible.

Sequential learning of CNN layers by solving shallow supervised learning problems is an alternative to end-to-end back-propagation. This classic (Ivakhnenko & Lapa, 1965) learning principle can directly specify the objective of every layer. It can encourage the refinement of specific properties of the representation (Greff et al., 2016), such as progressive linear separability. The development of theoretical tools for deep greedy methods can naturally draw from the theoretical understanding of shallow sub-problems. Indeed, (Arora

[1]Mila, University of Montreal [2]University of California, Berkeley [3]CentraleSupelec, University of Paris-Saclay / INRIA Saclay. Correspondence to: Eugene Belilovsky <eugene.belilovsky@umontreal.ca>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

et al., 2018; Bengio et al., 2006; Bach, 2014; Janzamin et al., 2015) show global optimal approximations, while other works have shown that networks based on sequential 1-hidden layer training can have a variety of guarantees under certain assumptions (Huang et al., 2017; Malach & Shalev-Shwartz, 2018; Arora et al., 2014): greedy layerwise methods could permit to cascade those results to bigger architectures. Finally, a greedy approach will rely much less on having access to a full gradient. This can have a number of benefits. From an algorithmic perspective, they do not require storing most of the intermediate activations nor to compute most intermediate gradients. This can be beneficial in memory-constrained settings. Unfortunately, prior work has not convincingly demonstrated that layer-wise training strategies can tackle the sort of large-scale problems that have brought deep learning into the spotlight.

Recently multiple works have demonstrated interest in determining whether alternative training methods (Xiao et al., 2019; Bartunov et al., 2018) can scale to large data-sets that have only been solved by deep learning. As is the case for many algorithms (not just training strategies) many of these alternative training strategies have been shown to work on smaller datasets (e.g. MNIST and CIFAR) but fail completely on large-scale datasets (e.g. ImageNet). Also, these works largely focus on avoiding the weight transport problem in backpropagation (Bartunov et al., 2018) while simple greedy layer-wise learning reduces the extent of this problem and should be considered as a potential baseline.

In this context, our contributions are as follows. **(a)** First, we design a simple and scalable supervised approach to learn layer-wise CNNs in Sec. 3. **(b)** Then, Sec. 4.1 demonstrates empirically that by sequentially solving 1-hidden layer problems, we can match the performance of the AlexNet on ImageNet. We motivate in Sec. 3.3 how this model can be connected to a body of theoretical work that tackles 1-hidden layer networks and their sequentially trained counterparts. **(c)** We show that layerwise trained layers exhibit a progressive linear separability property in Sec. 4.2. **(d)** In particular, we use this to help motivate learning layer-wise CNN layers via shallow k-hidden layer auxiliary problems, with $k > 1$. Using this approach our sequentially trained 3-hidden layer models can reach the performance level of VGG models (Sec. 4.3) and end-to-end learning. **(e)** Finally, we suggest an approach to easily reduce the model size *during training* of these networks.

2. Related Work

Several authors have previously studied layerwise learning. In this section we review related works and re-emphasize the distinctions from our work.

Greedy unsupervised learning has been a popular topic of research in the past. Greedy unsupervised learning of deep generative models (Bengio et al., 2007; Hinton et al., 2006) was shown to be effective as an initialization for deep supervised architectures. Bengio et al. (2007) also considered supervised greedy layerwise learning as *initialization* of networks for subsequent end-to-end supervised learning, but this was not shown to be effective with the existing techniques at the time. Later work on large-scale supervised deep learning showed that modern training techniques permit avoiding layerwise initialization entirely (Krizhevsky et al., 2012). We emphasize that the supervised layerwise learning we consider is distinct from unsupervised layerwise learning. Moreover, here layerwise training is not studied as a *pretraining* strategy, but a *training* one.

Layerwise learning in the context of constructing supervised NNs has been attempted in several works. It was considered in multiple earlier works Ivakhnenko & Lapa (1965); Fahlman & Lebiere (1990b); Lengellé & Denoeux (1996) on very simple problems and in a climate where deep learning was not a dominant supervised learning approach. These works were aimed primarily at structure learning, building up architectures that allow the model to grow appropriately based on the data. Others works were motivated by the avoidance of difficulties with vanishing gradients. Similarly, Cortes et al. (2016) recently proposed a progressive learning method that builds a network such that the architecture can adapt to the problem, with theoretical contributions to structure learning, but not on problems where deep networks are unmatched in performance. Malach & Shalev-Shwartz (2018) also train a supervised network in a layerwise fashion, showing that their method provably generalizes for a restricted class of image models. However, the results of these model are not shown to be competitive with handcrafted approaches (Oyallon & Mallat, 2015). Similarly (Kulkarni & Karande, 2017; Marquez et al., 2018) revisit layerwise training, but in a limited experimental setting.

Huang et al. (2017) combined boosting theory with a residual architecture (He et al., 2016) to sequentially train layers. However, results are presented for limited datasets and indicate that the end-to-end approach is often needed ultimately to obtain competitive results. This proposed strategy does not clearly outperform simple non-deep-learning baselines. By contrast, we focus on settings where deep CNN based-approaches do not currently have competitors and rely on a simpler objective function, which is found to scale well and be competitive with end-to-end approaches.

Another related thread is methods which add layers to existing networks and then use end-to-end learning. These approaches usually have different goals from ours, such as stabilizing end-to-end learned models. Brock et al. (2017) builds a network in stages, where certain layers are progressively frozen, permitting faster training. Mosca & Magoulas

(2017); Wang et al. (2017) propose methods that progressively stack layers, performing end-to-end learning on the resulting network at each step. A similar strategy was applied for training GANs in Karras et al. (2017). By the nature of our goals in this work, we never perform fine-tuning of the whole network. Several methods also consider auxiliary supervised objectives (Lee et al., 2015) to stabilize end-to-end learning, but this is different from the case where these objectives are not solved jointly.

3. Supervised Layerwise Training of CNNs

In this section we formalize the architecture, training algorithm, and the necessary notations and terminology. We focus on CNNs, with ReLU non-linearity denoted by ρ. Sec. 3.1 describes a layerwise training scheme using a succession of auxiliary learning tasks. We add one layer at a time: the first layer of a k-hidden layer CNN problem. Finally, we discuss the distinctions in varying k.

3.1. Architecture Formulation

Our architecture has J blocks (see Fig. 1), which are trained in succession. From an input signal x, an initial representation $x_0 \triangleq x$ is propagated through j convolutions, giving x_j. Each x_j feeds into an *auxiliary classifier* to obtain prediction z_j, which computes an intermediate classification output. At depth j, denote by W_{θ_j} a convolutional operator with parameters θ_j , C_{γ_j} an auxiliary classifier with all its parameters denoted γ_j, and P_j a down-sampling operator. The parameters correspond to 3×3 kernels with bias terms. Formally, from layer x_j we iterate as follows:

$$\begin{cases} x_{j+1} = \rho W_{\theta_j} P_j x_j \\ z_{j+1} = C_{\gamma_j} x_{j+1} \in \mathbb{R}^c \end{cases} \tag{1}$$

where c is the number of classes. For the pooling operator P we choose the *invertible downsampling* operation described in Dinh et al. (2017), which consists in reorganizing the initial spatial channels into the 4 spatially decimated copies obtainable by 2×2 spatial sub-sampling, reducing the resolution by a factor 2. We decided against strided pooling, average pooling, and the non-linear max-pooling, because these strongly encourage loss of information. As is standard practice, P is applied at certain layers ($P_j = P$), but not others ($P_j = Id$). The CNN classifier C_{γ_j} is given by:

$$C_{\gamma_j} x_j = \begin{cases} LAx_j & \text{for } k = 1 \\ LA\rho\tilde{W}_{k-2}...\rho\tilde{W}_0 x_j & \text{for } k > 1 \end{cases} \tag{2}$$

where $\tilde{W}_0, ..., \tilde{W}_{k-2}$ are convolutional layers with constant width, A is a spatial averaging operator, and L a linear operator whose output dimension is c. The averaging operation is important for maintaining scalability at early layers. For $k = 1$, C_{γ_j} is a linear model. In this case our architecture is trained by a sequence of 1-hidden layer CNN.

3.2. Training by Auxiliary Problems

Our training procedure is layerwise: at depth j, while keeping all other parameters fixed, θ_j is obtained via an *auxiliary problem*: optimizing $\{\theta_j, \gamma_j\}$ to obtain the best training accuracy for *auxiliary classifier* C_{γ_j}. We formalize this idea for a training set $\{x^n, y^n\}_{n \leq N}$: For a function $z(\cdot; \theta, \gamma)$ parametrized by $\{\theta, \gamma\}$ and a loss l (e.g. cross entropy), we consider the classical minimization of the empirical risk: $\hat{\mathcal{R}}(z; \theta, \gamma) \triangleq \frac{1}{N} \sum_n l(z(x^n; \theta, \gamma), y^n)$.

At depth j, assume we have constructed the parameters $\{\hat{\theta}_0, ..., \hat{\theta}_{j-1}\}$. Our algorithm can produce samples $\{x_j^n\}$. Taking $z_{j+1} = z(x_j^n; \theta_j, \gamma_j)$, we employ an optimization procedure that aims to minimize the risk $\hat{\mathcal{R}}(z_{j+1}; \theta_j, \gamma_j)$. This procedure (Alg. 1) consists in training (e.g. using SGD) the shallow CNN classifier C_j on top of x_j, to obtain the new parameter $\hat{\theta}_{j+1}$. Under mild conditions, it improves the training error at each layer as shown below:

Proposition 3.1 (Progressive improvement)**.** *Assume that $P_j = Id$. Then there exists $\tilde{\theta}$ such that:*
$\hat{\mathcal{R}}(z_{j+1}; \hat{\theta}_j, \hat{\gamma}_j) \leq \hat{\mathcal{R}}(z_{j+1}; \tilde{\theta}, \hat{\gamma}_{j-1}) = \hat{\mathcal{R}}(z_j; \hat{\theta}_{j-1}, \hat{\gamma}_{j-1})$.

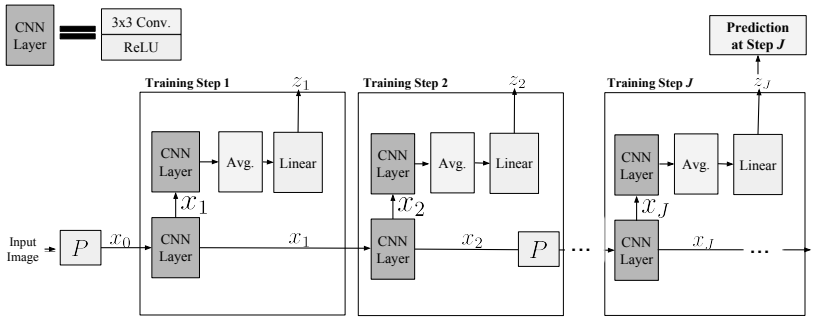
A technical requirement for the actual optimization procedure is to not produce a worse objective than the initialization, which can be achieved by taking the best result along the optimization trajectory.

The cascade can inherit from the individual properties of each auxiliary problem. For instance, as ρ is 1-Lipschitz, if each $W_{\hat{\theta}_j}$ is 1-Lipschitz then so is x_J w.r.t. x. Another example is the nested objective defined by Alg. 1: the optimality of the solution will be largely governed by the optimality of the sub-problem solver. Specifically, if the auxiliary problem solution is close to optimal then the solution of Alg. 1 will be close to optimal.

Proposition 3.2. *Assume the parameters $\{\theta_0^*, ..., \theta_{J-1}^*\}$ are obtained via a optimal layerwise optimization procedure. We assume that $W_{\theta_j^*}$ is 1-lipschitz without loss of generality and that the biases are bounded uniformly by B. Given an input function $g(x)$, we consider functions of the type $z_g(x) = C_\gamma \rho W_\theta g(x)$. For $\epsilon > 0$, we call $\theta_{\epsilon,g}$ the parameter provided by a procedure to minimize $\hat{\mathcal{R}}(z_g; \theta; \gamma)$ which leads to a 1-lipschitz operator that satisfies:*
1. $\underbrace{\|\rho W_{\theta_{\epsilon,g}} g(x) - \rho W_{\theta_{\epsilon,\tilde{g}}} \tilde{g}(x)\| \leq \|g(x) - \tilde{g}(x)\|}_{\text{(stability)}}, \forall g, \tilde{g},$
2. $\underbrace{\|W_{\theta_j^*} x_j^* - W_{\theta_{\epsilon,x_j^*}} x_j^*\| \leq \epsilon(1 + \|x_j^*\|)}_{\text{(ϵ-approximation)}},$
with, $\hat{x}_{j+1} = \rho W_{\theta_{\epsilon,\hat{x}_j}} \hat{x}_j$ and $x_{j+1}^* = \rho W_{\theta_j^*} x_j^*$ with $x_0^* = \hat{x}_0 = x$, then, we prove by induction:

$$\|x_J^* - \hat{x}_J\| = \mathcal{O}(J^2 \epsilon) \tag{3}$$

The proof can be found in the Appendix. This demonstrates an example of how the training strategy can permit to extend



Algorithm 1 Layer Wise CNN

Input: Training samples $\{x_0^n, y^n\}_{n \leq N}$
for $j \in 0..J-1$ **do**
  Compute $\{x_j^n\}_{n \leq N}$ (via Eq.(1))
  $(\theta_j^*, \gamma_j^*) = \arg\min_{\theta_j, \gamma_j} \hat{\mathcal{R}}(z_{j+1}; \theta_j, \gamma_j)$
end for

Figure 1. High level diagram of the layerwise CNN learning experimental framework using a $k = 2$-hidden layer. P, the down-sampling (see Figure 2 (Jacobsen et al., 2018)) , is applied at the input image as well as at $j = 2$.

results from shallow CNNs to deeper CNNs, in particular for $k = 1$. Applying an existing optimization strategy could give us a bound on the solution of the overall objective of Alg. 1, as we will discuss below.

3.3. Auxiliary Problems & The Properties They Induce

We now discuss the properties arising from the auxiliary problems. We start with $k = 1$, for which the auxiliary classifier consists of only the linear A and L operators. Thus, the optimization aims to obtain the weights of a 1-hidden layer NN. For this case, as discussed in Sec. 1, a variety of theoretical results exist (e.g. (Cybenko, 1989; Barron, 1994)). Moreover, (Arora et al., 2018; Ge et al., 2017; Du & Goel, 2018; Bach, 2014) proposed provable optimization strategies for this case. Thus the analysis and optimization of the 1-hidden layer problem is a case that is relatively well understood compared to deep counterparts. At the same time, as shown in Prop. 3.2, applying an existing optimization strategy could give us a bound on the solution of the overall objective of Alg. 1. To build intuition let us consider another example where the analysis can be simplified for $k = 1$ training. Recall the classic least square estimator (Barron, 1994) of a 1-hidden layer network:

$$(\hat{L}, \hat{\theta}) = \arg \inf_{(L,\theta)} \sum_n \|f(x^n) - L\rho W_\theta x^n\|^2 \qquad (4)$$

where f is the function of interest. Following a suggestion from (Mallat, 2016) (detailed in Appendix A) we can state there exists a set Ω_j and f_j, $\forall x \in \Omega_j, f(x) = f_j \circ \rho W_{\hat{\theta}_{j-1}} ... \rho W_{\hat{\theta}_1} x$ where $\{\hat{\theta}_{j-1}, ..., \hat{\theta}_1\}$ are the parameters of greedily trained layers (with width α_j) and ρ is sigmoidal. For simplicity, let us assume that $x^n \in \Omega_j, \forall n$. It implies that at step j of a greedy training procedure with $k = 1$, the corresponding sample loss is:

$$\|f(x^n) - z(x_j^n; \theta_j, \gamma_j)\|^2 = \|f_j(x_j^n) - z(x_j^n; \theta_j, \gamma_j)\|^2 .$$

In particular, the right term is shaped as Eq. (4) and thus we can apply standard bounds available only for 1-hidden layer settings (Barron, 1994; Janzamin et al., 2015). In the case of jointly learning the layers we could not make this kind of formulation. For example if one now applied the algorithm of (Janzamin et al., 2015) and their Theorem 5 it would give the following risk bound:

$$\mathbb{E}_{X_j}[\|f_j(X_j) - z(X_j; \hat{\theta}_j, \hat{\gamma}_j)\|^2] \leq \mathcal{O}\left(C_{f_j}^2 (\frac{1}{\sqrt{\alpha_j}} + \delta_\rho)^2\right) + \mathcal{O}(\epsilon^2) ,$$

where $X_j = \rho W_{\hat{\theta}_{j-1}} ... \rho W_{\hat{\theta}_1} X_0$ is obtained from an initial bounded distribution X_0, C_{f_j} is the Barron Constant (as described in (Lee et al., 2017)) of f_j, δ_ρ a constant depending on ρ and ϵ an estimation error. Furthermore, if we fit the f_j layerwise using a method such as (Janzamin et al., 2015), we reach an approximate optimum for each layer given the state of the previous layer. Under Prop 3.2, we observe that small errors at each layer, even taken cumulatively, will not affect the final representation learned by the cascade. Observe that if C_{f_j} decreases with j, the approximation bound on f_j will correspondingly improve.

Another view of the $k = 1$ training using a standard classification objective: the optimization of the 1-hidden layer network will encourage the hidden layer outputs to make its classification output maximally linearly separable with respect to its inputs. Specializing Prop. 3.1 for this case shows that the layerwise $k = 1$ procedure will try to progressively improve the linear separability. Progressive linear separation has been empirically studied in end-to-end CNNs (Zeiler & Fergus, 2014; Oyallon, 2017) as an indirect consequence, while the $k = 1$ training permits us to study this basic principle more directly as the layer objective. Concurrent work (Elad et al., 2019) follows the same argument to use a layerwise training procedure to evaluate mutual information more directly.

Unique to our layerwise learning formulation, we consider the case where the auxiliary learning problem involves auxiliary hidden layers. We will interpret, and empirically verify, in Sec. 4.2 that this builds layers that are progressively better inputs to shallow CNNs. We will also show

a link to building, in a more progressive manner, linearly separable layers. Considering only shallow (with respect to total depth) auxiliary problems (e.g. $k = 2, 3$ in our work) we can maintain several advantages. Indeed, optimization for shallower networks is generally easier, as we can for example diminish the vanishing gradient problem, reducing the need for identity loops or normalization techniques (He et al., 2016). Two and three hidden layer networks are also appealing for extending results from one hidden layer (Allen-Zhu et al., 2018) as they are the next natural member in the family of NNs.

4. Experiments and Discussion

We performed experiments on the large-scale ImageNet-1k (Russakovsky et al., 2015), a major catalyst for the recent popularity of deep learning, as well as the CIFAR-10 dataset. We study the classification performance of layerwise models with $k = 1$, comparing them to standard benchmarks and other sequential learning methods. Then we inspect the representations built through our auxiliary tasks and motivate the use of models learned with auxiliary hidden layers $k > 1$, which we subsequently evaluate at scale.

We highlight that many algorithms do not scale to large datasets (Bartunov et al., 2018) like ImageNet. For example a SOTA hand-crafted image descriptor combined with a linear model achieves 82% on CIFAR-10 while only 17.4% on ImageNet (Oyallon et al., 2018). 1-hidden layer CNNs from our experiments can obtain 61% accuracy on CIFAR-10 while the same CNN results on ImageNet only gives 12.3% accuracy. Alternative learning methods for deep networks can sometimes show similar behavior, highlighting the importance of assessing their scalability. For example Feedback Alignment (Lillicrap et al., 2016) which is able to achieve 63% on CIFAR-10 compared to 68% with backprop, on the 1000 class ImageNet obtains only 6.6% accuracy compared to 50.9% (Bartunov et al., 2018; Xiao et al., 2019). Thus, based on these observations and the sparse and small-scale experimental efforts of related works on greedy layerwise learning it is entirely unclear whether this family of approaches can work on large datasets like ImageNet and be used to construct useful models. Noting that ImageNet models not only represent benchmarks but have generic features (Yosinski et al., 2014) and thus AlexNet- and VGG-like accuracies for CNN's on this dataset typically indicate representations that are generic enough to be useful for downstream applications.

Naming Conventions We call M the number of feature maps of the first convolution of the network and $\tilde{M}$ the number of feature maps of the first convolution of the auxiliary classifiers. This fully defines the width of all the layers, since input width and output width are equal unless the layer has downsampling, in which case the output width is twice the input width. Finally, A is chosen to average over the four spatial quadrants, yielding a 2×2-shaped output. Spatial averaging before the linear layer is common in ResNets (He et al., 2016) to reduce size. In our case this is critical to permit scalability to large image sizes at early layers of layer-wise training. For computational reasons on ImageNet, an invertible downsampling is also applied (reducing the signal to output 12×112^2). We also construct an ensemble model, which consists of a weighted average of all auxiliary classifier outputs, i.e. $Z = \sum_{j=1}^{J} 2^j z_j$. Our sandbox architectures to study layerwise learning end with a final auxiliary network that becomes part of the final model. In some cases we may want to use a different final auxiliary network after training the layer. We will use $\tilde{M}_f$ to denote the width of the final auxiliary CNN. For shorthand we will denote our simple CNN networks SimCNN.

We briefly introduce the datasets and preprocessing. CIFAR-10 consists of small RGB images with respectively $50k$ and $10k$ samples for training and testing. We use the standard data augmentation and optimize each layer with SGD using a momentum of 0.9 and a batch-size of 128. The initial learning rate is 0.1 and we use the reduced schedule with decays of 0.2 every 15 epochs (Zagoruyko & Komodakis, 2016), for a total of 50 epochs in each layer. ImageNet consists of $1.2M$ RGB images of varying size for training. Our data augmentation consists of random crops of size 224^2. At testing time, the image is rescaled to 256^2 then cropped at size 224^2. We used SGD with momentum 0.9, weight-decay of 10^{-4} for a batch size of 256. The initial learning rate is 0.1 (He et al., 2016) and we use the reduced schedule with decays of 0.1 every 20 epochs for 45 epochs. We use 4 GPUs to train our ImageNet models.

4.1. AlexNet Accuracy with 1-Hidden Layer Auxiliary Problems

We consider the atomic, layerwise CNN with $k = 1$ which corresponds to solving a sequence of 1-hidden layer CNN problems. As discussed in Sec. 2, previous attempts at supervised layerwise training (Fahlman & Lebiere, 1990a; Arora et al., 2014; Huang et al., 2017; Malach & Shalev-Shwartz, 2018), which rely solely on sequential solving of shallow problems have yielded performance well below that of typical deep learning models even on the CIFAR-10 dataset. We show, surprisingly, that it is possible to go beyond the AlexNet performance barrier (Krizhevsky et al., 2012) without end-to-end backpropagation on ImageNet with elementary auxiliary problems. To emphasize the stability of the training process we do not apply any batch-normalization to this model.

CIFAR-10. We trained a model with $J = 5$ layers, downsampling at layers $j = 2, 3$, and layer sizes starting at $M = 256$. We obtain 88.3% and note that this accuracy is close to the AlexNet model performance (Krizhevsky et al., 2012) for CIFAR-10 (89.0%). Comparisons in Table

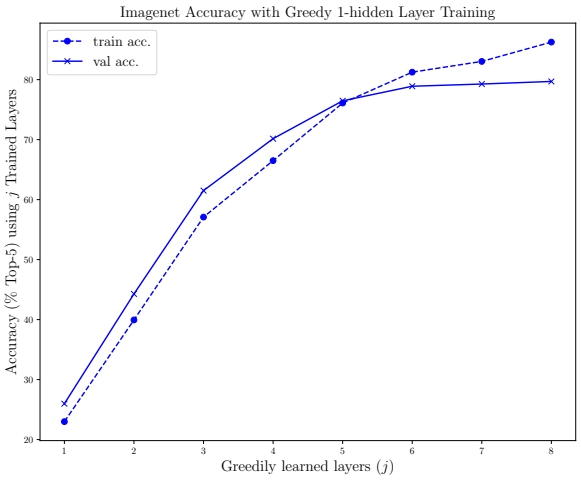


Figure 2. One-hidden layer trained ImageNet model shows rapid progressive improvement

2 show that other sequentially trained 1-hidden layer networks have yielded performances that do not exceed those of the top hand-crafted methods or those using unsupervised learning. To the best of our knowledge they obtain 82.0% accuracy (Huang et al., 2017). The end-to-end version of this model obtains 89.7%, while using $5\times$ more GPU memory than $k = 1$ training.

ImageNet. Our model is trained with $J = 8$ layers and downsampling operations at layers $j = 2, 3, 4, 6$. Layer sizes start at $M = 256$. Our final trained model achieves **79.7%** top-5 single crop accuracy on the validation set and **80.8%** a weighted ensemble of the layer outputs. In addition to exceeding AlexNet, this model compares favorably to alternatives to end-to-end supervised CNNs including hand-crafted computer vision techniques (Sánchez et al., 2013). Full results are shown in Table 3 where we also highlight the substantial performance gaps to multiple alternative training methods. Figure 2 shows the per-layer classification accuracy. We observe a remarkably rapid progression (near linear in the first 5 layers) that takes the accuracy from $\sim 23\%$ to $\sim 80\%$ top-5. We leave a full theoretical explanation of this fast rate as an open question. Finally, in Appendix we demonstrate that this model maintains the transfer learning properties of deep networks trained on ImageNet.

We also note that our final training accuracy is relatively high for ImageNet (87%), which indicates that appropriate regularization may lead to a further improvement in test accuracy. We now look at empirical properties induced in the layers and subsequently evaluate the distinct $k > 1$.

4.2. Empirical Separability Properties

We study the intermediate representations generated by the layerwise learning procedure in terms of linear separability as well as separability by a more general set of classifiers. Our aims are **(a)** to determine empirically whether $k = 1$ indeed progressively builds more and more linearly separable data representations and **(b)** to determine how linear separability of the representations evolves for networks constructed with $k > 1$ auxiliary problems. Finally we ask whether the notion of building progressively better inputs to a linear model ($k = 1$ training) has an analogous counterpart for $k > 1$: building progressively better inputs for shallow CNNs (discussed in Sec 3.3).

We define *linear separability* of a representation as the maximum accuracy achievable by a linear classifier. Further we define the notion of *CNN*-p-*separability* as the accuracy achieved by a p-layer CNN trained on top of the representation to be assessed.

We focus on CNNs trained on CIFAR-10 without downsampling. Here, $J = 5$ and the layer sizes follow $M = 64, 128, 256$. The auxiliary classifier feature map size, when applicable, is $\tilde{M} = 256$. We train with 5 random initializations for each network and report an average standard deviation of 0.58% test accuracy. Each layer is evaluated by training a one-versus-rest logistic regression, as well as $p = 1, 2$-hidden-layer CNN on top of these representations. Because the linear representation has been optimized for it, we spatially average to a 2×2 shape before feeding them to the separability evaluation. Fig. 3 shows the results of each of these evaluations plotting test set accuracy curves as a function of NN depth for each of the 3 evaluations. For these plots we averaged over initial layer sizes M and classifier layer sizes $\tilde{M}$ and random seeds. Each individual curve closely resembles these average curves, with slight shifts in the y-axis.

As expected from Sec. 3.3, linear separability monotonically increases with depth for $k = 1$. Interestingly, linear separability also improves in the case of $k > 1$, even though it is not directly specified by the auxiliary problem objective. At earlier layers, linear separation capability of models trained with $k = 1$ increases fastest as a function of layer depth compared to models trained with deeper auxiliary networks, but flattens out to a lower asymptotic linear separability at deeper layers. Thus, the simple principle of the $k = 1$ objective that tries to produce the maximal linear separation at each layer might not be an optimal strategy for achieving progressive linear separation.

We also notice that the deeper the auxiliary classifier, the slower the increase in linear separability initially, but the higher the linear separability at deeper layers. From the two right diagrams we also find that the CNN-p-separability progressively improves - but much more so for $k > 1$ trained networks. This shows that linear separability of a layer is not the sole criterion for rendering a representation a good "input" for a CNN. It further shows that our sequential training procedure for the case $k > 1$ can build a representation

Layer-wise Trained	Acc. (Ens.)
SimCNN ($k = 1$ train)	88.3 (88.4)
SimCNN ($k = 2$ train)	90.4 (90.7)
SimCNN($k = 3$ train)	91.7 (**92.8**)
BoostResnet (Huang et al., 2017)	82.1
ProvableNN (Malach et al., 2018)	73.4
(Mosca et al., 2017)	81.6
Reference e2e	
AlexNet	89
VGG [1]	92.5
WRN 28-10 (Zagoruyko et al. 2016)	**96.0**
Alternatives	[Ref.]
Scattering + Linear	82.3
FeedbackAlign (Bartunov et al., 2018)	62.6 [67.6]

Table 2. Results on CIFAR-10. Compared to the few existing methods using *only* layerwise training schemes we report much more competitive results to well known benchmark models that like ours do not use skip connnections.In brackets e2e trained version of the model is shown when available.

that is progressively a better input to a shallow CNN.

4.3. Scaling up Layerwise CNNs with 2 and 3 Hidden Layer Auxiliary Problems

We study the training of deep networks with $k = 2, 3$ hidden layer auxiliary problems. We limit ourselves to this setting to keep the auxiliary NNs shallow with respect to the network depth. We employ widths of $M = 128$ and $\tilde{M} = 256$ for both CIFAR-10 and ImageNet. For CIFAR-10, we use $J = 4$ layers and a down-sampling at $j = 2$. For ImageNet we closely follow the VGG architectures, which with their 3×3 convolutions and absence of skip-connections bear strong similarity to ours. We use $J = 8$ layers. As we start at halved resolution we do only 3 down-samplings at $j = 2, 4, 6$. Unlike the $k = 1$ case we found it helpful to employ batch-norm for these auxiliary problems.

We report our results for $k = 2, 3$ in Table 2 (CIFAR-10) and Table 3 (ImageNet) along with our $k = 1$ model. The reference model accuracies for ImageNet AlexNet, VGG, and ResNet use the same input sizes and single crop evaluation[1]. As expected from the previous section, the transition from $k = 1$ to $k = 2, 3$ improves the performances substantially. We compare our CIFAR-10 results to other sequentially trained propositions in the literature. Our methods exceed these in performance by a large margin. While the ensemble model of $k = 3$ surpasses the VGG, the other sequential models perform do not exceed unsupervised methods. No alternative sequential models are available for ImageNet. We thus compare our results on ImageNet to the standard reference CNNs and the best-performing alternatives to end-to-end Deep CNNs. Our $k = 3$ layerwise ensemble model

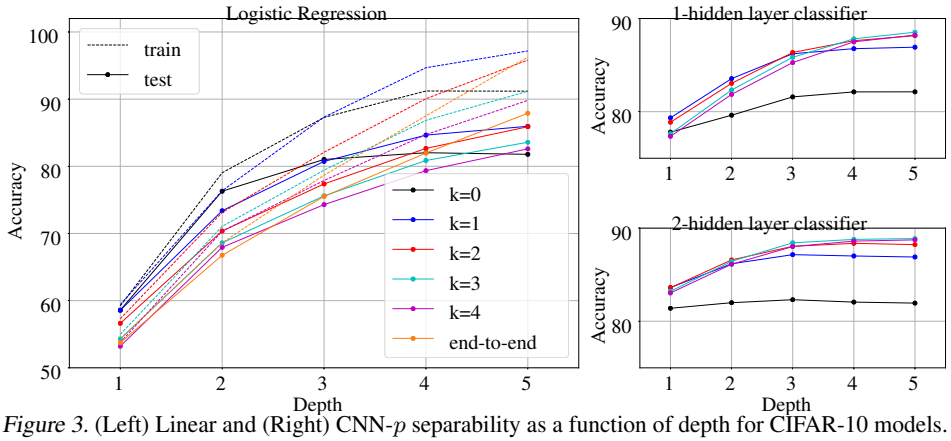
[1]Accuracies from http://torch.ch/blog/2015/07/30/cifar.html and https://pytorch.org/docs/master/torchvision/models.html.

	Top-1 (Ens.)	Top-5 (Ens.)
SimCNN ($k = 1$ train)	58.1 (59.3)	79.7 (80.8)
SimCNN ($k = 2$ train)	65.7 (67.1)	86.3 (87.0)
SimCNN ($k = 3$ train)	69.7 (71.6)	88.7 (89.8)
VGG-11 ($k = 3$ train)	67.6 (70.1)	88.0 (89.2)
VGG-11 (e2e train)	67.9	88.0
Alternative	[Ref.]	[Ref.]
DTargetProp (Bartunov et al., 2018)	1.6 [28.6]	5.4 [51.0]
FeedbackAlign (Xiao et al., 2019)	6.6 [50.9]	16.7 [75.0]
Scat. + Linear (Oyallon et al., 2018)	17.4	N/A
Random CNN	12.9	N/A
FV + Linear (Sánchez et al., 2013)	54.3	74.3
Reference e2e CNN		
AlexNet	56.5	79.1
VGG-13	69.9	89.3
VGG-19	72.9	90.9
Resnet-152	78.3	94.1

Table 3. Single crop validation acc. on ImageNet. Our SimCNN models use $J = 8$. In parentheses see the ensemble prediction. Layer-wise models are competitive with well known ImageNet benchmarks that similarly don't use skip connections. $k = 3$ training can yield equal performance to end to end on VGG-11. We highlight many methods and alternative training do not work at all on ImageNet. In brackets, e2e acc. is shown when available.

achieves **89.8%** top-5 accuracy, which is comparable to VGG-13 and largely exceeds AlexNet performance. Recall that $\tilde{M}_f$ denotes the width of the final auxiliary network, which becomes part of the model. In the experiments above this is relatively large ($\tilde{M}_f = 2048$), while the final layers have diminishing returns. To more effectively incorporate the final auxiliary network into the model architecutre, we reduce the width of the final *auxiliary* network for the $k = 3$ recomputing the $j = 7$ step. We use auxiliary networks of size $\tilde{M}_f = 512$ (instead of 2048). In Table 1 we observe that while the model size is reduced substantially, there is only a limited loss of accuracy. Thus, the final auxiliary model structure does not heavily affect performance, suggesting we can train more generic architectures with this approach.

Comparison to end-to-end VGG We now compare this training method to end-to-end learning directly on a common model, VGG-11. We use $k = 3$ training for all but the final convolutional layer. We additionally reduce the spatial resolution by $2\times$ before applying the auxiliary convolutions. As in the last experiment we use a different final auxiliary network so that the model architecture matches VGG-11. The last convolutional layers' auxiliary model simply becomes the final max-pooling and 2-hidden layer



	Top-5
$\tilde{M}_f = 2048$	88.7
$\tilde{M}_f = 512$	88.5

Table 1. We compare ImageNet SimCNN ($k = 3$) with a much smaller final auxiliary model. The final accuracy is diminished only slightly while the model becomes drastically smaller.

Figure 3. (Left) Linear and (Right) CNN-p separability as a function of depth for CIFAR-10 models. For Linear separability we aggregate across $M = 64, 128, 256$, individual results are shown in Appendix, the relative trends are largely unchanged. For CNN-p probes, all models achieve 100% train accuracy at the first or 2nd layer, thus only test accuracy is reported.

fully connected network of VGG. We train a baseline VGG-11 model using the same 45 epoch training schedule. The performance of the $k = 3$ training strategy matches that of the end-to-end training, with the ensemble model being better. The main differences of this model to SimCNN, besides the final auxiliary, is the max-pooling and the input image starting from full size.

Reference models relying on residual connections and very deep networks have better performance than those considered here. We believe that one can extend layer-wise learning to these modern techniques. However, this is outside the scope of this work. Moreover, recent ImageNet models (after VGG) are developed in industry settings, with large-scale infrastructure available for architecture and hyper-parameter search. Better design of sub-problem optimization geared for this setting may further improve results.

We emphasize that this approach enables the training of larger layer-wise models than end-to-end ones on the same hardware. This suggests applications in fields with large models (e.g. 3-D vision and medical imaging). We also observed that using outputs of early layers that were not yet converged still permitted improvement in subsequent layers. This suggests that our work might allow an extension that solves the auxiliary problems in parallel to a certain degree.

Layerwise Model Compression Wide, over-parametrized, layers have been shown to be important for learning (Neyshabur et al., 2018), but it is often possible to reduce the layer size *a posteriori* without losing significant accuracy (Hinton et al., 2014; LeCun et al., 1990). For the specific case of CNNs, one technique removes channels heuristically and then fine-tunes (Molchanov et al., 2016). In our setting, a natural strategy presents itself, which integrates compression into the learning process: **(a)** train a new layer (via an auxiliary problem) and **(b)** immediately apply model compression to the new layer. The model-compression-related fine-tuning operates over a single layer, making it fast and the subsequent training steps have a smaller input and thus fewer parameters, which speeds up the sequential training. We implement this approach using the filter removal technique of Molchanov et al. (2016) only at each newly trained layer, followed by a fine-tuning of the auxiliary network. We test this idea on CIFAR-10. A baseline network of 5 layers of size 64 (no downsampling, trained for 120 epochs and lr drops each 25 epochs) obtains an end-to-end performance of 87.5%. We use our layer-wise learning with $k = 3, J = 3, M = 128, \tilde{M} = 128$. At each step we prune each layer from 128 to 64 filters and subsequently fine-tune *the auxiliary network* to the remaining features over 20 epochs. We then use a final auxiliary of $\tilde{M}_f = 64$ obtaining a sequentially learned, final network of the same architecture as the baseline. The final accuracy is 87.6%, which is very close to the baseline. We note that each auxiliary problem incurs minimal reduction in accuracy through feature reduction.

5. Conclusion

We have shown that an alternative to end-to-end learning of CNNs relying on simpler sub-problems and no feedback between layers can scale to large-scale benchmarks such as ImageNet and can be competitive with standard CNN baselines. We build these competitive models by training only shallow CNNs and using standard architectural elements (ReLU, convolution). Layer-wise training opens the door to applications such as larger models under memory constraints, model prototyping, joint model compression and training, parallelized training, and more stable training for challenging scenarios. Importantly, our results suggest a number of open questions regarding the mechanisms that underlie the success of CNNs and provide a major simplification for theoretical research aiming at analyzing high performance deep learning models. Future work can study whether the 1-hidden layer cascades objective can be better specified to more closely mimic the $k > 1$ objective.

Acknowledgements

We would like to thank Kyle Kastner, John Zarka, Louis Thiry, Georgios Exarchakis, Maxim Berman, Amal Rannen, Bogdan Cirstea for helpful discussions. EB is partially funded by a Google Focused Research Awards. EO was supported by a GPU donation from NVIDIA.

References

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. CoRR, abs/1811.04918, 2018. URL http://arxiv.org/abs/1811.04918.

Arora, R., Basu, A., Mianjy, P., and Mukherjee, A. Understanding deep neural networks with rectified linear units. International Conference on Learning Representations (ICLR), 2018.

Arora, S., Bhaskara, A., Ge, R., and Ma, T. Provable bounds for learning some deep representations. In International Conference on Machine Learning, pp. 584–592, 2014.

Bach, F. Breaking the curse of dimensionality with convex neural networks. arXiv preprint arXiv:1412.8690, 2014.

Barron, A. R. Approximation and estimation bounds for artificial neural networks. Machine Learning, 14(1):115–133, 1994.

Bartunov, S., Santoro, A., Richards, B. A., Hinton, G. E., and Lillicrap, T. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. arXiv preprint arXiv:1807.04587, 2018.

Bengio, Y., Roux, N. L., Vincent, P., Delalleau, O., and Marcotte, P. Convex neural networks. In Advances in neural information processing systems, pp. 123–130, 2006.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In Advances in neural information processing systems, pp. 153–160, 2007.

Brock, A., Lim, T., Ritchie, J., and Weston, N. Freezeout: Accelerate training by progressively freezing layers. arXiv preprint arXiv:1706.04983, 2017.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on, pp. 4960–4964. IEEE, 2016.

Cortes, C., Gonzalvo, X., Kuznetsov, V., Mohri, M., and Yang, S. Adanet: Adaptive structural learning of artificial neural networks. arXiv preprint arXiv:1607.01097, 2016.

Cybenko, G. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS), 2(4):303–314, 1989.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. International Conference on Learning Representations (ICLR), 2017.

Du, S. S. and Goel, S. Improved learning of one-hidden-layer convolutional neural networks with overlaps. CoRR, abs/1805.07798, 2018. URL http://arxiv.org/abs/1805.07798.

Elad, A., Haviv, D., Blau, Y., and Michaeli, T. The effectiveness of layer-by-layer training using the information bottleneck principle, 2019. URL https://openreview.net/forum?id=r1Nb5i05tX.

Fahlman, S. E. and Lebiere, C. The cascade-correlation learning architecture. In Touretzky, D. S. (ed.), Advances in Neural Information Processing Systems 2, pp. 524–532. Morgan-Kaufmann, 1990a.

Fahlman, S. E. and Lebiere, C. The cascade-correlation learning architecture. In Advances in neural information processing systems, pp. 524–532, 1990b.

Ge, R., Lee, J. D., and Ma, T. Learning one-hidden-layer neural networks with landscape design. arXiv preprint arXiv:1711.00501, 2017.

Greff, K., Srivastava, R. K., and Schmidhuber, J. Highway and residual networks learn unrolled iterative estimation. arXiv preprint arXiv:1612.07771, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

Hinton, G., Vinyals, O., and Dean, J. Dark knowledge. Presented as the keynote in BayLearn, 2, 2014.

Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. Neural computation, 18 (7):1527–1554, 2006.

Huang, F., Ash, J., Langford, J., and Schapire, R. Learning deep resnet blocks sequentially using boosting theory. arXiv preprint arXiv:1706.04964, 2017.

Ivakhnenko, A. G. and Lapa, V. Cybernetic predicting devices. CCM Information Corporation, 1965.

Jacobsen, J.-H., Smeulders, A., and Oyallon, E. i-revnet: Deep invertible networks. In ICLR 2018-International Conference on Learning Representations, 2018.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. arXiv preprint arXiv:1608.05343, 2016.

Janzamin, M., Sedghi, H., and Anandkumar, A. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. arXiv preprint arXiv:1506.08473, 2015.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2017.

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

Kulkarni, M. and Karande, S. Layer-wise training of deep networks using kernel similarity. arXiv preprint arXiv:1703.07115, 2017.

LeCun, Y., Denker, J. S., and Solla, S. A. Optimal brain damage. In Advances in neural information processing systems, pp. 598–605, 1990.

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. Deeply-supervised nets. In Artificial Intelligence and Statistics, pp. 562–570, 2015.

Lee, H., Ge, R., Ma, T., Risteski, A., and Arora, S. On the ability of neural nets to express distributions. arXiv preprint arXiv:1702.07028, 2017.

Lengellé, R. and Denoeux, T. Training mlps layer by layer using an objective function for internal representations. Neural Networks, 9(1):83–97, 1996.

Lillicrap, T. P., Cownden, D., Tweed, D. B., and Akerman, C. J. Random synaptic feedback weights support error backpropagation for deep learning. Nature communications, 7:13276, 2016.

Malach, E. and Shalev-Shwartz, S. A provably correct algorithm for deep learning that actually works. arXiv preprint arXiv:1803.09522, 2018.

Mallat, S. Understanding deep convolutional networks. Phil. Trans. R. Soc. A, 374(2065):20150203, 2016.

Marquez, E. S., Hare, J. S., and Niranjan, M. Deep cascade learning. IEEE Transactions on Neural Networks and Learning Systems, 29(11):5475–5485, Nov 2018. ISSN 2162-237X.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440, 2016.

Mosca, A. and Magoulas, G. D. Deep incremental boosting. arXiv preprint arXiv:1708.03704, 2017.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. Towards understanding the role of over-parametrization in generalization of neural networks. arXiv preprint arXiv:1805.12076, 2018.

Oyallon, E. Building a regular decision boundary with deep networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pp. 1886–1894, 2017.

Oyallon, E. and Mallat, S. Deep roto-translation scattering for object classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2865–2873, 2015.

Oyallon, E., Zagoruyko, S., Huang, G., Komodakis, N., Lacoste-Julien, S., Blaschko, M. B., and Belilovsky, E. Scattering networks for hybrid representation learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 1–1, 2018. ISSN 0162-8828. doi: 10.1109/TPAMI.2018.2855738.

Pinkus, A. Approximation theory of the mlp model in neural networks. Acta numerica, 8:143–195, 1999.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864, 2017.

Sánchez, J., Perronnin, F., Mensink, T., and Verbeek, J. Image classification with the fisher vector: Theory and practice. International journal of computer vision, 105 (3):222–245, 2013.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. Nature, 550(7676):354, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In Advances in Neural Information Processing Systems, pp. 5998–6008, 2017.

Venturi, L., Bandeira, A., and Bruna, J. Neural networks with finite intrinsic dimension have no spurious valleys. arXiv preprint arXiv:1802.06384, 2018.

Wang, G., Xie, X., Lai, J., and Zhuo, J. Deep growing learning. In Proceedings of the IEEE International Conference on Computer Vision, pp. 2812–2820, 2017.

Xiao, W., Chen, H., Liao, Q., and Poggio, T. A. Biologically-plausible learning algorithms can scale to large datasets. International Conference on Learning Representations, 2019.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In Advances in neural information processing systems, pp. 3320–3328, 2014.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579, 2015.

Zagoruyko, S. and Komodakis, N. Wide residual networks. arXiv preprint arXiv:1605.07146, 2016.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In European conference on computer vision, pp. 818–833. Springer, 2014.

A. Proof of Proposition

Proposition 3.1 (Progressive improvement)**.** *Assume that $P_j = Id$. Then there exists θ_0 such that:*

$$\hat{\mathcal{R}}(z_{j+1}; \theta_j^*, \gamma_j^*) \leq \hat{\mathcal{R}}(z_{j+1}; \theta_0, \gamma_{j-1}^*) = \hat{\mathcal{R}}(z_j; \theta_{j-1}^*, \gamma_{j-1}^*) \, .$$

Proof. As $\rho(\rho(x)) = \rho(x)$, we simply have to chose θ_0 such that $W_{\theta_0} = Id$. □

We will now show that given an optimization ϵ-optimal procedure for the sub-problem optimization, the optimization of Algo 1. can be used directly to obtain an error on the overall. solution. Denote the parameters $\{\theta_1^*, ..., \theta_J^*\}$ the optimal solutions for Algo 1.

Proposition 3.2. *Assume the parameters $\{\theta_0^*, ..., \theta_{J-1}^*\}$ are obtained via a optimal layerwise optimization procedure. We assume that $W_{\theta_j^*}$ is 1-lipschitz without loss of generality and that the biases are bounded uniformly by B. Given an input function $g(x)$, we consider functions of the type $z_g(x) = C_\gamma \rho W_\theta g(x)$. For $\epsilon > 0$, we call $\theta_{\epsilon,g}$ the parameter provided by a procedure to minimize $\hat{\mathcal{R}}(z_g; \theta; \gamma)$ which leads to a 1-lipschitz operator that satisfies:*

1. $\underbrace{\|\rho W_{\theta_{\epsilon,g}} g(x) - \rho W_{\theta_{\epsilon,\tilde{g}}} \tilde{g}(x)\| \leq \|g(x) - \tilde{g}(x)\|}_{\textit{(stability)}}, \forall g, \tilde{g},$

2. $\underbrace{\|W_{\theta_j^*} x_j^* - W_{\theta_{\epsilon,x_j^*}} x_j^*\| \leq \epsilon(1 + \|x_j^*\|)}_{\textit{(ϵ-approximation)}},$

with, $\hat{x}_{j+1} = \rho W_{\theta_{\epsilon,\hat{x}_j}} \hat{x}_j$ and $x_{j+1}^* = \rho W_{\theta_j^*} x_j^*$ with $x_0^* = \hat{x}_0 = x$, then, we prove by induction:

$$\|x_J^* - \hat{x}_J\| = \mathcal{O}(J^2 \epsilon) \tag{5}$$

Proof. First observe that $\|x_{j+1}^*\| \leq \|x_j^*\| + B$ by non expansivity. Thus, by induction, $\|x_j^*\| \leq jB + \|x\|$. Then, let us show that: $\|x_j^* - \hat{x}_j\| \leq \epsilon(\frac{j(j-1)}{2} B + j\|x\| + j)$ by induction. Indeed, for $j + 1$:

$$\begin{aligned}
&\|x_{j+1}^* - \hat{x}_{j+1}\| \\
&= \|\rho W_{\theta_j^*} x_j^* - \rho W_{\theta_{\epsilon,\hat{x}_j}} \hat{x}_j\| \\
&= \|\rho W_{\theta_j^*} x_j^* - \rho W_{\theta_{\epsilon,x_j^*}} x_j^* + \rho W_{\theta_{\epsilon,x_j^*}} x_j^* - \rho W_{\theta_{\epsilon,\hat{x}_j}} \hat{x}_j\| \\
&\text{by non-expansivity} \\
&\leq \|W_{\theta_j^*} x_j^* - W_{\theta_{\epsilon,x_j^*}} x_j^*\| + \|\rho W_{\theta_{\epsilon,x_j^*}} x_j^* - \rho W_{\theta_{\epsilon,\hat{x}_j}} \hat{x}_j\| \\
&\leq \epsilon \|x_j^*\| + \epsilon + \|x_j^* - \hat{x}_j\| \text{ from the assumptions} \\
&\leq \epsilon(jB + \|x\| + 1) + \|x_j^* - \hat{x}_j\| \text{ from above} \\
&\text{by induction} \\
&\leq \epsilon(jB + \|x\| + 1) + \epsilon(\frac{j(j-1)}{2} B + j + j\|x\|)\| \\
&= \epsilon(\frac{j(j+1)}{2} B + (j+1)\|x\| + (j+1))
\end{aligned}$$

As $x_0^* = \hat{x}_0 = x$, the property is true for $j = 0$.

□

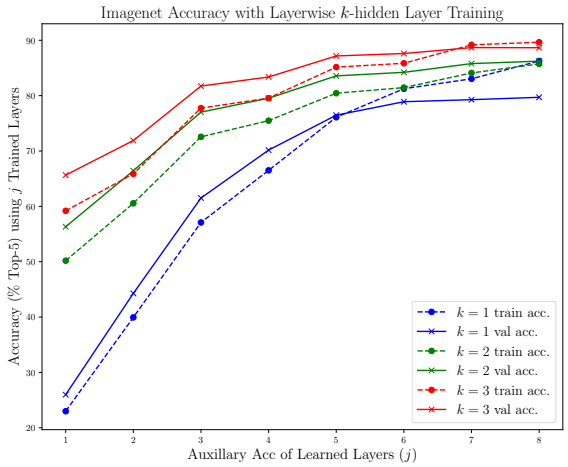


Figure 4. Intermediate Accuracies of models in Sec. 4.3. We note that the $k = 1$ model is larger than the $k = 2, 3$ models.

A Note on Sec.2 From (Mallat, 2016)

We first briefly discuss the result of Sec.2 of (Mallat, 2016). Let us introduce: $\Omega_j = \{x : \forall x', f(x) \neq f(x') \Rightarrow \rho W_{\hat{\theta}_{j-1}} \cdots \rho W_{\hat{\theta}_1} x \neq \rho W_{\hat{\theta}_{j-1}} \cdots \rho W_{\hat{\theta}_1} x'\}$. We introduce: $\mathcal{Y}_j = \{y : \exists x \in \Omega_j, y = \rho W_{\hat{\theta}_{j-1}} \cdots \rho W_{\hat{\theta}_1} x\}$. For $y \in \mathcal{Y}_j$, we define $\hat{f}_j(y) \triangleq f(x)$. Observe this defines indeed a function, and that:

$$\forall x \in \Omega_j, f(x) = \hat{f}_j \circ \rho W_{\hat{\theta}_{j-1}} \cdots \rho W_{\hat{\theta}_1} x$$

Observe also that $\Omega_{j+1} \subset \Omega_j$. The set Ω_j is simply the set of samples which are well discriminated by the neural network $\rho W_{\hat{\theta}_{j-1}} \cdots \rho W_{\hat{\theta}_1}$.

B. Additional Details on Imagenet Models and Performance

For ImageNet we report the improvement in accuracy obtained by adding layers in Figure 4 as seen by the auxiliary problem solutions. We observe that indeed the accuracy of the model on both the training and validation is able to improve from adding layers as discussed in depth in Section 4.2. We observe that $k = 1$ also over-fits substantially, suggesting better regularization can help in this setting.

We provide a more explicit view of the network sizes in Tab. 4 and Tab. 5. We also show the number of parameters in the ImageNet networks in Tab. 7. Although some of the models are not as parameter efficient compared to the related ones in the literature, this was not a primary aim of the investigation in our experiments and thus we did not optimize the models

Layer	spatial size	layer output size
Input	112×112	12
1	112×112	128
2	112×112	128
3	56×56	256
4	56×56	256
5	28×28	512
6	28×28	512
7	14×14	1024
8	14×14	1024

Table 4. Network structure for $k = 2, 3$ imagenet models, not including auxiliary networks. Note an invertible downsampling is applied on the input 224x224x3 image to producie the initial input. The default auxillary networks for both have $\tilde{M}_f = 2048$ with 1 and 2 auxillary layers, respectively. Note auxiliary networks always reduce the spatial resolution to $2x2$ before the final linear layer.

Layer	spatial size	layer output size
Input	112×112	12
1	112×112	256
2	112×112	256
3	56×56	512
4	28×28	1024
5	14×14	2048
6	14×14	2048
7	7×7	4096
8	7×7	4096

Table 5. Network structure for $k = 1$ ImageNet models, not including auxiliary networks. Note an invertible down-sampling is applied on the input 224x224x3 image to produce the initial input. Note this network does not include any batch-norm.

for parameter efficiency (except explicitly at the end of Sec. 4.3), choosing our construction scheme for simplicity. We highlight that this is not a fundamental problem in two ways: (a) for the $k = 1$ model we note that removing the last two layers reduces the size by $1/4$, while the top 5 accuracy at the earlier J=6 layer is 78.8 (versus 79.7), see Figure 4 for detailed accuracies. (b) Our models for $k = 2, 3$ have most of their parameters in the final auxiliary network which is easy to correct for once care is applied to this specific point as at the end of Sec. 4.3. We note also that the model with $k = 3, M_f = 512$, is actually more parameter efficient than those in the VGG family while having similar performance. We also point out that we use for simplicity the VGG style construction involving only $3x3$ convolutions and downsampling operations that only half the spatial resolution, which indeed has been shown to lead to relatively less parameter efficient architectures (He et al., 2016), using less uniform construction (larger filters and bigger pooling early on) can yield more parameter efficient models.

Layer-wise Trained	Acc.
Strided Convolution	87.8
Invertible Down	88.3
AvgPool	87.6
MaxPool	88.0

Table 6. Comparison of different downsampling operations

C. Additional Studies

We report additional studies that elucidate the critical components of the system and demonstrate the transferability properties of the greedily learned features.

C.1. Choice of Downsampling

In our experiments we use primarily the invertible downsampling operator as the downsampling operation. This choice is to reduce architectural elements which may be inherently lossy such as average pooling. Compared to max-pooling operations it also helps to maintain the network in Sec. 4.1 as a pure ReLU network, which may aid in analysis

Models	Number of Parameters
SimCNN $k = 3$, $M_f = 512$	46M
SimCNN $k = 3$	102M
SimCNN $k = 2$	64M
SimCNN $k = 1, J = 6$	96M
AlexNet	60M
VGG-16	138 M

Table 7. Overall parameter counts for SimCNN models trained in Sec. 4 and from literature.

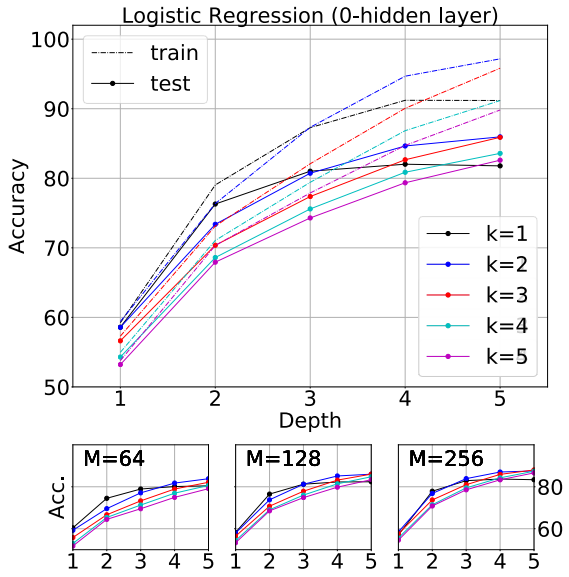


Figure 5. Linear separability of differently trained sequential models. We show how the data varies for the different M, observing similar trends to the aggregated data.

as the maxpooling introduces an additional non-linearity. We show here the effects of using alternative downsampling approaches including: average pooling, maxpooling and strided convolution. On the CIFAR dataset in the setting of $k = 1$ we find that they ultimately lead to very similar results with invertible downsampling being slightly better. This shows the method is rather general. In our experiments we follow the same setting described for CIFAR. The setting here uses $J = 5$ and downsamplings at $j = 2, 3$. The size is always halved in all cases and the downsampling operation and the output sizes of all networks are the same. Specifically the Average Pooling and Max Pooling use 2×2 kernels and the strided convolution simply modifies the 3×3 convolutions in use to have a stride of 2. Results are shown in Tab. 6.

C.2. Effect of Width

We report here an additional view of the aggregated results for linear separability discussed in Sec. 4.2. We observe that the trend of the aggregated diagram is similar when comparing only same sized models, with the primary differences in model sizes being increased accuracy.

C.3. Transfer Learning on Caltech-101

Deep CNNs such as AlexNet trained on Imagenet are well known to have generic properties for computer vision tasks, permitting transfer learning on many downstream applications. We briefly evaluate here if the $k = 1$ imagenet model (Sec. 4.1) shares this generality on the Caltech-101 dataset.

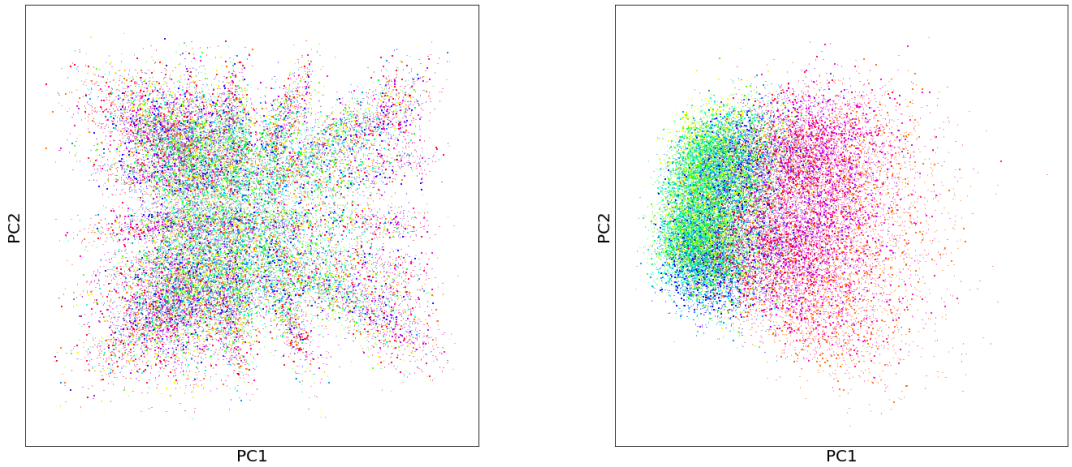
	Accuracy
ConvNet from scratch (Zeiler & Fergus, 2014)	$46 \pm 1.7\%$
Layer 1	45.5 ± 0.9
Layer 2	59.9 ± 0.9
Layer 3	70.0 ± 0.9
Layer 4	75.0 ± 1.0
Layer 8	82.6 ± 0.9

Table 8. Accuracy obtained by a linear model using the features of the $k = 1$ network at a given layer on the Caltech-101 dataset. We also give the reference accuracy without transfer.

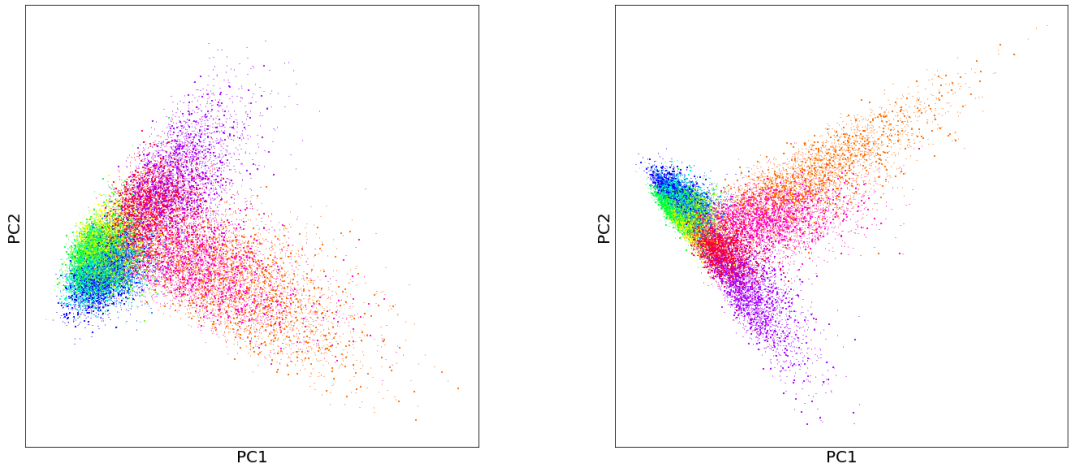
This dataset has 101 classes and we follow the same standard experimental protocol as (Zeiler & Fergus, 2014): 30 images per class are randomly selected, and the rest is used for testing. The average per class accuracy is reported using 10 random splits. As in (Zeiler & Fergus, 2014) we restrict ourselves to a linear model. We use a multinomial logistic regression applied on features from different layers including the final one. For the logistic regression we rely on the default hyperparameter settings for logistic regression of the `sklearn` package using the SAGA algorithm. We apply a linear averaging and PCA transform (for each fold) to reduce the dimensionality to 500 in all cases. We find the results are similar to those reported in (Zeiler & Fergus, 2014) for their version of the AlexNet. This highlights the model has similar transfer properties and also shows similar progressive linear separability properties as end-to-end trained models.

C.4. Experiments with incremental-PCA

For CIFAR-10 and the $k = 1$ model, we visualize the separability of the representations we built via incremental-PCA at various depths. Results are summarized in Figure 6.



(a) (Left) Layer 1, (Right) Layer 2



(b) (Left) Layer 3, (Right) Layer 4

Figure 6. Projected representations of the model $k = 0$ via an incremental PCA, on CIFAR-10. Each color corresponds to a distinct class.