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

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

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 $j+1$:

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

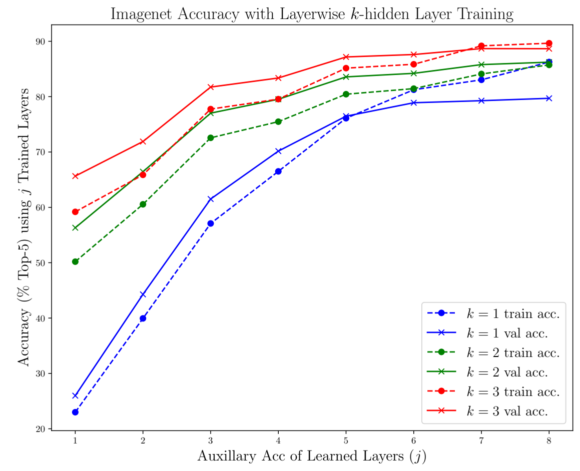


Figure 4. Intermediate Accuracies of models in Sec. 4.3. We note that the $k = 1$ model is larger than the $k = 2, 3$ models.

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

□

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.

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

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

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.

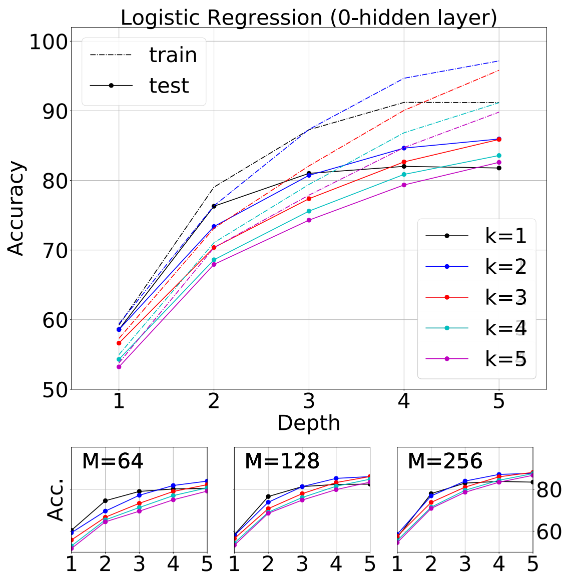


Figure 5. Linear separability of differently trained sequential models. We show how the data varies for the different M, observing similar trends to the aggregated data.

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.

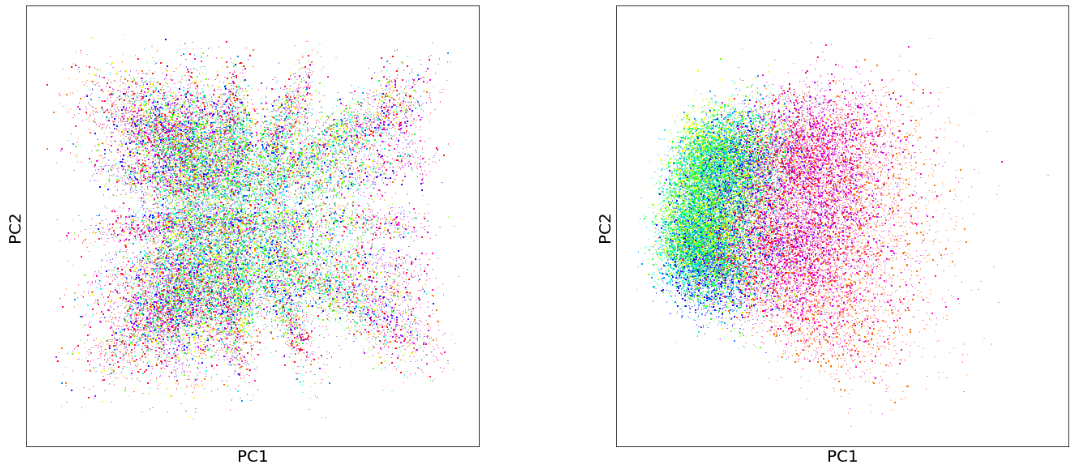
	Accuracy
ConvNet from scratch (Zeiler & Fergus, 2014)	$46 \pm 1.7\%$
Layer 1	45.5 ± 0.9
Layer 2	59.9 ± 0.9
Layer 3	70.0 ± 0.9
Layer 4	75.0 ± 1.0
Layer 8	82.6 ± 0.9

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.

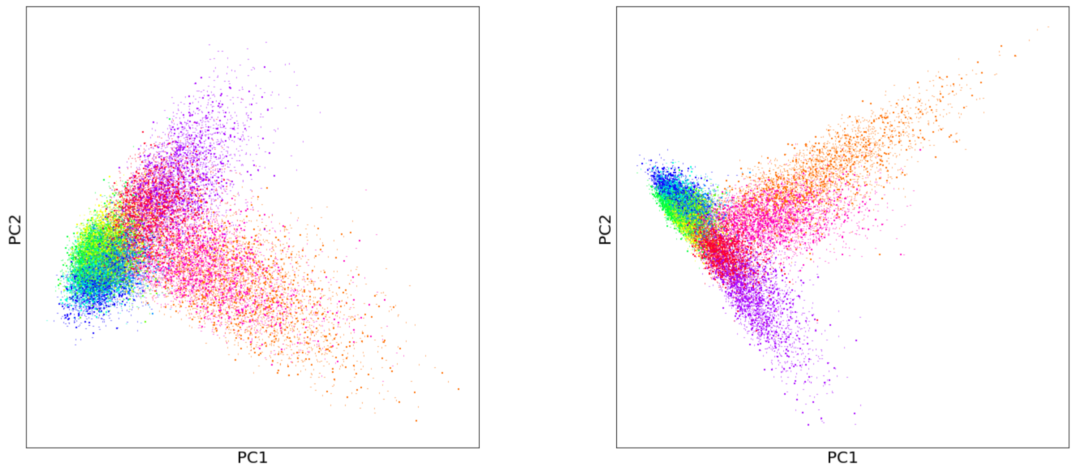
This dataset has 101 classes and we follow the same standard experimental protocol as (Zeiler & Fergus, 2014): 30 images per class are randomly selected, and the rest is used for testing. The average per class accuracy is reported using 10 random splits. As in (Zeiler & Fergus, 2014) we restrict ourselves 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.