Neural Network Pruning Denoises the Features and Makes Local Connectivity Emerge in Visual Tasks

Franco Pellegrini [1] **Giulio Biroli** [1]

Abstract

Pruning methods can considerably reduce the size of artificial neural networks without harming their performance and in some cases they can even uncover sub-networks that, when trained in isolation, match or surpass the test accuracy of their dense counterparts. Here, we characterize the inductive bias that pruning imprints in such "winning lottery tickets": focusing on visual tasks, we analyze the architecture resulting from iterative magnitude pruning of a simple fully connected network. We show that the surviving node connectivity is local in input space, and organized in patterns reminiscent of the ones found in convolutional networks. We investigate the role played by data and tasks in shaping the architecture of the pruned sub-network. We find that pruning performances, and the ability to sift out the noise and make local features emerge improve by increasing the size of the training set, and the semantic value of the data. We also study different pruning procedures, and find that iterative magnitude pruning is particularly effective in distilling meaningful connectivity out of features present in the original task. Our results suggest the possibility to automatically discover new and efficient architectural inductive biases in other datasets and tasks.

1. Introduction

In the last decade deep and over-parametrized neural networks achieved breakthroughs in a variety of contexts (Krizhevsky et al., 2012; He et al., 2016; Vaswani et al., 2017; Brown et al., 2020; Silver et al., 2017). To keep pushing the boundaries of their capacity, these networks have grown in size to over billions of parameters (Tan & Le, 2020). Network pruning approaches have recently received renewed attention as effective procedures to counterbalance this growth by reducing networks size without harming their performance. Several pruning methods have been introduced (Hoefler et al., 2021): they are mostly based on the idea of cutting weights from a trained, or partially trained, network in order to obtain a sparse sub-network with similar accuracy to the original one. For applications, pruning is very efficient in improving storage and computational cost; on the theoretical side, it led to the discovery that within a randomly-initialized network one can find sub-networks, *winning lottery tickets*, that even when trained in isolation can match the test accuracy of the original network (Frankle & Carbin, 2019).

Advanced network architectures, with geometries suited to specific tasks, resulting in faster training and overall better generalizing properties, can also be seen from a similar perspective. For instance, Convolutional Neural Networks (Krizhevsky et al., 2012; He et al., 2016; LeCun et al., 1989; Simonyan & Zisserman, 2015; Szegedy et al., 2015) (CNN), which are very efficient for visual tasks, can be embedded in the Fully Connected Network (FCN) class. CNNs can therefore be interpreted as sparse FCNs trained from very specific initial conditions well adapted to the task (d'Ascoli et al., 2019). Also in this case the size is drastically reduced (with respect to the FCN counterpart). The difference with pruning is that the main characteristics of CNNs—local connectivity and weight sharing—do not come from an automatic procedure but are "hand-designed" features introduced starting from analogies with the visual cortex (Hubel & Wiesel, 1962; Fukushima, 1980), studies of the invariant properties of natural images (Ruderman, 1994), and engineering improvements (LeCun et al., 1989). Pruning instead proceeds in an agnostic way in order to find smaller and efficient architectures; it takes advantage of properties which are imprinted in the network through training and which result from the interplay of data, task and optimization algorithm. Pruning induces an *inductive bias* customized to the learning task at hand, which leads to a network with better training and generalization properties compared to one of similar size trained from scratch. Surprisingly, it has been shown that the same winning lot-

[1]Laboratoire de Physique de l'École normale supérieure, ENS, Université PSL, CNRS, Sorbonne Université, Université de Paris — F-75005 Paris, France. Correspondence to: Franco Pellegrini <franco.pellegrini@phys.ens.fr>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

tery ticket generalizes across training conditions and similar datasets (Morcos et al., 2019). This implies that the bias induced by pruning is sufficiently generic to transfer the pruned networks within the same data domain.

Here, we characterize the nature of such inductive bias in the case of dense networks trained to classify natural images, and reveal that the winning lottery tickets of FCNs display the key features of CNNs. We apply *iterative magnitude pruning* (IMP) (Frankle & Carbin, 2019) on FCNs trained on a low resolution version of ImageNet (Chrabaszcz et al., 2017). We show that the sub-network obtained by IMP is characterized by *local connectivity*, especially in the first hidden layer, and masks leading to *local features* with patterns very reminiscent of the ones of trained CNNs (Zeiler & Fergus, 2014). We study how data and task affect these pruning abilities. We show a crossover from a large dataset regime (large signal-to-noise ratio), where the inductive bias is present, to a small dataset regime in which pruning loses performance and concomitantly the properties described above disappear. A similar crossover takes place when going from a meaningful task to one with low semantic value. We also compare different pruning procedures: IMP seems to be particularly effective in producing well defined local structures, as opposed to more direct pruning procedures.

2. Related works

Several recent works have studied pruning from the perspective of the lottery ticket hypothesis (Frankle & Carbin, 2019): some investigated more general weight reinitializations (Zhou et al., 2019), whereas others actually questioned its validity on larger nets (Liu et al., 2019; Gale et al., 2019). Later, Frankle et al. (2019; 2020a) showed that in order to obtain good results for complex networks, datasets, and tasks it is important to modify the rewinding time compared to the initial formulation of the hypothesis. Other studies concerned hyperparameters modifications (Frankle et al., 2020b; Renda et al., 2020), and concentrated on the transferability (Morcos et al., 2019; Sabatelli et al., 2020) of the pruned networks. To the best of our knowledge, our work is the first to analyze the internal structure of winning tickets in order to characterize the inductive bias induced by pruning and relate it to architectural properties.

The possibility of learning CNN-like inductive bias from data and through training was investigated in Neyshabur (2020). It was shown that training using a modified ℓ_1 regularization is a way to induce local masks for visual tasks. This is similar to our results on pruning: enforcing sparsity during training leads to structures characterized by locality. d'Ascoli et al. (2019) studies the role of CNN-like inductive biases by embedding convolutional architectures within the general FCN class. It shows that enforcing CNN-like features in an FCN can improve performance even beyond that of its CNN counterpart. Finally, Tolstikhin et al. (2021) shows that by considering a particular multilayer perceptron architecture, called MLP-mixer, some of the CNN features can be learned from scratch using a large training dataset.

3. Method, observables and notation

Data, networks, and training procedure Throughout this work our reference dataset, referred to as ImageNet32, consists of the almost 1.3M images classified in 1000 categories of the ILSVRC-12 image classification challenge (Russakovsky et al., 2015), cropped and scaled down to 32×32 resolution as detailed in (Chrabaszcz et al., 2017) (some experiments at higher resolution are presented in SM C.5). Considering the 3 color channels (RGB) the input size is $n_0 = 3072$ (we index the input as layer 0). The validation test contains 10k images. We do not use any form of augmentation in order to avoid biases towards translationally invariant features (the choice of a large dataset is important in this sense to obtain good statistics, as shown in Sec. 5.2). In an effort to keep a lightweight setup that can be easily trained and analyzed, in most of this work we consider a FCN having 3 hidden layers of equal size $n_1 = n_2 = n_3 = 1024$ (similar results for slightly larger or deeper FCN architectures are presented in SM A.2, and an 8 layer network is considered in SM B.6 to explicitly obtain a more hierarchical structure). We apply batch normalization (Ioffe & Szegedy, 2015) and we use rectified linear unit (ReLU) nonlinearities for each layer. We denote w_{ij}^{l+1} the weights between node i of layer l and node j of layer $l+1$. Weights are initialized from a normal distribution $w^{l+1} \sim \mathcal{N}(0, 2/(n_l + n_{l+1}))$ (Glorot & Bengio, 2010). Each node also has a bias b_i^l, initialized at 0. The size of the output layer equals the number of categories ($n_4 = 1000$ in this first experiment, 10 in Section 5.3), which are converted to probabilities to compute the cross-entropy loss. The reported accuracy is the ratio of correct, most probable, labels. We train on mini-batches of 1000 images, and minimize through stochastic gradient descent with learning rate $\alpha = 0.1$. Each training run corresponds to 10^5 steps (around 78 epochs): while this is not sufficient to fit the training set, it is effective in identifying a winning lottery ticket (You et al., 2020). We checked that our results are robust while changing the optimization procedure, see the SM A.1.

Pruning We define the *pruning* of a weight as multiplying it by a zero mask, effectively removing it from the network. The main procedure considered in this work is layer-wise IMP, i.e. the pruning of a certain percentage of the weights of a layer, chosen as the smallest ones in absolute value at the end of training. The procedure is iterated by rewinding the weights to their initial values and retraining the now

pruned network, using the new final magnitudes to select a further percentage of weights to be pruned, as originally described in Frankle & Carbin (2019). In most of this work, we remove $p = 30\%$ of previously unpruned weights at each iteration from all layers except for the output (pruning the last, much smaller, layer at the same rate hinders performance, we therefore opt not to prune it at all). When analyzing the pruned networks, we denote u the ratio of unpruned weights per layer: after n iterations $u = (1-p)^n$. As suggested in Frankle et al. (2019), we rewind the weights after each training not to the initial values, but to the ones obtained after 1000 steps of optimization in the first run. Instead of a winning lottery ticket the resulting network has been called a matching ticket (Frankle et al., 2019) (we have verified that the main result holds when rewinding to 0, see SM A.3). We iterate the training and pruning until less than 20% of the nodes in any layer retain any connection to the previous layer, i.e. until more than 80% of the nodes have been completely pruned away. More experiments modifying pruning parameters, including total training time and the pruning ratio p, and performing global pruning, are presented in SM A.3. In Sec. 5.1 we compare IMP to a one-shot pruning at high sparsity by performing a single pruning with larger p.

We also compare our results to SNIP (Lee et al., 2019), an algorithm for one-shot pruning at initialization. This procedure defines the saliency criterion for a weight as the absolute value of the derivative of the loss with respect to its mask. The saliency is computed on a single batch at initialization and the chosen percentage of weights with smallest saliency (network wide) are pruned in one shot. In this work we estimate saliency over a batch of 10^4 images (we have verified that increasing the number of images does not qualitatively modify the masks).

Lastly, we compare our results to SynFlow (Tanaka et al., 2020), a data free pruning scheme. In this case, an alternative loss function is used to estimate information flow through the network, constructed by multiplying the absolute value of all the network weight matrices (at initialization) with ones in input and output. The derivative of this with respect to the weights, multiplied by the weights themselves, represents the synaptic flow scores used to establish which connections to prune. Like IMP, the desired sparsity is reached iteratively by pruning a growing percentage of the weights and re-evaluating the synaptic flow. In Sec. 5.1 we reach the desired 90% sparsity in 100 iterations (following the exponential schedule suggested by the authors to reach compression ratio $\rho = 10$ (Tanaka et al., 2020), but we verified that the structure of the masks is stable with respect to these parameters).

Observables In order to analyze the structure of the subnetwork induced by pruning we focus on the following observables:

- *Masks*: for each layer, we define masks by assigning 0 to pruned weights, and 1 to all the others. We indicate with m^l_{ij} the mask formed by the unpruned weights w^l_{ij}.
- *Node connectivity*: for each node j in layer l we define the input connectivity $C^{\text{in},l}_j = \sum_i m^l_{ij}$ as the number of input unpruned nodes, and $C^{\text{out},l}_j = \sum_i m^{l+1}_{ji}$ the output connectivity.
- *Local distances*: in order to asses the locality of the masks, we count the number of pixels in a given relative position, $\mathbf{d}$, that are connected to the same hidden node, as shown schematically in Fig. 2**a**. More precisely: two inputs i and i' are connected through node j if $m^1_{ij}m^1_{i'j} = 1$. For such inputs we define the displacement $\mathbf{d}_{ii'}$: if input index i identifies a pixel of spatial position (x, y) and i' corresponds to position (x', y'), we define their displacement $\mathbf{d}_{ii'} \equiv (x' - x, y' - y)$. The histogram of connected pixels at a given displacement is denoted $S(\mathbf{d}) = \sum_j \sum_{i,i'|\mathbf{d}_{ii'}=\mathbf{d}} m^1_{ij}m^1_{i'j}$. We consider both the sum over pixels belonging to the same color channel (S^{sc}) and over different color channels (S^{dc}).

4. Pruning sifts out local features from the noise and recovers CNN-like masks

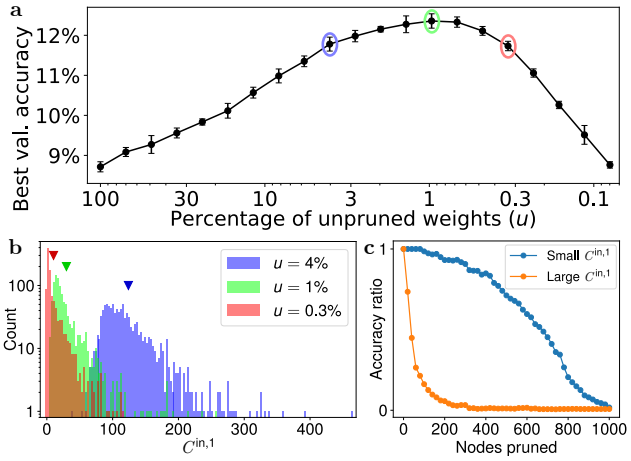

Figure 1. **a:** Highest validation accuracy at different percentage of unpruned weights. Averages and standard deviations over 4 independent IMP runs. **b:** Histograms of the number of $C^{\text{in},1}$ for three configurations at different sparsities. The triangles represent the average connections $n_0 u$ at each sparsity. **c:** Accuracy with respect to the full network for networks with a varying number of nodes removed, in increasing or decreasing order of $C^{\text{in},1}_j$.

Fig. 1**a** shows the highest (early stop) validation accuracy as a function of u throughout the IMP procedure. As ex-

pected (Blalock et al., 2020), pruning allows to find sub-networks of considerably smaller sizes that work as well as the original ones. Actually, as shown in Fig. 1**a**, IMP leads to a strong improvement in the accuracy which goes from less than 9% to more than 12% when 99% of the weights are pruned[1].

The pruning-induced masks are local To characterize these pruned network, we concentrate on 3 sparsity values close to the maximum accuracy reached by IMP: $u = 4\%$, 1%, and 0.3%, as circled in Fig. 1**a**. We start by showing the histograms of $C^{\text{in},1}$, i.e. the input connectivity per node of the first hidden layer. For a random pruning, this would be binomially distributed around the value $n_0 u$, i.e. 123, 31, and 10 respectively for the sparsities considered. Fig. 1**b** shows that the distributions obtained after pruning instead have long positive tails, representing a population of nodes that preserve a large number of unpruned input weights. Since IMP should preserve "important" connections, it is interesting to check whether these nodes represent meaningful *features* that have been sifted out by the procedure. To this end, we consider the relative accuracy of the final network when either the nodes with the largest or with the smallest $C^{\text{in},1}$ have been completely removed (i.e. we set $m^1_{ik} = 0$ for those nodes k, without any further training). As shown in Fig. 1**c**, the accuracy as a function of number of removed nodes drops much more quickly when removing the highly connected nodes, thus confirming their relevance.

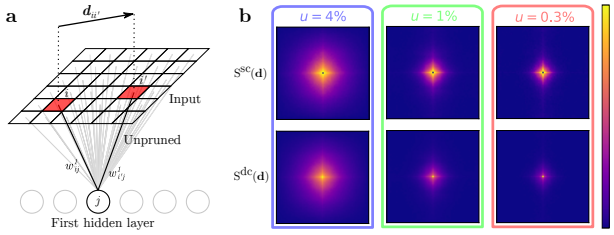

Figure 2. **a:** Sketch of the displacement and connectivity considered when computing $S(\mathbf{d})$. **b:** S^{sc} (top) and S^{dc} (bottom) for each sparsity considered (columns). Relative positions are shown with respect to the center of each image ($\mathbf{d} = 0$) and the color scale of each plot goes from 0 to its maximum. For S^{sc}, the single point at $\mathbf{d} = 0$ is removed as it is trivially always connected.

In order to investigate the emerging local structure of the unpruned weights, we consider the number of connected pixels in a given relative position $S(\mathbf{d})$. A 2D map for any possible displacement $\mathbf{d}$ is shown in Fig. 2**b**, for the 3 sparsities considered and for either same color or different color channels. Remarkably, we find that inputs connected to the same node are locally close, i.e. their distance is small, both within and between color channels. Note that the same 2D maps for the unpruned layer or pruned layers with random connections, are very different and non-local as shown in the SM B.1. We also find that straight horizontal and vertical relative displacements are slightly more likely. As the pruning progresses, the localization becomes stronger and the anisotropy more apparent. The same 2D maps are shown by only selecting nodes with a given number of connections in SM B.3. Our results show that the locality of the masks holds for all nodes except the ones with the smallest values of $C^{\text{in},1}$. For the larger masks the effect is truly remarkable given their large number of unpruned weights. The set of results discussed above unveils one key feature of the matching (and winning) lottery ticket of FCNs: they display *local masks* that emerge by pruning.

The local features are structured We now concentrate on the nodes with the largest $C^{\text{in},1}_j$, which have been found to be most relevant, and focus on the spatial structure of the associated masks. Fig. 3**a** shows the masks of the 30 most connected nodes for the best IMP iteration ($u \simeq 1\%$): they all present well localized distributions, either in a single location, symmetric with respect to the vertical axis, or sometimes around the edges of the images. Several masks also retain the same pixels through multiple color channels, often with a well defined pattern (e.g. 2 channels out of 3). Moreover, many of the masks show a specific preference for vertical and horizontal directions, in line with the previously observed anisotropy, with some consisting of parallel lines separated by one or two pixels (the effect is even more pronounced for stronger pruning, see SM B.4, where we also show a larger sample of masks). A video of the evolution of first layer masks is provided as Supplementary Material. Masks with a lower number of connections do not show such well defined structures. In Fig. 3**b** we select a few nodes within the top 5% most connected ones to highlight their specific structure: vertical lines, horizontal lines, oblique lines and complementary patches of color. To clarify the features selected by these nodes, we show not only the masks, but also the final (masked) weights and the 3 images in the validation with the highest activations over these nodes. The weights highlight further structure in the localized patches, with alternating positive and negative weights perpendicular to the preferential direction for long masks, or between different patches. These are very reminiscent of Gabor filters (Marĉelja, 1980), and very similar to CNN masks, which respond preferentially to local high contrast details, e.g. edges in a specific direction or color patches (Zeiler & Fergus, 2014). Moreover, very similar

[1]The overall accuracy is low compared to the state of the art. This is due to the fact that we are focusing on a simple (hence fully analyzable) setup and by choice we avoid improvements that could lead to biases in the results. Yet, considering the low resolution, lack of augmentation, and simplicity of the network for a notoriously complex dataset, the improvement over the random 10^{-3} is a clear signal of learning. The top-5 accuracy is around 26%.

filters can be seen at different locations, showing the emergence of partial translational invariance in the local features even without data augmentation[2]. Lastly, the example images unveil the response of these nodes to local features, not directly related to the final classification task. This directly relates to the good transfer properties of the subnetworks corresponding to the winning tickets (Morcos et al., 2019; Sabatelli et al., 2020). Further experiments highlighting the structure of the masks are presented in SM C.4, where the images are rotated, in SM C.5, where higher resolution images are used as input, and in SM C.6, where we restore translational invariance by considering all possible image translations.

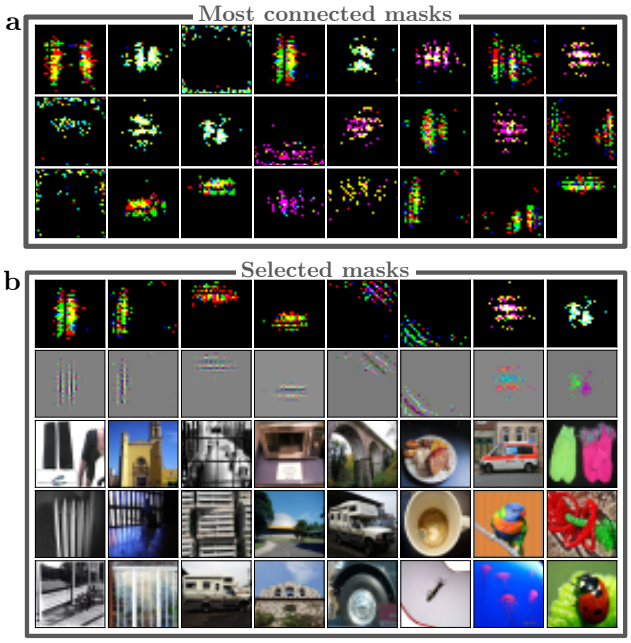

Figure 3. **a:** Masks of the 30 nodes with largest $C_j^{\text{in},1}$ for the best IMP iteration. Binary masks are directly converted to RGB intensities. **b:** For a hand-picked selection of the most connected nodes (columns), the top row reports the mask and the second row represent the masked weights $w_{ij}^1 m_{ij}^1$ normalized from minimum to maximum and converted to RGB. The last 3 rows represent the images with highest activation for that specific node.

Towards a hierarchical structure To continue the parallel with CNNs, we now focus on the deeper layers. For a second layer node j, we can define an effective mask as $\mu_{ij} = \theta(\sum_k m_{ik}^1 m_{kj}^2 - 1/2)$ (θ being the step function), which equals 1 only if pixel i is connected to *any* first hidden layer node connected to j, and 0 otherwise. Ideally the receptive field should still be localized and grow progressively with each layer, but since the network is just 3 layers deep, and the receptive fields are already fairly large in the first layer, this effect is hard to visualize. However, even for this very small network, we find the connectivity of the second layer masks to be still fairly local, especially at high pruning. These masks are presented in SM B.4, while in SM B.6 we present some specific examples of second layer nodes highlighting how first layer masks similar in shape and position are combined. In order to obtain a more clear hierarchical structure, while keeping the network size contained, in SM B.6 we consider a deeper network of 8 layers (all of size 1024) and prune them one at a time starting from the input, to increase the probability of finding meaningful connections within a relatively narrow network. At high pruning, this procedure indeed shows a progressive combination of local masks with more slowly growing receptive fields.

[2]Note that images in the training set have a bias for structure in the center and at the boundary, a characteristic that emerges also from pruning: features tend to have more connections towards the center and edges of the image, as shown in SM B.2.

5. Pruning, data and task

In order to clarify the relative contribution of different parts of our training and pruning scheme, in this section we first showcase the main effects of changing the pruning procedure. We then focus on the role of the dataset the network is trained on, or simply the nature of the classification task. We will see that, while the masks clearly highlight some structures extracted from the data by the training procedure, it is only on a well defined task and through a progressive pruning that we can obtain the results showcased in the previous sections. Failing any of these elements, the masks will gradually lose their structure, until they become indistinguishable from random pruning. This reveals that pruning acts as a denoising method to make local features emerge; in order to do that it needs a large enough signal to noise ratio (enough data, meaningful enough task).

5.1. Other pruning approaches

Fig. 4 shows the weights at the end of dense training (first row) for a few selected nodes in the first layer, compared to the masks at different levels of IMP. This comparison clearly shows how the features are already suggested by the first round of training and progressively refined through the IMP. However, it is not clear whether they could be more efficiently recovered. To investigate this possibility we consider the same initial network, and perform a single "one-shot" pruning operation (Frankle & Carbin, 2019) to the desired sparsity. Not only, as expected from IMP theory, is the iterative pruning more effective, but the masks obtained in one shot are in general more noisy, less localized and less structured. We can thus suggest that the effect of iterative pruning is that of de-noising or stabilizing the masks, which are only suggested by the first training round.

In order to compare to other one-shot pruning approaches,

we also perform SNIP (Lee et al., 2019) pruning at initialization on the same network. While the pruning is quite effective in preserving the performances of the network, and the masks still show a certain degree of locality, they do not bear much resemblance to the final weights (unsurprisingly, since they are obtained at initialization, before the training dynamics has had the possibility to select them). More graphs showing the locality and connectivity of the first layer are presented in SM B.5.

Lastly, we perform SynFlow (Tanaka et al., 2020) data-free pruning. We iteratively prune a percentage of nodes with the smallest saliency as defined in (Tanaka et al., 2020), reaching 90% sparsity. In this case, we can clearly do not expect any correlation with the data, since they are not employed in pruning. With no information on the data, the masks cannot show any locality or structure, and to maximize synaptic flow few highly-connected nodes are selected (see SM B.5).

All in all, this analysis with different pruning procedures confirms our previous findings and shows that IMP is very effective.

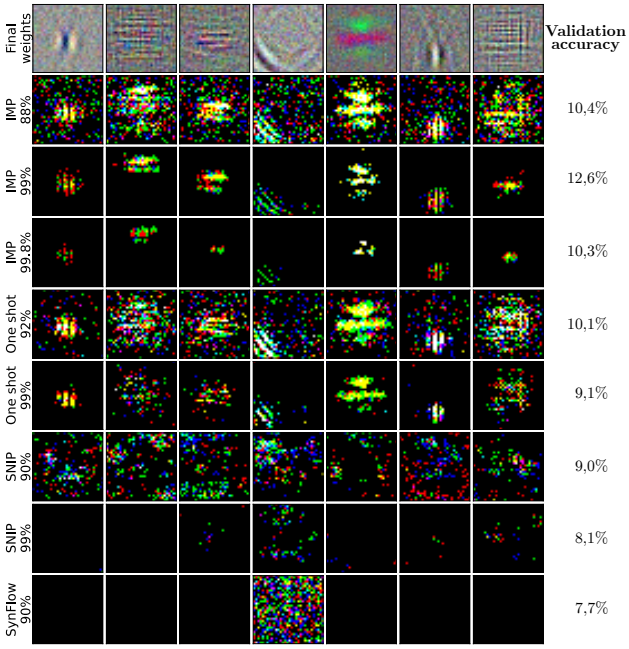

Figure 4. Weights and masks for a selection of nodes under different pruning. Each column represents a specific first layer node of networks trained from the same initial conditions; the last column reports the best validation accuracy achieved with each pruning scheme at the specified sparsity. The first row represent the final weights after dense training, the following rows show the masks at different IMP steps of for one-shot magnitude pruning. The last three rows represent the masks found at initialization for the same nodes through SNIP and SynFlow.

5.2. The importance of data

We now study to what extent the effects found in the previous section depend on the properties of the dataset, in particular the *number of data* used to train the network. We repeat the training and the IMP with datasets of progressively smaller size, down to 2% of the original. While this may seem like an extreme reduction of the dataset, even just 5% of the data is actually larger than commonly used datasets such as CIFAR-10/100. As expected, the overall accuracy of dense training decreases with dataset size. At the same time, we find that the improvement due to IMP also decreases, until pruned network can only maintains the dense accuracy (see SM C.3). The connectivity and locality of the subnetworks thus found is also drastically affected. Considering a fixed sparsity $u = 0.5\%$, Fig. 5**b** shows that decreasing the dataset size the connectivity distribution becomes essentially identical to the one corresponding to pruning weights at random independently of their magnitude (dashed line). Concomitantly, masks become more and more non-local (panel **a**) and less structured (panel **c**). More graphs for other dataset sizes are shown in SM C.3.

Our results show that the dataset size clearly plays a major role for pruning. Our interpretation is that by decreasing the number of data one decreases the signal to noise ratio, hence effectively increasing random idiosyncratic fluctuations in the weights. When such fluctuations become of the same order of magnitude as the ones of the good subnetworks embedded in the original dense one, it becomes impossible to unveil the matching lottery ticket by pruning. In consequence, in this regime the subnetworks obtained by IMP are random, featureless and do not display any local feature. This interpretation is in line with the general idea that less specialized architectures require more data, and explains why we could not employ a smaller dataset in the present work without resorting to very strong augmentation. More results on the disruption of local features due to small dataset sizes are shown in SM B.4.

5.3. The role of the task

To further understand the role played by the properties of the problem—and verify this is not just an average property of the data—we now keep the dataset fix and modify the task instead. We cluster the original 1000 classes of the training set and test set in 10 balanced macro-classes in two different ways: the first one, called *random*, groups the original categories in a non-meaningful way (on the basis of their numerical label in the dataset modulo 10). This produces 10 classes which are all very similar. The second one, called *semantic*, clusters together similar original classes, into macro-categories such as “dog”, “device”, “transport and furnishing”, etc. This produces 10 classes which characterize the data in a meaningful way. A detailed description of

the classes and an example of the difference between these two ways of grouping is reported in SM C.1. After training the accuracy obtained for the random task is at best 13%, while it surpasses 36% for the semantic one. This difference is expected since in the former case elements of different classes are very difficult to be distinguished and training mainly leads to memorization and not learning. More interestingly, pruning is beneficial in the semantic case but not in the random one, and the geometrical properties of the subnetwork induced by pruning are very different in the two cases. In Fig. 5 we show the histogram of the connectivities $C^{\text{in},1}$ and of the local displacements $\mathbf{d}_{ii'}$ for these cases, too. These results reveal that the subnetwork obtained by pruning displays local features and non-trivial connectivity distribution for the semantic task only. In the other case, the subnetwork is essentially featureless, non-local and like one where weights have been pruned at random. More sample masks for the two tasks are shown in SM B.

These results highlight the role of the task in shaping the properties of the network obtained by pruning: only for the task that the network can efficiently learn, and not just memorize, local features emerge. This result is a direct consequence of the *qualitative* difference in the network solutions for the two tasks: features do not simply emerge from the data in an unsupervised fashion, but they represent structures useful for optimization with respect to a meaningful loss. If the task is solved in an unstructured way, e.g. through memorization, the features are simply not selected. In SM C.2 we provide more evidence by focusing on other tasks in between the semantic and the random ones.

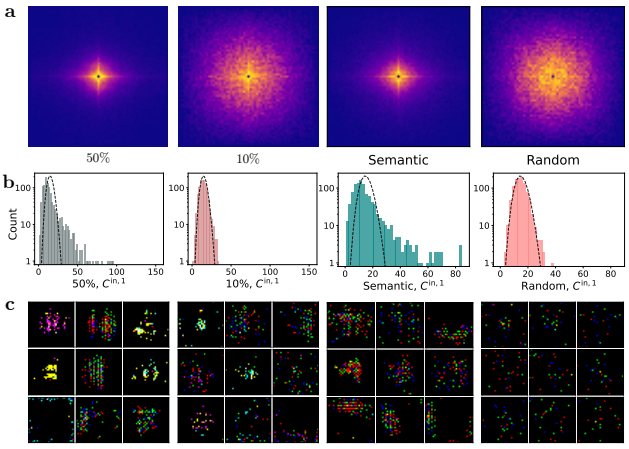

Figure 5. **a:** $S^{\text{sc}}(\mathbf{d})$ for the four tasks at the iteration $u \simeq 0.5\%$. Normalization and color map as shown in Fig. 2. **b:** Histogram of $C^{\text{in},1}$ for the four tasks at the iteration $u \simeq 0.5\%$. The dashed line is the theoretical binomial distribution for random pruning. **c:** Most connected first layer masks for the four tasks.

6. Discussion

Task-specific neural network architectures can great improve accuracy with respect to generic models. One of the most successful cases is that of CNNs, which have been essential in the solution of multiple visual tasks. It is therefore natural to wonder whether some of the characteristics of a CNN architecture could be naturally obtained from the data or the task alone. A recent work (Neyshabur, 2020) takes a first important step in this direction, with a custom modification of the LASSO technique (Tibshirani, 1996), and working on a small but highly augmented dataset, it shows the emergence in training of local and sparse nonzero weights. In this work, we have followed a different perspective. With the aim of characterizing the inductive bias induced by pruning, we have limited ourselves to the original inputs, and we have applied IMP. This approach has two benefits: it naturally leads to *masks*, i.e. it irreversibly changes the architecture of the original network, and it leads to a final model that can be trained *from scratch* to the final accuracy (i.e. from the original initialization values, although these are still correlated with the masks). In combination with the lack of augmentation, this makes features selected from this procedure the natural structure emerging from the dataset and task at hand.

Note that recovering a realistic CNN would be actually prohibitively expensive, as the size of the natural embedding in FCN space is extremely large [3]. Since the pruning procedure is not perfectly efficient, the starting number of nodes would likely be ever larger, easily resulting in a starting network with a billion parameters. As this would prove quite unwieldy for our work, we have opted instead to focus our efforts on the simplest setup which allows a thorough analysis. Pruning this simple network of three moderately sized layers proves indeed sufficient to make several interesting observations: the pruning procedure does not affect all nodes equally, but a small number of them retains a disproportionate amount of inputs. These nodes are particularly important for the performance of the network and they correspond to input masks that are local in space and color channels, with a preference for specific patterns, found throughout the image (although for these images some features are clearly more likely to appear at some specific position than others). At visual inspection, these features bear a strong resemblance to those found in CNNs (Zeiler & Fergus, 2014), the visual cortex (Marĉelja, 1980; Wolf et al., 2005) and obtained with $\beta-$LASSO (Neyshabur, 2020). At high pruning, the features become more local and sparse, and they are combined in the next layer with a preference for similar patterns and close location. While other features

[3] Even for the very small input considered in this work (size 3072), we can roughly estimate with typical 3×3 masks, stride 1 and even the most conservative number of channels, we would require tens of thousands of nodes in the first layers

of CNNs like weight sharing simply cannot be reproduced in this model, the presence of similar masks at different locations is a sign of translational invariance (an experiment with complete translational invariance is presented in SM C.6). In this sense, this procedure could suggest network structures even better aligned with the data from a real dataset, leading to more effective architectures. For instance the sparse sampling, even at this low image resolution, is reminiscent of dilated convolutions (Yu & Koltun, 2016).

As it could be expected, only training on a large enough dataset leads to the emergence of these features. Interestingly, the success of the IMP itself seems to be related to their presence, hinting at the different structure of the solution, in line with recent findings about generalization and prunability (Kuhn et al., 2021). Similarly, for other pruning procedures the presence (or absence) of these features seems to be linked to the ability to improve the performance of the dense network. In the same direction, we have shown that locality and defined features disappear when we replace the task with a more vague one, where IMP is also ineffective. Features are thus not a simple consequence of the dataset, but depend on the structure imposed by the task. This denoising effect could also be related to more universal behaviors as suggested in (Redman et al., 2021). Interestingly, similar-looking structures had been recovered in (Olshausen & Field, 1996) by simply enforcing a sparse coding and high information content: while no task is specified in that setting, the preservation of image-related information seems to play a similar role.

There are several directions which are worth exploring in future works. More complex starting architectures and larger or augmented training sets could be explored, to obtain even closer approximations to CNNs or other more effective architectures. The dependence of the architectural properties induced by pruning on the optimization protocol is another interesting aspect. For instance, we have found that just training until the beginning of overfitting is sufficient to obtain these features (You et al., 2020) (see SM A.3). We have also noticed that precursors are already visible at the early stages of pruning (see video link in SM D), suggesting the possibility of their early identification and more refined search strategies. In addition, it would be interesting to study more pruning algorithms such as (Evci et al., 2020; Lin et al., 2020; Wang et al., 2020; Aghasi et al., 2016; Verma & Pesquet, 2021), especially iterative or inherently sparse ones that could explore structures corresponding to prohibitively large networks, thus solving the problem of size when embedding in fully connected space. Finally, repeating our analysis for tasks different from visual ones, e.g. for audio (van den Oord et al., 2016) and time series (Gamboa, 2017), and study the architectural bias induced by pruning is certainly an interesting direction for future research. This approach could also highlight useful architectures in fields that are still using more standard FCNs such as material modeling (Behler, 2016).

In conclusion, our work underlines the effectiveness of pruning as a tool to uncover not only more efficient networks, but also specific architectural properties associated to the structure of the data and relevant for the training task. From this point of view, pruning schemes could offer a way to uncover effective new architectures for a wide category of problems.

References

Aghasi, A., Abdi, A., Nguyen, N., and Romberg, J. Net-trim: Convex pruning of deep neural networks with performance guarantee, 2016.

Behler, J. Perspective: Machine learning potentials for atomistic simulations. *The Journal of Chemical Physics*, 145(17):170901, 2016. doi: 10.1063/1.4966192.

Blalock, D., Ortiz, J. J. G., Frankle, J., and Guttag, J. What is the state of neural network pruning? *arXiv preprint arXiv:2003.03033*, 2020.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. *arXiv preprint 2005.14165*, 2020.

Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the CIFAR datasets. *arXiv preprint 1707.08819*, 2017.

d'Ascoli, S., Sagun, L., Biroli, G., and Bruna, J. Finding the needle in the haystack with convolutions: on the benefits of architectural bias. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Evci, U., Gale, T., Menick, J., Castro, P. S., and Elsen, E. Rigging the lottery: Making all tickets winners. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 471–481, 2020.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*. OpenReview.net, 2019.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. *arXiv preprint 1912.05671*, 2019.

Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Stabilizing the lottery ticket hypothesis. *arXiv preprint 1903.01611*, 2020a.

Frankle, J., Schwab, D. J., and Morcos, A. S. The early phase of neural network training. *arXiv preprint 2002.10365*, 2020b.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, April 1980. doi: 10.1007/bf00344251.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *arXiv preprint 1902.09574*, 2019.

Gamboa, J. C. B. Deep learning for time-series analysis. *arXiv preprint 1701.01887*, 2017.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

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 (CVPR)*, June 2016.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *arXiv preprint 2102.00554*, 2021.

Hubel, D. H. and Wiesel, T. N. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, Jan 1962. doi: 10.1113/jphysiol.1962.sp006837.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint 1412.6980*, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

Kuhn, L., Lyle, C., Gomez, A. N., Rothfuss, J., and Gal, Y. Robustness to pruning predicts generalization in deep neural networks. *arXiv preprint 2103.06002*, 2021.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

Lee, N., Ajanthan, T., and Torr, P. H. S. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint 1810.02340*, 2019.

Lin, T., Stich, S. U., Barba, L., Dmitriev, D., and Jaggi, M. Dynamic model pruning with feedback. *arXiv preprint 2006.07253*, 2020.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. *arXiv preprint 1810.05270*, 2019.

Marĉelja, S. Mathematical description of the responses of simple cortical cells∗. *J. Opt. Soc. Am.*, 70(11):1297–1300, Nov 1980. doi: 10.1364/JOSA.70.001297.

Morcos, A. S., Yu, H., Paganini, M., and Tian, Y. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. *arXiv preprint 1906.02773*, 2019.

Neyshabur, B. Towards learning convolutions from scratch. *arXiv preprint 2007.13657*, 2020.

Olshausen, B. A. and Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, Jun 1996. ISSN 1476-4687.

Redman, W. T., Chen, T., Wang, Z., and Dogra, A. S. Universality of winning tickets: A renormalization group perspective. *arXiv preprint 1611.05162*, 2021.

Renda, A., Frankle, J., and Carbin, M. Comparing rewinding and fine-tuning in neural network pruning. *arXiv preprint 2003.02389*, 2020.

Ruderman, D. L. The statistics of natural images. *Network: Computation in Neural Systems*, 5(4):517–548, 1994. doi: 10.1088/0954-898X_5_4_006.

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.

Sabatelli, M., Kestemont, M., and Geurts, P. On the transferability of winning tickets in non-natural image datasets. *arXiv preprint 2005.05232*, 2020.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint 1712.01815*, 2017.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint 1409.1556*, 2015.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint 1905.11946*, 2020.

Tanaka, H., Kunin, D., Yamins, D. L. K., and Ganguli, S. Pruning neural networks without any data by iteratively conserving synaptic flow, 2020.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. doi: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Tolstikhin, I., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Keysers, D., Uszkoreit, J., Lucic, M., et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv preprint arXiv:2105.01601*, 2021.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint 1609.03499*, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint 1706.03762*, 2017.

Verma, S. and Pesquet, J.-C. Sparsifying networks via subdifferential inclusion. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10542–10552. PMLR, 18–24 Jul 2021.

Wang, C., Zhang, G., and Grosse, R. Picking winning tickets before training by preserving gradient flow. *arXiv preprint 2002.07376*, 2020.

Wolf, L., Serre, T., and Poggio, T. Object recognition with features inspired by visual cortex. In *Proceedings. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 3, pp. 994–1000, Los Alamitos, CA, USA, jun 2005. IEEE Computer Society. doi: 10.1109/CVPR.2005.254.

You, H., Li, C., Xu, P., Fu, Y., Wang, Y., Chen, X., Baraniuk, R. G., Wang, Z., and Lin, Y. Drawing early-bird tickets: Towards more efficient training of deep networks. *arXiv preprint 1909.11957*, 2020.

Yu, F. and Koltun, V. Multi-scale context aggregation by dilated convolutions, 2016.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. (eds.), *Computer Vision – ECCV 2014*, pp. 818–833, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. *arXiv preprint 1905.01067*, 2019.

A. Convergence tests

In this section we report the result of modifying different hyperparameters of the training or the pruning procedure, to show the robustness of the results presented in the main text.

A.1. Minimization

Fig. 6 shows the best validation and training curves when modifying the optimization procedure either by changing the learning rate or by employing a more advanced minimizer (Adam (Kingma & Ba, 2014)). While there are small differences in the absolute accuracy, the main behavior is robust with respect to the minimization procedure. Although the value 0.5 for the learning rate leads to slightly better accuracy, we have employed a more conservative 0.1 throughout all experiments in the main text, as the final accuracy is not the main scope of this work.

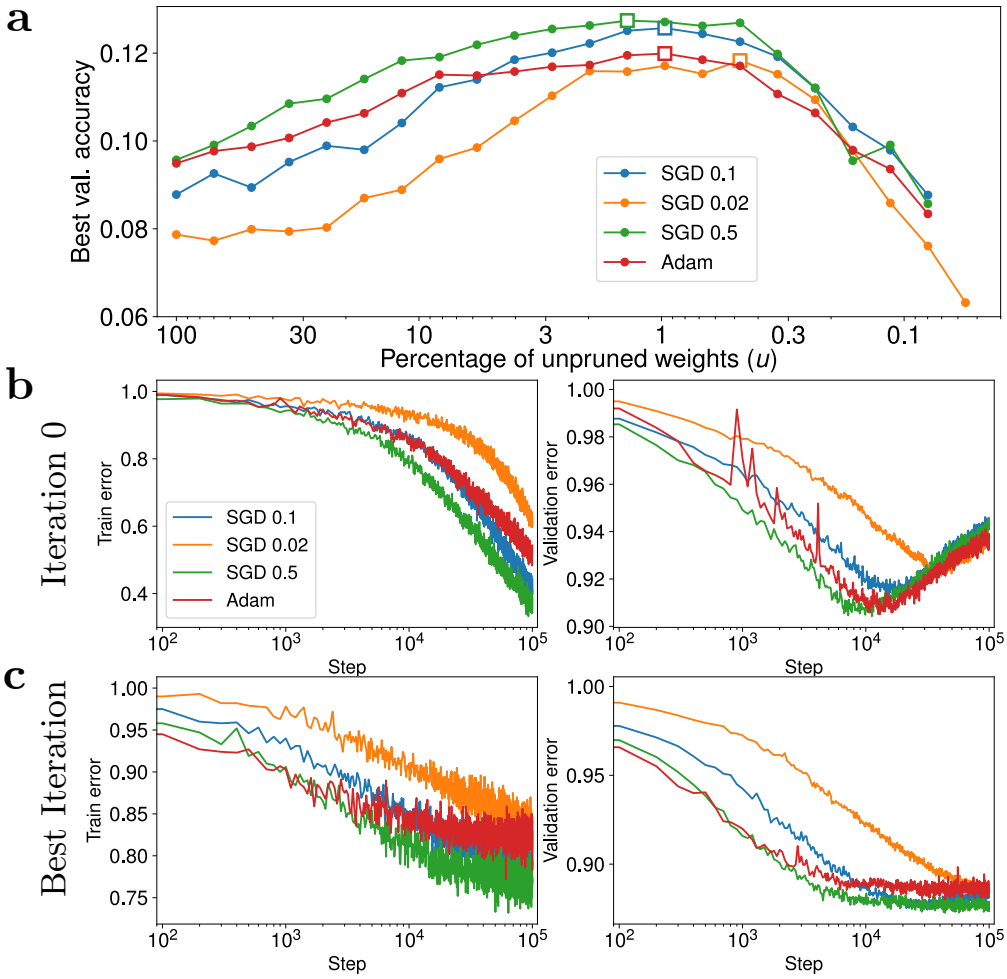

Figure 6. **a:** Best validation accuracy as a function of u for different minimizers: stochastic gradient descent with varying learning rate (SGD in the legend, followed by the learning rate) and Adam with standard parameters and learning rate 0.01. **b:** Training error (left) and validation error (right) for the initial dense iteration for all minimizers. **c:** Training error (left) and validation error (right) for the best iteration of each minimizer (square in panel **a**). While training is much slower, all models are able to reach stationary validation within 10^5 steps, apart from learning rate 0.02 that barely reaches it.

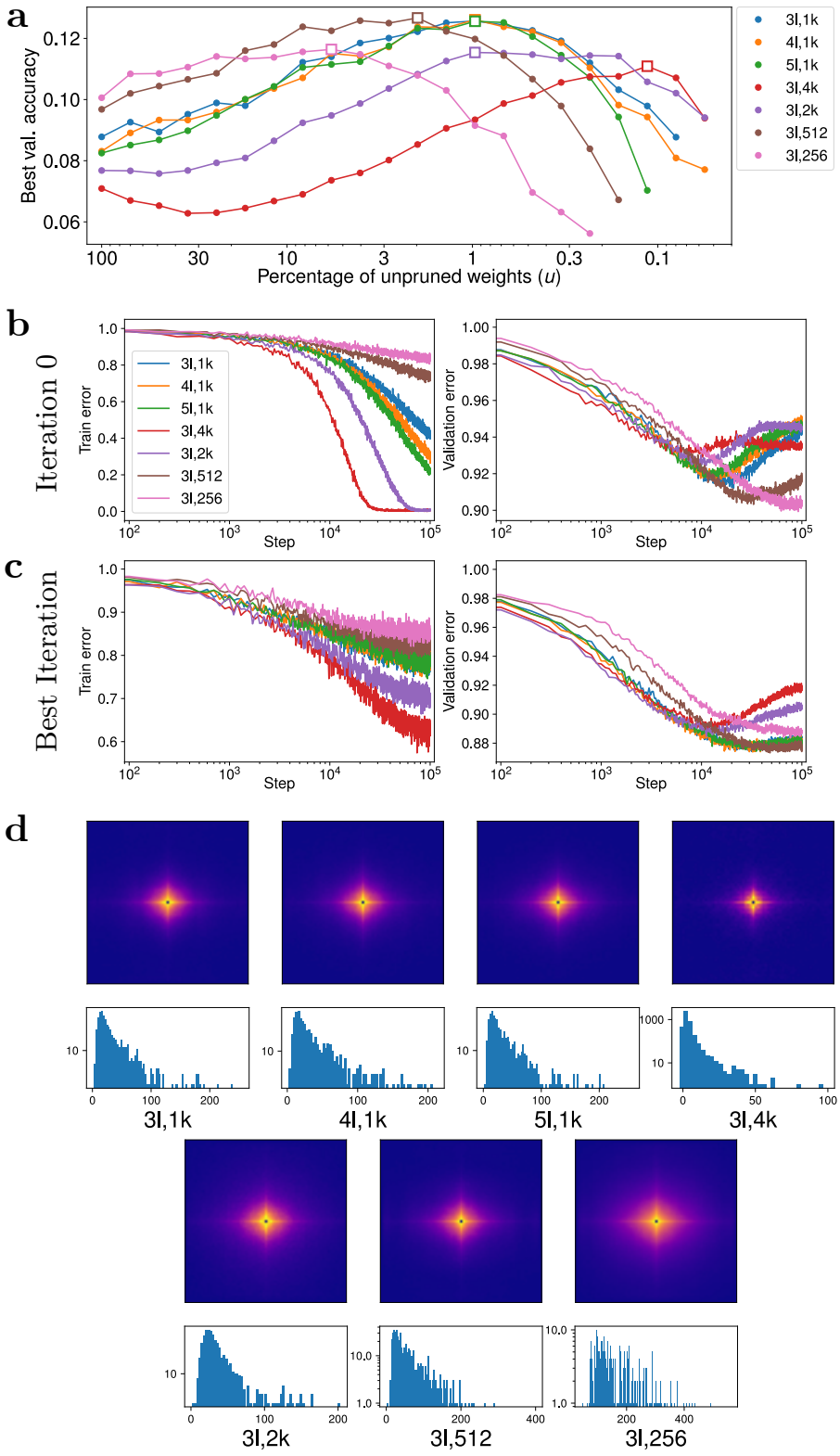

Figure 7. **a:** Best validation accuracy as a function of u for different architectures (see legend and text for notation). **b:** Training error (left) and validation error (right) for the initial dense iteration for all architectures. **c:** Training error (left) and validation error (right) for the best iteration of each architecture (square in panel **a**). **d:** S^{sc} (see Fig. 2 for details) and histograms of $C^{\mathrm{in},1}$ for the best iteration of each architecture (square in panel **a**).

A.2. Other architectures

In this section, we modify the network architecture from the original 3 hidden layers of size 1024 (3l, 1k): we deepen the network with 4 (4l, 1k) or 5 (5l, 1k) layers of the same size, or we change the width of each layer (uniformly) to wider 2048 (3l, 2k) and 4096 (3l, 4k), or smaller 512 (3l, 512) and 256 (3l, 256). All training parameters are kept the same in all these experiments, although better results could be achieved with better minimization and at least the smallest (3l, 256) network could benefit from longer training.

Fig. 7**a-c** shows the best validation and training curves for each architecture. We see that, at least with these parameters, deepening the network does not have a big effect. For the width: a moderately narrower network is as effective, but larger ones are slightly worse. This might be due to the training procedure, but the difference in performance is already present for the dense networks, and the first pruning steps are not as effective in this case, so these networks might be too overparametrized, leading to a qualitatively different solution. Panel **d** shows the locality and connection histogram for the first layer of each network at the best step, indicating that the features discussed in the main text (locality and excess connections nodes) are present in all the cases.

A.3. IMP parameters

We now consider the parameters related to the IMP procedure. In particular, we act separately on 3 fronts: decreasing the pruning ratio per step ($p = 0.2$), decreasing the rewind time for the weights (500, 100 or 0) and varying the total training time per iteration, i.e. the time at which we consider the weights for updating the pruning mask ($5 \cdot 10^5$, 10^4 and $5 \cdot 10^3$). In addition, we keep the original pruning parameters, but prune the weights globally instead of layer-wise (indicated as "Glob"), initializing all weights from the same distribution and excluding only the last layer to output weights from the pruning.

Fig. 8**a** shows the best validation for each of these experiments (train time, rewind time, p in legend). We can see that most curves lead to the same accuracy, apart from training times shorter than the validation minimum $\sim 2 \cdot 10^4$, which is unsurprising. This justifies our practice of stopping training before complete training set overfitting, since the longer run does not offer any benefit. The pruning ratio and rewind time are also clearly converged, we have decided to keep a nonzero rewind time to be more conservative for the other experiments, which could be more brittle. Global pruning also leads to very similar training curves as a function of the global surviving weights. Moreover, the pruning ratio is also quite uniform through layers: e.g. for the best iteration the global value $u = 0.67\%$ corresponds to a layer sparsity of $u = 0.55\%, 0.87\%$ and 0.86% for first, second, and third layer, respectively.

The other panels of the same figure show dense and best iteration training curves and locality and connectivity plots. The only cases showing a deviation from the main text results are once again the very short trains. Interestingly, some of the feature seem to survive even for these cases where the network is not even allowed to reach the best validation in each run, suggesting that some of the relevant features are already present in the early stages of training.

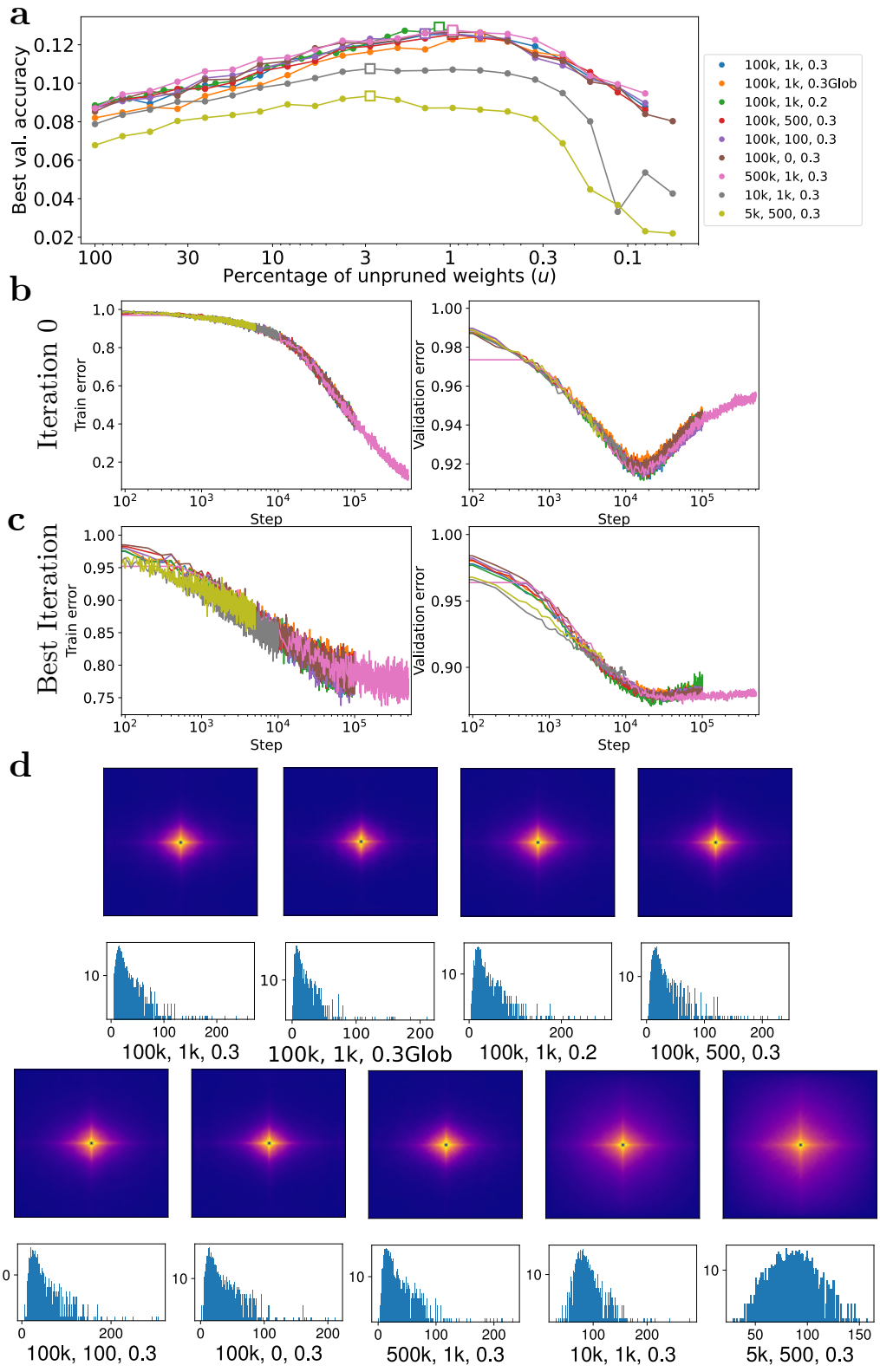

Figure 8. **a:** Best validation accuracy as a function of u for different IMP parameters (train time, rewind time, p in legend). **b:** Training error (left) and validation error (right) for the initial dense iteration for experiments of different total steps. **c:** Training error (left) and validation error (right) for the best iteration of each experiments (square in panel **a**). **d:** S^{sc} (see Fig. 2 for details) and histograms of $C^{\mathrm{in},1}$ for the best iteration of each experiments (square in panel **a**).

B. Further connectivity analysis

In this section we present more specific plots of $S^{\mathrm{sc}}(\mathbf{d})$ and masks for more experiments.

B.1. Random pruning

As a reference, Fig. 9 presents the S^{sc} plot for single nodes pruned at random at different sparsity levels. Even for a fully connected node, the finite size of the image makes local connections more likely than farther connections, but with a decay clearly slower than what shown for localized nodes. For higher sparsity, we see more and more delocalized patterns appearing (of course this is true for single nodes, while the sum of many random sparse nodes would eventually approximate the fully connected one).

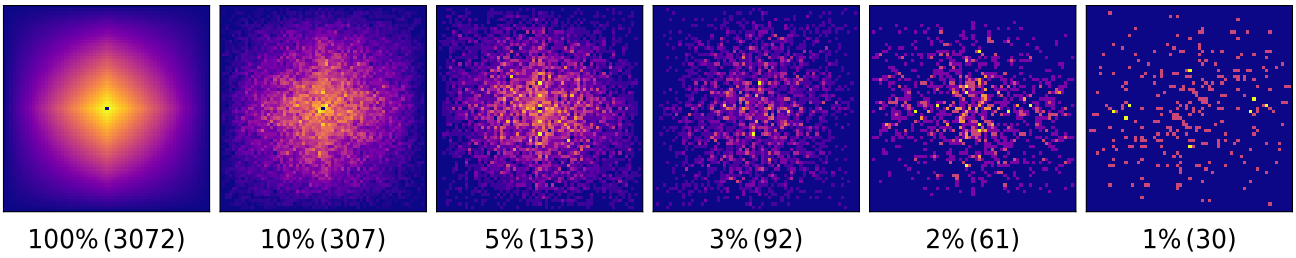

Figure 9. S^{sc} for single nodes randomly pruned at a given u (number of surviving connection in parenthesis, see Fig. 2 for visualization details).

B.2. Input connectivity

Another interesting quantity, shown in Fig. 10 for different networks, is the plot of $C^{\mathrm{out},0}$, i.e. the amount of connections surviving to each pixel. We can see that for well performing networks the connectivity is higher to the center of the image, with some extra connections to the edges and corners, while less effective networks tend to have more uniform connectivity.

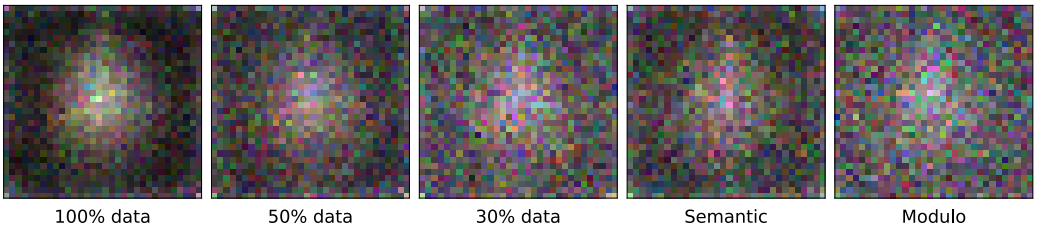

Figure 10. $C^{\mathrm{out},0}$ for different experiments proposed in the main text, at $u \sim 1\%$.

B.3. Locality per number of connections

Fig. 11 presents plots of S^{sc} when we restrict the computation to first hidden layer nodes having values of $C_j^{\mathrm{in},1}$ in a given range. This can give us further insight on whether the locality of a network is due to many close pixels in the more pruned nodes, or few localized clusters in the more connected ones. We present graphs for the standard network, one trained on 30% of the data and the two versions (semantic and random) of the experiments on 10 classes. All networks are at the same level of pruning $u \sim 1\%$.

For the initial network, we see most bins showing some form of locality, more apparent in the most connected bins (some of the later bins contain only a few nodes, strongly affecting the shape of the distance plot). But for the smaller dataset, we see that only some of the bins with more connections seem to show proper locality, with the less connected ones no better than random. A similar behavior is seen in the network trained on 10 semantic classes. In both these cases features are still partially visible, but we can see that the structure is not preserved in all nodes. Finally, the network trained on 10 random classes does not show any locality at any scale.

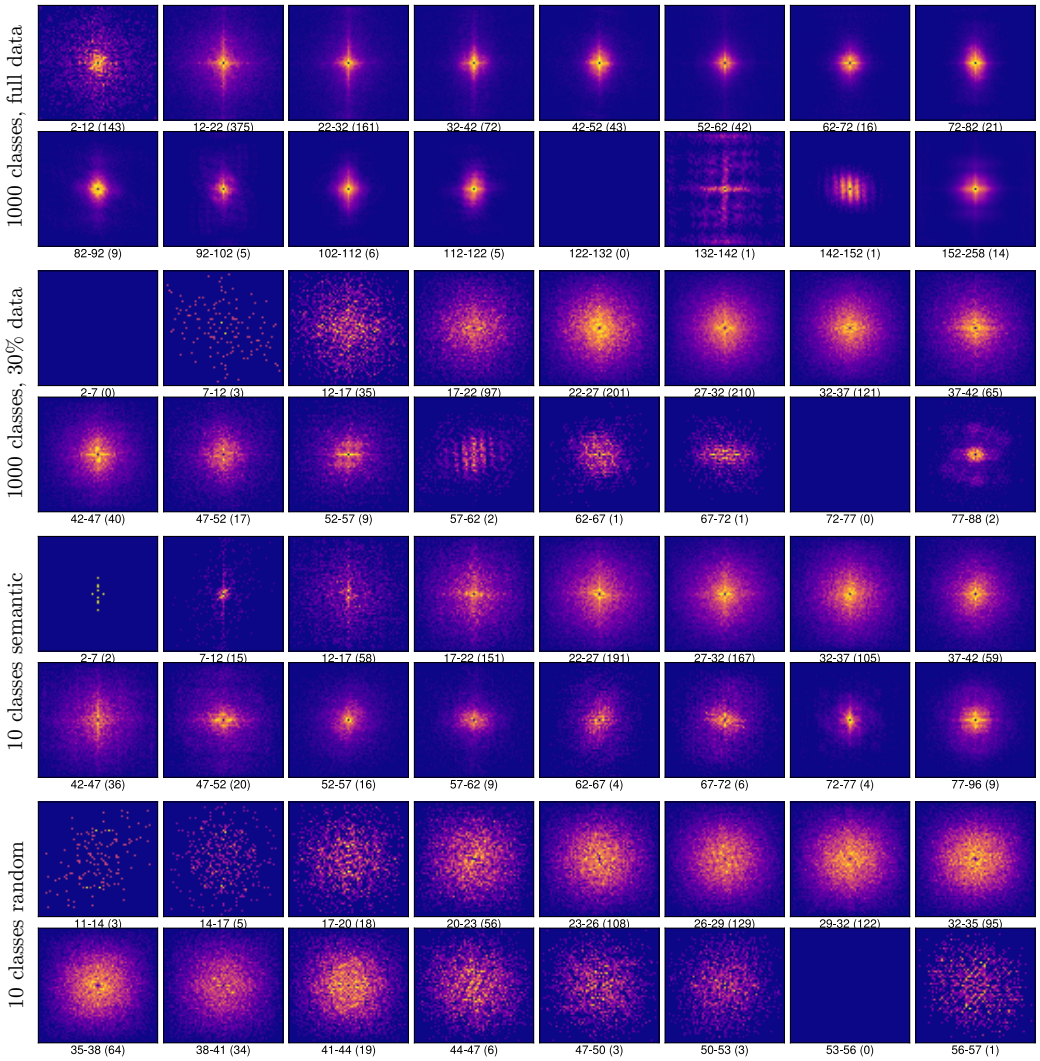

Figure 11. S^{sc} of connections to nodes with different values of $C^{\mathrm{in},1}$. From top to bottom: standard network, training on 30% data, training on 10 semantic classes and training on 10 random classes. All networks are taken at $u \sim 1\%$ Each plot represents the sum of all nodes in an interval of $C^{\mathrm{in},1}$ shown below the image, with the number of nodes falling in the bin in parenthesis. Intervals are of 10, 5, 5, and 3 bins respectively, to cover the range of connectivity of each network. The last bin covers all remaining nodes.

B.4. Highly connected masks

We show here a larger sample of masks of the most connected nodes. Fig. 12 covers the original network of Fig. 1: for the best sparsity $u \sim 1\%$ we show the top 120 most connected layer 1 nodes, showing the variety of features sifted out, and the top 30 effective masks μ_{ij} of layer 2, showing how they cover most of the image. For the higher sparsity $u \sim 0.3\%$ we show the 60 most connected of the first layer, highlighting how features are preserved, although more local and with higher sparsity. We also show the effective masks of layer 2, partially local, and of layer 3, covering most of the image.

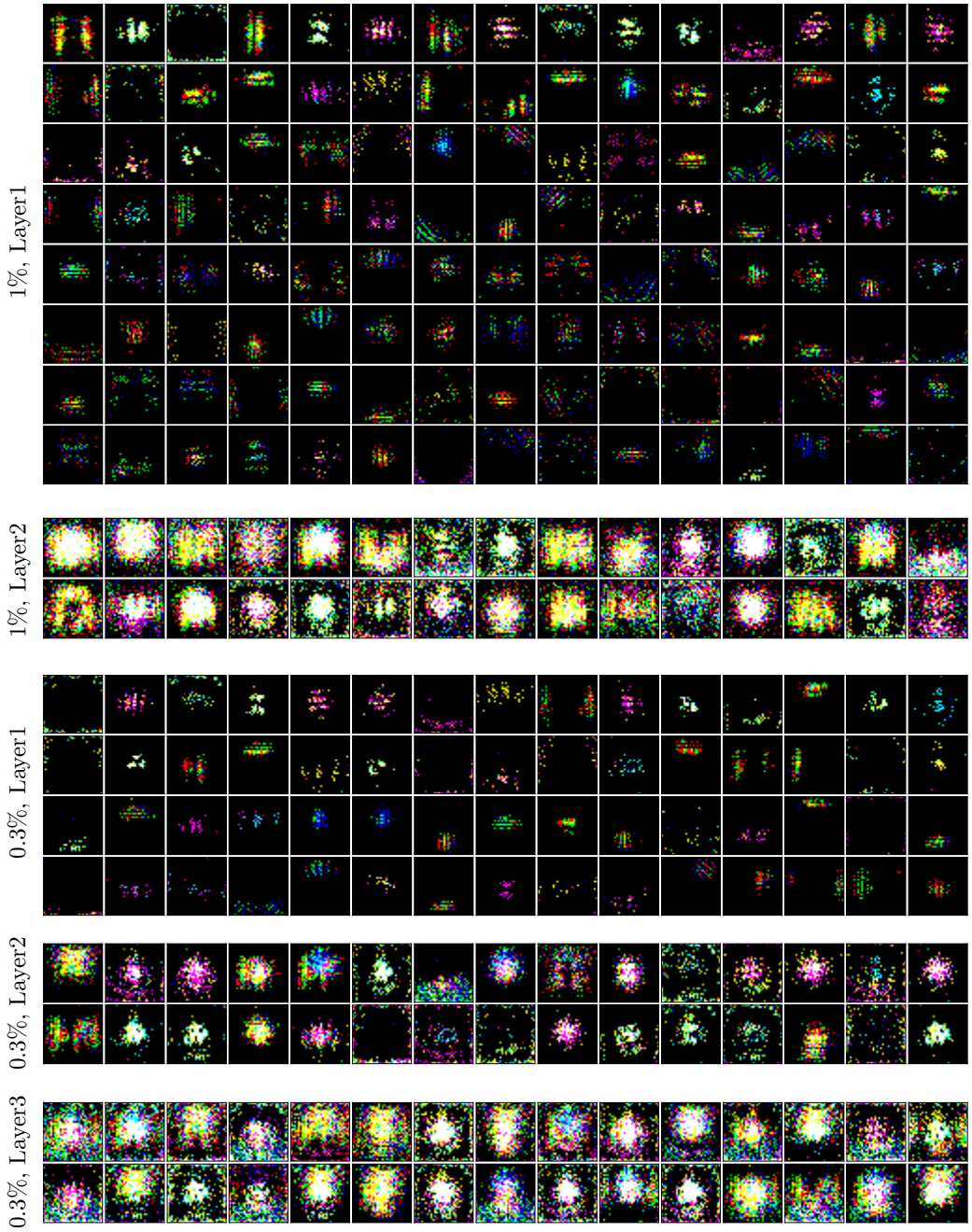

Figure 12. Masks of the most connected nodes for different layers and sparsity of the original network (same representation as Fig. 3).

Fig. 13 shows the 30 most connected masks in the first layer for other networks: trained on 50% of the data, 30% of the data, 10 semantic classes or 10 random classes. We can see more and more featureless masks among the most connected as we

shrink the dataset. For the semantic classes, we still find local structures, although less clear than the original task, while the random classes show no structure at all.

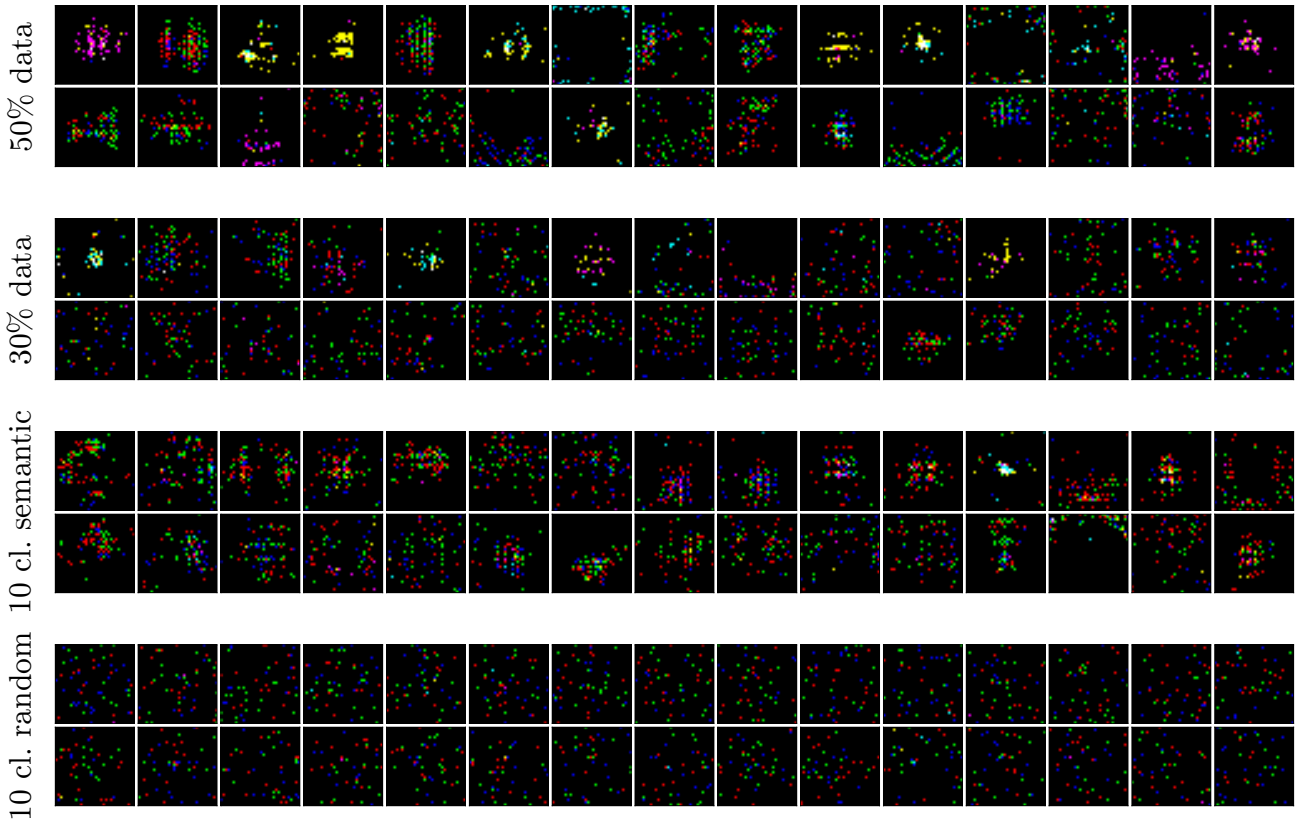

Figure 13. Masks of the most connected nodes for different experiments: original task on 50% and 30% of the data, and task modified to 10 super-classes either semantically or randomly grouped.

B.5. Connectivity for other pruning approaches

In Fig. 14, we show the locality through $S^{\text{sc}}(\mathbf{d})$ and histograms of connectivity of the first layer of networks pruned at initialization through SNIP (target sparsity 90% and 99%), and SynFlow at sparsity (90%).

While SNIP, employing the data even just in one show, obtains relatively localized masks, the data free approach of SynFlow leads to no local structure. Similarly, while SNIP connectivity shows the same long tails as IMP, SynFlow retains few highly connected nodes, while completely pruning the others.

B.6. Masks hierarchy

For the original network, at IMP $u \sim 0.3\%$, Fig. 15 shows a more in depth analysis of the masks μ_{ij} for the most connected nodes of the second hidden layer. The masks are shown in the center of the image and below them are the 3 images of the validation set that activate the nodes the most. For 3 sample nodes, the top of the image shows explicitly all the first hidden layer nodes connected to them: they have rather uniform features and a specific local spatial distribution. Although not as clean as in a real CNN, we can see how the first layer highlights local features, that are aggregated in a larger but still localized fashion in the second layer, often aggregating similar features at different position, like convolutions preferring a specific channel.

To obtain a more clear hierarchical structure, we then consider a deeper network of 8 layers (all of size 1024) and prune them one at a time starting from the input, to increase the probability of finding meaningful connections within a relatively

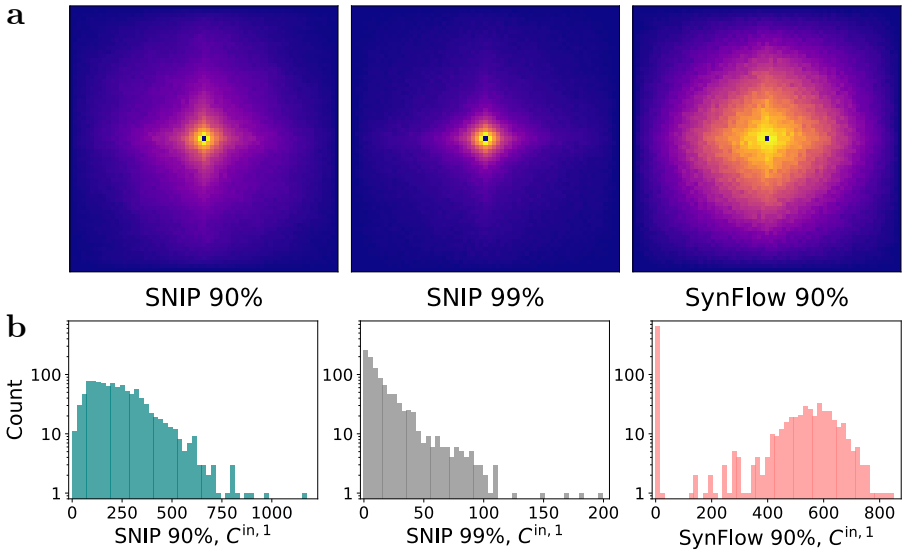

Figure 14. $S^{\text{sc}}(\mathbf{d})$ (**a**) and histograms of $C^{\text{in},1}$ (**b**) for masks obtained through different pruning algorithms: SNIP at two different sparsities and SynFlow.

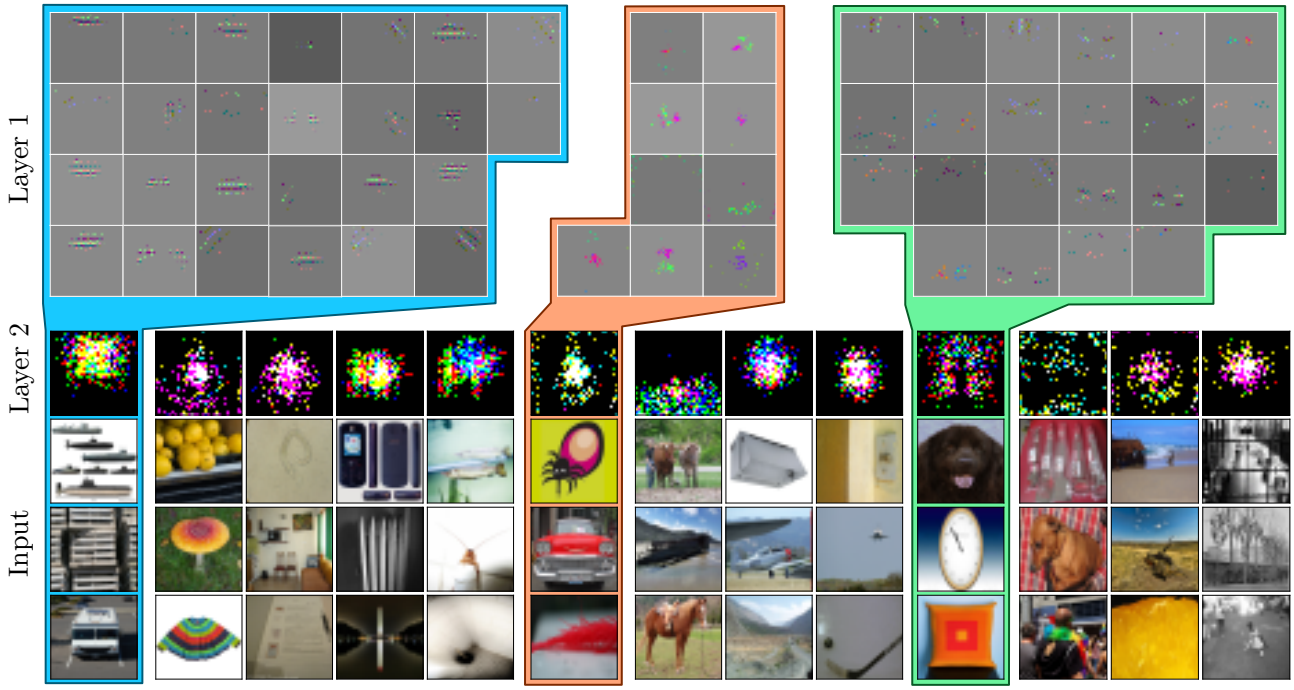

Figure 15. Masks of the second layer from input $\mu_{ij} = \theta(\sum_k m^1_{ik} m^2_{kj} - 1/2)$ with most nonzero element (center, marked "Layer 2"), images with the largest activation on them (below, marked "Input") and for 3 of them masks in the first layer connected to them (top, marked "Layer 1"), showing the masked weights $w^1_{ij} m^1_{ij}$ as detailed in Fig. 3.

narrow network.

Specifically, we prune each layer until the resulting network starts having performances lower than the initial dense network, then we move to the next layer freezing the last state of the previous ones. This results in the first 3 layers reaching $u = 0.08\%$, 0.33%, and 0.48% sparsity, respectively (see Fig. 16**a**). This procedure highlights more clearly the hierarchical

structure of the network. As as example, in Fig. 16**b** we trace the connectivity of a single layer 3 node through the connected masks of layers 2 and 1. Within the variability of a stochastic procedure, we can see that progressively larger receptive fields are covered through the layers.

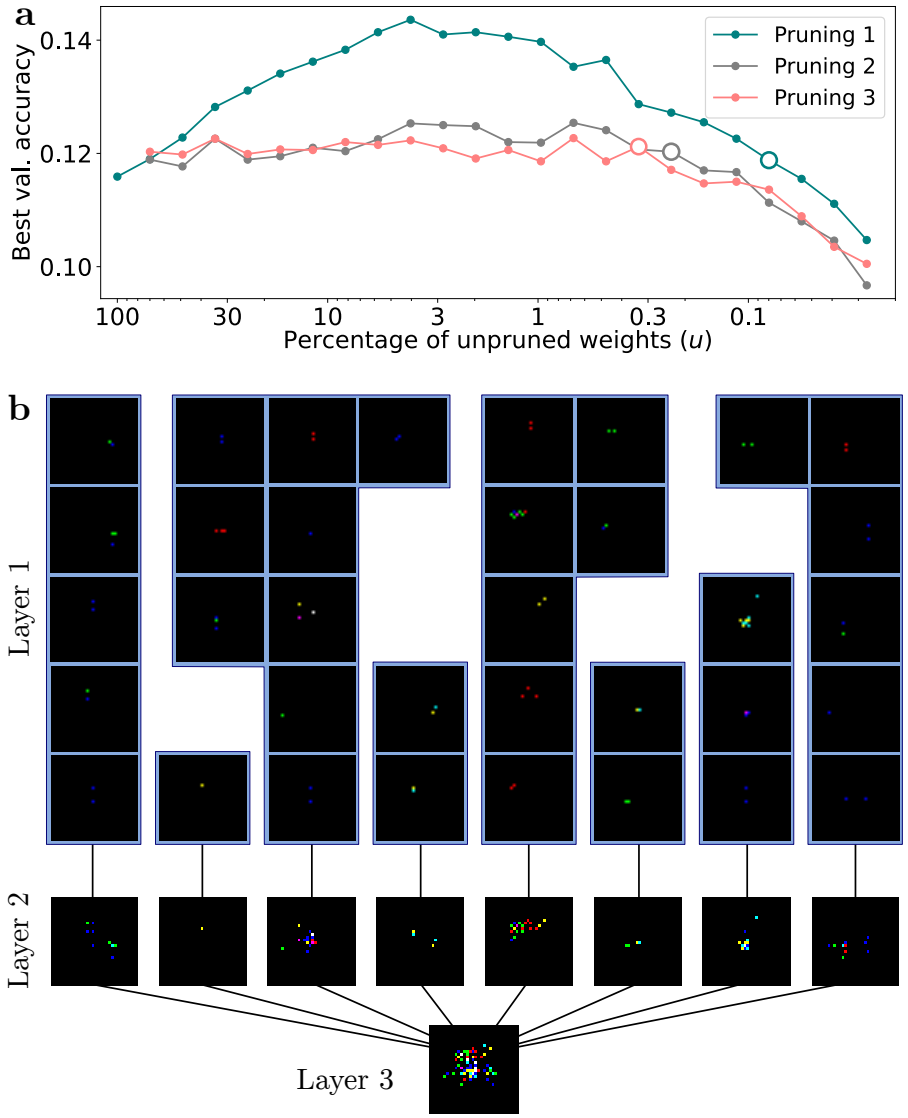

Figure 16. **a:** Best validation accuracy as a function of u for progressive IMP of each layer. For each layer, the last iteration with better performance than the dense network (large white dots) is taken as a starting point for the next IMP (and for masks analysis). **b:** Hierarchical structure leading to a specific third layer mask at high pruning: for each layer all masks retaining connections to the node in the next layer are shown.

Table 1. Proposed 10 aggregated ImageNet classes: for each class we give a rough definition of the class in terms of macro-categories, the number of original ImageNet classes and single images in those classes that fall into this class (the symbol A\B indicates all classes that fall in the macro class A but not in B).

#	CLASS COMPOSITION (ROUGH)	#CLASSES	#ELEMENTS
0	DOG	118	147873
1	MAMMAL\DOG	100	129728
2	BIRD, REPTILE	95	122895
3	FISH, AMPHIBIAN, INVERTEBRATE	85	110034
4	DEVICE\MUSICAL INSTRUMENT	104	131965
5	COVERING, MUSICAL INSTRUMENT	111	142423
6	CONTAINER\VEHICLE, APPLIANCE, EQUIPMENT	99	126603
7	TRANSPORT, FURNISHING	95	122661
8	PLANT, FUNGUS, FOOD, VEGETABLE, FRUIT, IMPLEMENT	96	122701
9	CONSTRUCTION, MISCELLANEOUS	97	124282
	TOTAL	1000	1281167

C. Modified tasks

We report here some experiments on modified datasets, that reinforce some of the statements of the main text

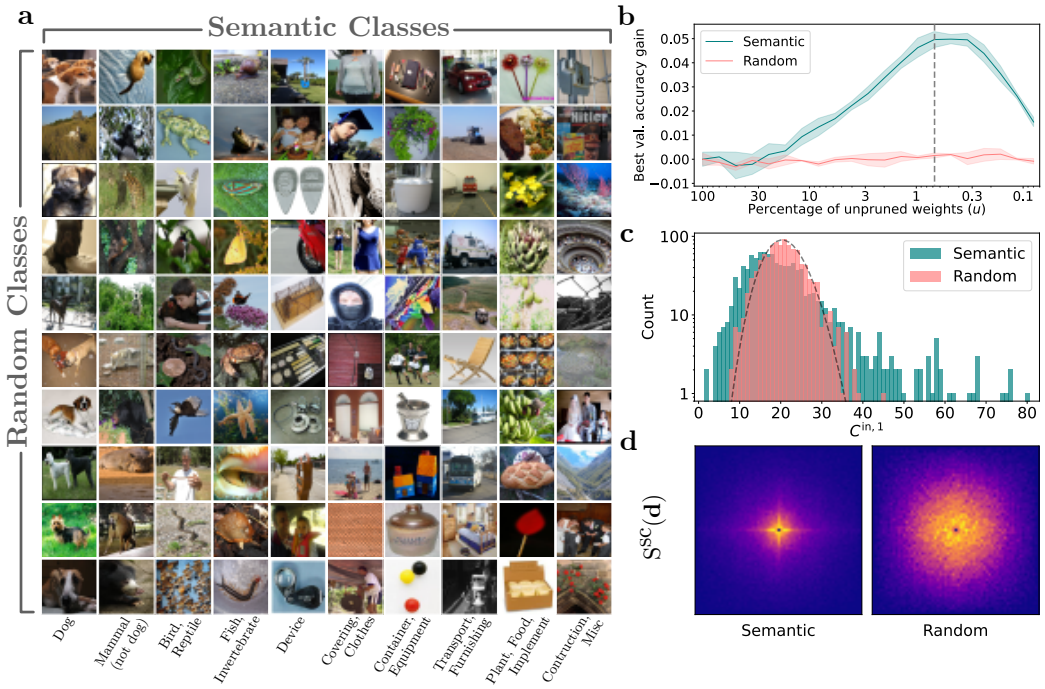

Figure 17. **a:** 10 sample images for each of the 10 macro classes. Each column shows a different *semantic* cluster (rough label at the bottom) where images can be seen to have similar content, while each row is a different *random* cluster, and no structure is visible. **b:** Gain in best validation accuracy (difference from the value corresponding to the unpruned network) for the random and semantic tasks. The lines are averages of 4 independent runs, with the shaded area representing one standard deviation. Each training was run for 500k steps for the semantic and 300k steps for the random clustering to ensure convergence. The dashed line marks the iteration at $u \simeq 0.7\%$ used in panels **c** and **d**. **c:** Histogram of $C^{\text{in},1}$ for the two cases at the iteration $u \simeq 0.7\%$. The dashed line is the theoretical binomial distribution for random pruning. **d:** $S^{\text{sc}}(\mathbf{d})$ for the two cases at the iteration $u \simeq 0.7\%$. Normalization and color map as shown in Fig. 2.

C.1. Class clustering

We give here more details about the construction of 10 *semantic* super-classes. Our goal was to design 10 classes that would: include all the original 1000, be similarly sized, and be easily identified in terms of larger WordNet categories.

The details of the final classes constructed are summarized in Table 1. The animals are divided into 4 categories (the large amount of dogs naturally falls into a slightly larger category). Objects make up most of the remaining 6 classes roughly based on sub-categories of "artifacts": device, covering (mostly clothes), container (excluding vehicles), transport and furnishing, implement and construction. Class 8 also contains all plant-derived categories and food. Class 9 also includes all categories not included in one of the previous macro categories, listed as miscellaneous (including e.g. some natural formations). A few classes (21) are included in more than 1 macro class by this division, and they were arbitrarily assigned to a single one. The specific attribution of each of the original classes is available in the code repository specified in D.

10 examples from each semantic class (columns) and 10 from each random class (rows) are shown in Fig. 17**a**. The same figure also shows the gain in accuracy through IMP, connectivity, and locality for the two tasks.

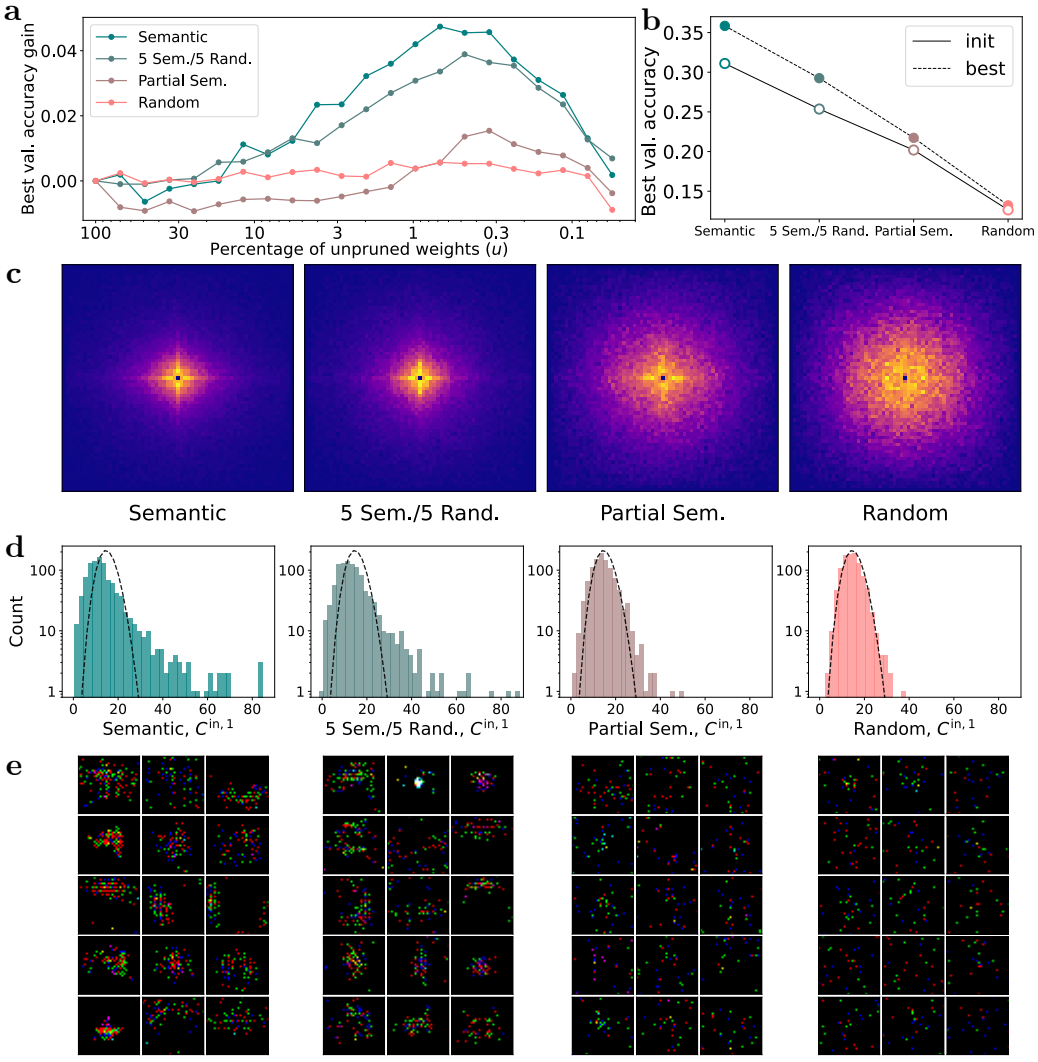

Figure 18. **a:** Gain in best validation accuracy (difference from the value corresponding to the unpruned network) for tasks from semantic to random. **b:** Best validation accuracy for the unpruned network (empty dots) and best IMP iteration (full dots) for the different tasks. **c:** $S^{\mathrm{sc}}(\mathbf{d})$ for the four tasks at the iteration $u \simeq 0.5\%$. Normalization and color map as shown in Fig. 2. **d:** Histogram of $C^{\mathrm{in},1}$ for the four tasks at the iteration $u \simeq 0.5\%$. The dashed line is the theoretical binomial distribution for random pruning. **e:** Most connected first layer masks for the four tasks.

C.2. More super-classes

We consider more ways of creating 10 macro classes from the original one. To create a task in between the "semantic" and "random" ones, we consider 2 options: one we call "5 semantic/5 random", where the first 5 classes are the ones described in Sec. C.2, while the other ones are random groupings of the remaining categories. The other we call "partial semantic" where each of our super-classes is composed of half (around 50) of the categories of the "semantic" classes and the other half picked at random between the remaining categories.

To gauge the complexity of the task, Fig. 18**b** reports the unpruned and best accuracy for each of these experiments, confirming that they are in between the two extremes, the one with 5 "fully meaningful" classes being simpler than the other. The full best validation curves (minus the unpruned values) reported in panel **a** and the locality and connectivity plots in panels **c** and **d** support the idea of a gradual progression between a network characterized by local structures and one where the task is performed in a different way. Lastly, the most connected first layer masks in panel **e** highlight the progressive disappearance of features.

C.3. Smaller dataset

Fig. 19 shows the gain in accuracy through IMP, connectivity, and locality for trainings using different fractions of the dataset.

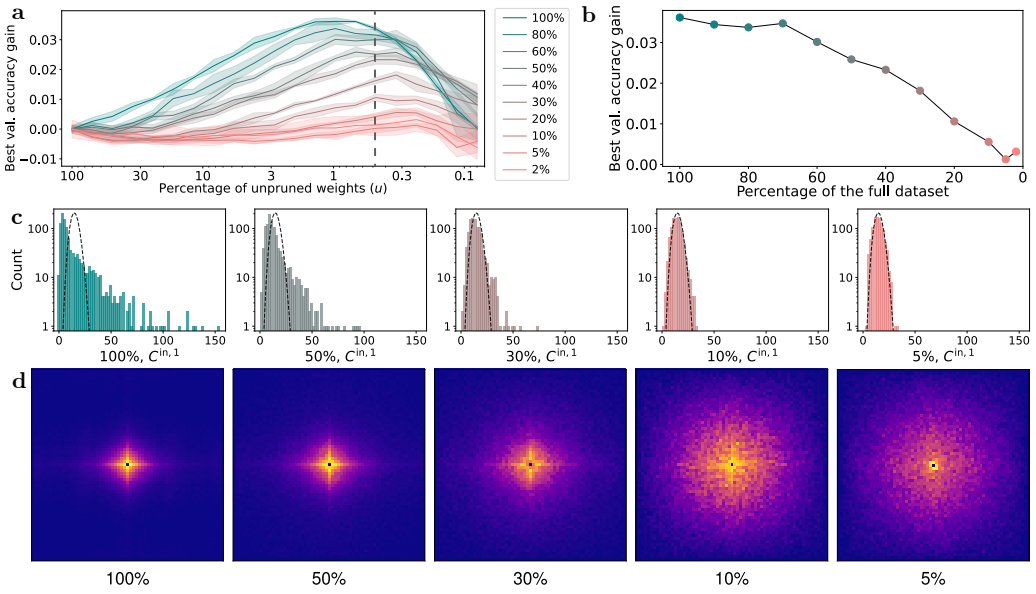

Figure 19. **a:** Gain in best validation accuracy (difference from the value corresponding to the unpruned network) as a function of u for training on different percentages of the original dataset. The lines are averages of 4 independent runs, with the shaded area representing one standard deviation. The dashed line marks the iteration at $u \simeq 0.5\%$ used in panels **c** and **d**. **b:** Largest gain in validation accuracy during the IMP procedure for different dataset sizes (percentage of the whole dataset). **c:** Histogram of $C^{\text{in},1}$ for different dataset sizes (see label at the bottom) at the iteration $u \simeq 0.5\%$. The dashed lines represent the theoretical binomial distribution for random pruning. **d:** $S^{\text{sc}}(\mathbf{d})$ for different dataset sizes at the iteration $u \simeq 0.5\%$. Normalization and color map as show in Fig. 2.

C.4. Rotated images

We rotate all images 20 degrees counterclockwise, and crop to obtain new 32×32 images (we mask out the pixels not corresponding to any original pixels). This is done to verify that the vertical and horizontal directions have nothing to do with the discretization of the images. The overall training is similar to the original one, with slightly lower accuracy. Fig. 20 shows $S(\mathbf{d})$ and masks at the best iteration. As expected, the preferred directions are rotated and masks tend to follow the new horizontal and vertical alignment.

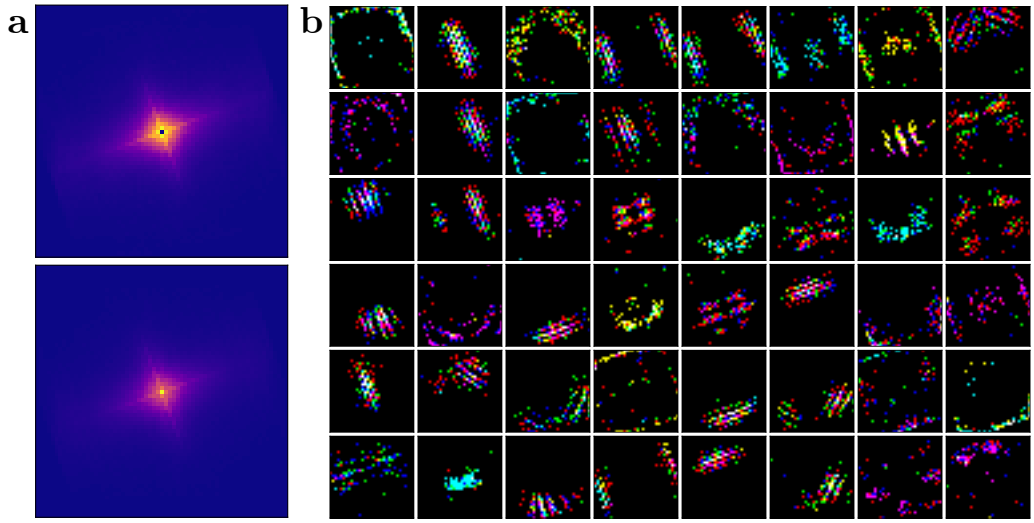

Figure 20. **a:** S^{sc} and S^{dc} for the best iteration of the experiment on rotated images. **b:** masks of the 48 most connected layer 1 nodes for the same iteration.

C.5. ImageNet64

We train a network on the 64×64 pixels, higher resolution ImageNet downsampling from the same source: ImageNet64 (Chrabaszcz et al., 2017). We use exactly the same parameters of our main experiment, and obtain only slightly lower accuracy and a similar IMP curve as the lower resolution version, as shown in Fig. 21**a**. We analyze the best iteration at $u \sim 0.3$: locality is still preserved as shown in panel **b**, with the main axes being preferred even more sharply. The 90 most connected layer 1 masks (panel **c**) show similar features to the ones found elsewhere, with even clearer preference for sparse sampling and separated lines along the main orientations.

C.6. Translated images

For this experiment, we enforce complete translational invariance by translating each image vertically and horizontally by a random amount (in $[0, 31]$) modified every time a mini-batch is created, and wrapping the image in both axes (periodic boundary conditions). This clearly causes the appearance of unrealistic boundaries in the image, but achieves the goal of this experiment or rendering each pixel is equivalent, on average. We validate this network on the original, untranslated images, achieving lower accuracy then the original network, as might be expected since the training dataset is much larger and not matching the validation dataset (we also do not re-optimize the training parameters, and just run long trainings of 500k steps). The IMP validation curve in Fig. 22**a** shows a peak at lower sparsity ($u \sim 8\%$), consistent with the modified dataset. Analyzing this best iteration, the S^{sc} map in panel **b** indicates local connectivity, with a stronger bias towards the horizontal and vertical directions. The large variety of maps in panel **c** (and their weightd counterparts in panel **d**) show surviving features in the shape of stripes and patches patterns with different orientations and periodicity. Despite the synthetic nature of this dataset, it is interesting to observe these patterns emerge from a translationally invariant dataset. Moreover, the combination of these features with the spatial focus of the original dataset (also visible in Fig. 10) helps explain the emergence of Gabor-like filters.

D. Supplementary Material

A repository with all the code required to reproduce the results of this paper is available at `https://github.com/phiandark/SiftingFeatures`. The main code is written in Python and based on tensorflow (1.xx). Input files are provided to reproduce our experiments and a jupyter notebook is available to post-process the data and recreate the figures of this work.

The same repository hosts a text file with the attribution of each ImageNet category to one of the 10 macro-class in our semantic clustering experiment.

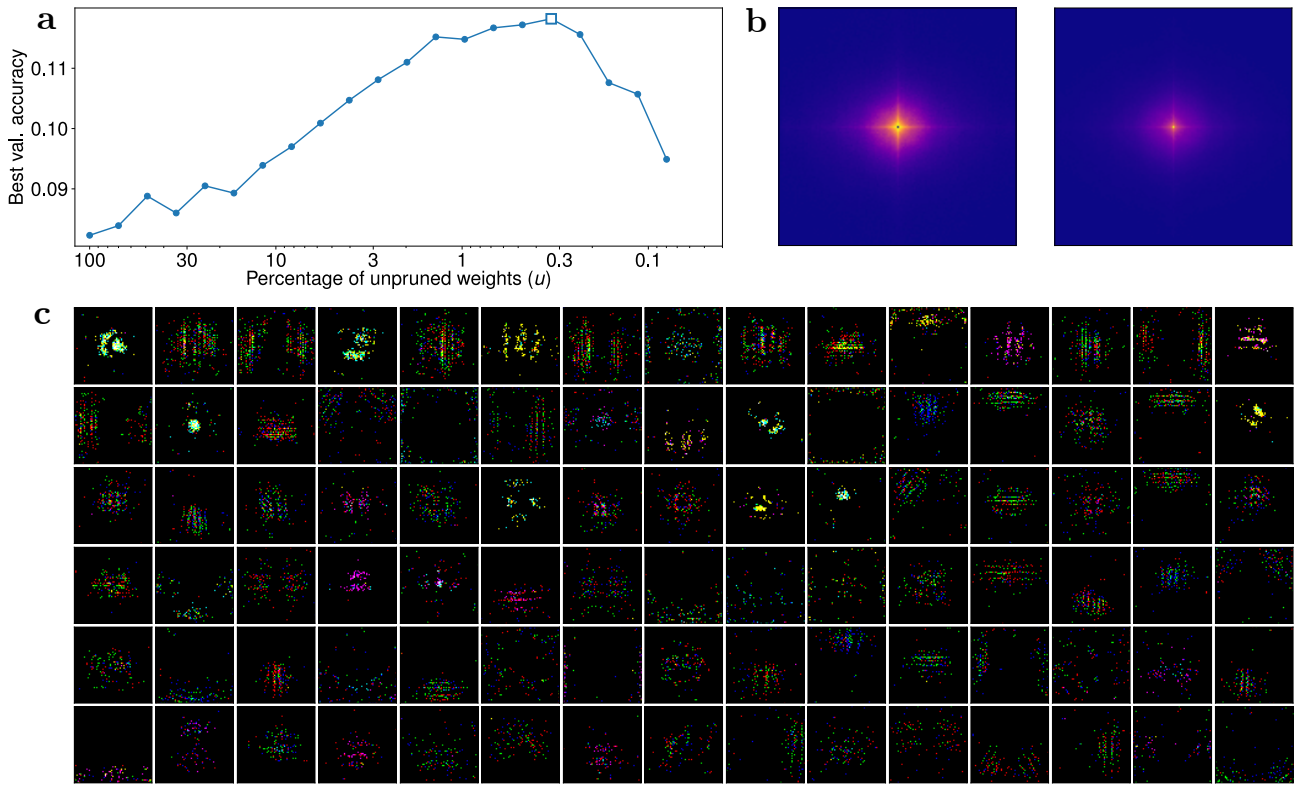

Figure 21. **a:** Best validation accuracy as a function of u for ImageNet64 dataset. **b:** S^{sc} and S^{dc} (see Fig. 2 for details) for the best iteration at $u \sim 0.3\%$ (square in panel **a**). **c:** 90 most connected layer 1 masks for the same iteration.

Videos are also available in the same location, showing the evolution of the most connected masks and their weighted versions during IMP for the main task of this work.

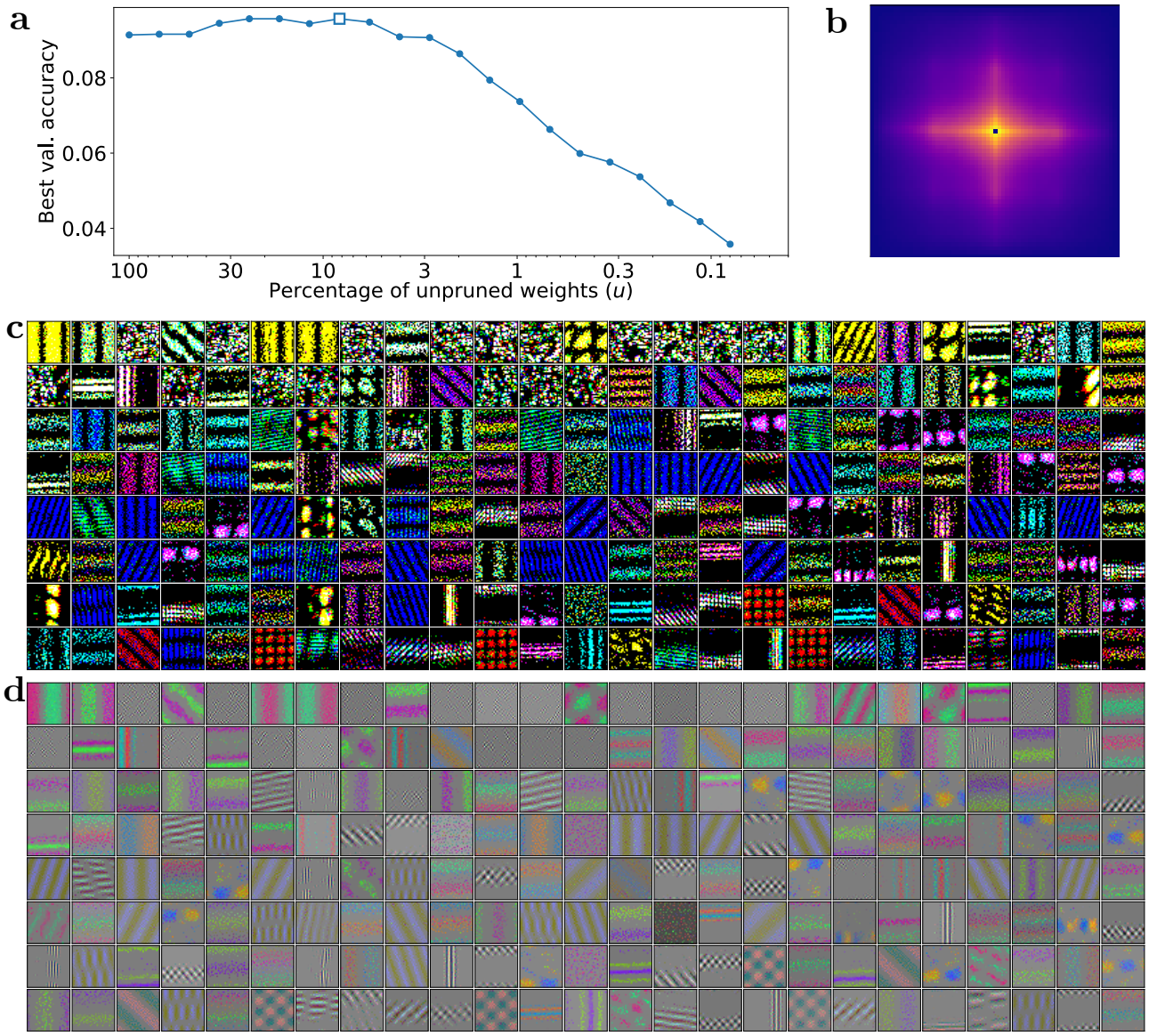

Figure 22. **a:** Best validation accuracy as a function of u for randomly translated dataset. **b:** S^{rsc} (see Fig. 2 for details) for the best iteration at $u \sim 8\%$ (square in panel **a**). **c:** 200 most connected layer 1 masks for the same iteration. **d:** The same masks, multiplied by the relative weights (see Fig. 3 for details).