Why do CNNs Learn Consistent Representations in their First Layer Independent of Labels and Architecture?

Rhea Chowers  
School of Computer Science and Engineering  
The Hebrew University  
Jerusalem, Israel  
rhea.chowers@mail.huji.ac.il

Yair Weiss  
School of Computer Science and Engineering  
The Hebrew University  
Jerusalem, Israel  
yair.weiss@mail.huji.ac.il

Abstract

It has previously been observed that the filters learned in the first layer of a CNN are qualitatively similar for different networks and tasks. We extend this finding and show a high quantitative similarity between filters learned by different networks. We consider the CNN filters as a filter bank and measure the sensitivity of the filter bank to different frequencies. We show that the sensitivity profile of different networks is almost identical, yet far from initialization. Remarkably, we show that it remains the same even when the network is trained with random labels. To understand this effect, we derive an analytic formula for the sensitivity of the filters in the first layer of a linear CNN. We prove that when the average patch in images of the two classes is identical, the sensitivity profile of the filters in the first layer will be identical in expectation when using the true labels or random labels and will only depend on the second-order statistics of image patches. We empirically demonstrate that the average patch assumption holds for realistic datasets. Finally we show that the energy profile of filters in nonlinear CNNs is highly correlated with the energy profile of linear CNNs and that our analysis of linear networks allows us to predict when representations learned by state-of-the-art networks trained on benchmark classification tasks will depend on the labels.

1 Introduction

Convolutional Neural Networks (CNNs) are being applied daily in numerous image analysis tasks, from medical image processing to autonomous cars, and have reached "human-like" accuracy in benchmark classification tasks. It is often assumed that much of this power derives from the high-level representations that these networks extract from images: rather than representing each image as a vector of RGB values at each pixel, each layer in a CNN provides an alternative representation of the input image. A large body of work shows that using these intermediate representations in order to represent images can yield excellent performance on a wide range of tasks, even if the task is very different from the task on which the network was trained [6, 13, 26].

In this paper, we seek to understand the representation that is learned in the very first layer of a deep CNN. In some sense, the transformation in the first layer is trivial: each patch in the input image is represented by the output of \( M \) linear filters. In the case of VGG [24], after the first layer each \( 3 \times 3 \times 3 \) RGB patch is represented by a 64 dimensional vector, which is a linear transformation of the patch. Yet this linear transformation can have a strong effect on the similarity of different patches: Fig. 1a shows a scatter plot of distances between pairs of patches when using the original 27 numbers vs. using the 64 numbers from the first layer of a pretrained VGG network. When the RGB distance is small, then (not surprisingly) so is the VGG distance. But two patches with large RGB distances may
be close or far in VGG distance. In a recent paper on perceptual similarities [3] the similarity metric induced by pretrained VGG networks was compared to that of VGG networks with random weights. It was found that "most of these differences [between the similarity metric of random weights and pretrained weights] disappear when using trained weights only for the convolutions in the very first layer of VGG and random weights for all other layers".

![Figure 1](image.png)

Figure 1: The first layer of VGGs of different depth trained on CIFAR10 induce highly correlated distance metrics (1b) relative to comparing the metric with distance between patches in RGB space (1a). The high correlation (in parentheses, calculated over initializations and samples of patch pairs) is maintained even when training VGG with random labels vs VGG trained with the original labels (1c, and see Section 2.2 for details).

The primary motivation for our paper is our observation that the representations learned by the first layer of a deep CNN are not only very different from the RGB representation, but also remarkably consistent for different architectures and initialization. Fig. 1b compares distances between pairs of patches when computed using the first layer of a pretrained VGG11 and pretrained VGG16. Despite the different initializations and architectures, the induced distance metrics are remarkably similar with a correlation coefficient over 0.97.

It has previously been observed that different architectures and initializations of deep CNNs often learn Gabor filters, i.e. localized filters that are sensitive to different spatial frequencies, in their first layer [26, 23, 18]. Yet this does not explain the quantitative consistency shown in Fig. 1b: the similarity between patches will depend on the distribution of Gabors at different spatial frequencies. Why do different networks that optimize a very non-convex loss learn almost exactly the same distribution over Gabors and spatial frequencies?

One possible answer for the consistency of representations is that trained CNNs learn to emphasize features that are useful for object recognition. A second surprising observation that motivates our work is shown in Fig. 1c: when we train a VGG architecture on images with random labels, the learned similarity is nearly identical to the one learned when using the true labels. It is therefore highly unlikely that the consistent representation that we see is driven by the network’s attempt to accentuate features that are useful for recognition.

The organization of this paper is as follows. We start by systematically and quantitatively characterizing the filters that are learned in the first layer of a deep CNN and show that the representation is indeed consistent across initial conditions, architectures and labels. To understand this effect, we derive an analytic formula for the sensitivity of the filters in the first layer of a linear CNN trained with gradient descent at any training iteration. We prove that under a realistic assumption that when the average patch of images in the two classes is identical, the sensitivity profile of the filters in the first layer will be identical in expectation when using the true labels or random labels and will only depend on the second-order statistics of image patches and the learning hyperparameters. Finally we show that our analysis of linear networks allows us to predict when representations learned by state-of-the-art networks trained on benchmark classification tasks will depend on the labels.

---

1 In fact, it possible to design a Gabor representation so that the distance between patches will be identical to RGB distance [12], but this transformation is not learned by the networks.
2 Quantifying the Consistency of First Layer Representations in CNNs

Learned weights of CNNs are high dimensional, therefore visualization and comparison of them may be complex. In particular, since we are interested in the similarity between distance metrics induced by the filters, a set of filters and an orthogonal transformation of the same filters will give the same similarity, even though all the filters will be different. Recent works [14, 20] suggest comparing two representations based on the distance between the distribution over patches induced by the two representations. But estimating this distance in high dimensions is nontrivial and it does not allow easy visualization of the weights. In this paper we propose a new method which is especially relevant for linear representations.

Given two patches \(x_1, x_2\) and a linear transformation \(K\), the squared distance between the transformed patches is \(\|Kx_1 - Kx_2\|^2\) or alternatively \((x_1 - x_2)^TK^TK(x_1 - x_2)\). Thus a natural way to understand how distances are transformed when going from \(x_1\) to \(Kx_1\) is to look at the eigendecomposition of \(K^TK\): the \(i\)th eigenvalue of \(K^TK\) measures how much distances in the direction of the \(i\)th eigenvector are increased or decreased by the transformation. The eigenvectors of \(K^TK\) are simply the principal components of the filters, and if we assume translation invariance of the filters, they will have the same principal components as those of natural image patches: namely sines and cosines of different spatial frequencies [1]. Thus the transformation of similarities is mostly driven by the eigenvalues of \(K^TK\) and we focus on these to define the consistency of learned filters.

Denote \(p_1, ..., p_k\) the PCA components computed from the training images’ patches and \(K\) the weights of the first layer of some model trained on these images (where each row of \(K\), \(K^T_j\) is a filter). We define the energy w.r.t each component \(p_i\):

\[
e_i = \|Kp_i\| = \sqrt{\sum_j (K^T_j p_i)^2}
\]

The energy profile of a set of filters is simply the vector \(e = (e_1..e_k)\) and we measure consistency of two different sets of filters by measuring the correlation coefficient between their energy profiles.

Note that this consistency measure is invariant to a rescaling of the filters, to a permutation of the filters and to any orthogonal transformation of the filters. This way of comparing linear representation is equivalent to considering the set of filters as a filter bank and measuring the sensitivity of the filter bank to different spatial frequencies.

2.1 Different Models Learn Similar Representations in First Layer

We compared the energy profiles of various pretrained models downloaded from different online sources, on different datasets with similar patch distributions (CIFAR10, CIFAR100 and ImageNet). In order to compute the energy profiles we used PCA vectors computed on all overlapping patches of the CIFAR10 training set (with patch dimension of 3x3). In the case of models using larger kernel sizes (7x7), the PCA vectors were calculated from randomly sampled patches from the ILSVRC 2012 dataset.

As can be seen in Fig. 2, different pretrained models learn highly similar energy profiles, and these are far from their random initialization. Factors that vary between models can be depth and width, architecture of internal building blocks, and even exact set of images and labels as portrayed in Fig. 2c. To further test this phenomenon, Fig. 3 displays VGG11 trained ourselves, varying random seeds and first layer widths. Consistency between the varying factors, as well as dissimilarity from the initialization are again observed over many repetitions. As shown in Table 1, the correlation coefficient between the energy profiles is typically very high, even when different architectures are used. Additional experiments with pretrained networks, as well as training that we performed ourselves and more correlation coefficients are given in the supplementary material.

2.2 Learned Representations in the First Layer are Independent of Labels

To further examine the factors affecting the learned representations, we trained VGG11 and ResNet18 on CIFAR10, CIFAR100 and on a Faces dataset.\(^2\) We repeated training for different first layer widths and initializations, and trained with true and random labels.

\(^2\)6000 images from CelebA divided into 10 classes based on celebrity identity.
Figure 2: Pretrained models from different sources exhibit similar energy profiles when trained on same datasets (Figs. 2a and 2b), or for same architecture trained on different datasets (Fig. 2c). Fig. 2b shows an example of how these profiles differ from that of a random initialization. Full correlation figures can be found in the supplementary.

Figure 3: VGG trained on CIFAR10 with different random seeds, first layer widths and at different steps of training. While Fig. 3a and Fig. 3c show high consistency between VGG trained with different seeds and widths, Fig. 3b shows that these final conditions are far from initialization.

As can be seen in Fig. 4, while energy profiles change between the datasets (as do the patch distributions) the models learns consistent representations independent of labels. The effect can be quantified by measuring the correlation coefficients (Table 2): training with random labels and training with the true labels consistently gives a correlation coefficient over 0.90. This consistency between true labels and random labels also holds when we considered the induced distance metric similarity as in Fig. 1c.

3 Analysis

As we have seen, the energy profile of filters in the first layer of deep CNNs is remarkably consistent across initializations, architectures and labels. To explain this phenomenon, we start with a simple theorem that provides some intuition.
Table 1: Correlation between energy profiles of VGG11, trained with different random seeds (initializations), first layer widths, over various datasets, and compared with ResNet18. It is clear that independent of architecture, seed and loss, the models trained are highly correlated while different from their random initializations.

| DATASET | RANDOM SEED | TRAINED VS RANDOM INIT. | FIRST LAYER WIDTH | VGG11 VS RESNET18 |
|---------|-------------|-------------------------|-------------------|------------------|
| CIFAR10 | 0.99 ± 0.004 | -0.13 ± 0.18            | 0.98 ± 0.008      | 0.87 ± 0.04      |
| CIFAR100| 0.97 ± 0.01  | -0.04 ± 0.04            | 0.98 ± 0.01       | 0.80 ± 0.02      |
| FACES   | 0.99 ± 0.004 | -0.18 ± 0.13            | 0.98 ± 0.006      | 0.92 ± 0.02      |

Figure 4: CNNs trained on a Faces dataset, CIFAR10 and CIFAR100 tasks exhibit similar energy patterns when trained with true and random labels, when using both VGG and ResNet architectures. These are also highly correlated and differ from initialization (see Table 2).

Theorem 3.1 (First Layer in CNNs). Let \( \{(X_i,y_i)\}_{i=1}^N \) be a dataset of images and labels, let \( P_i = \{p_{i1} \ldots p_{ik}\} \) be the patches of the \( i \)’th image. Consider a CNN of any architecture whose first layer is a convolutional layer with filters \( f_1 \ldots f_M \). If the filters are initialized sufficiently close to zero, then at any step of training with GD and loss \( L \), any filter \( f_j \) (for \( j \in [M] \)) is equal to a weighted average of patches from the input:

\[
f_j = \sum_{i \in [N]} \sum_{p \in P_i} \delta_p \cdot p
\]  

(2)

Table 2: Correlation between energy profiles of VGG11 trained with true and random labels for different datasets as depicted in Fig. 4. We show the average correlation (over multiple random initializations and first layer widths) and the standard deviations. It is clear that the first layer is highly correlated when training with true and random labels.

| DATASET    | VGG (TRUE) vs INIT. | VGG (RANDOM) vs INIT. | VGG (RANDOM) vs VGG (TRUE) |
|------------|---------------------|-----------------------|---------------------------|
| CIFAR10    | -0.13 ± 0.17        | 0.03 ± 0.22           | 0.90 ± 0.02               |
| CIFAR100   | -0.044 ± 0.04       | 0.14 ± 0.13           | 0.91 ± 0.01               |
| FACES      | -0.18 ± 0.13        | 0.085 ± 0.07          | 0.96 ± 0.03               |
Theorem 3.2. Let $K$ be a matrix whose $i$’th row is the average image patch of the $i$’th image and $y$ is a vector with the labels of all images, and let $\bar{K} = KU$ be the same matrix in the PC basis (with $U$ being the PCA eigenvector matrix). The squared energy profile of weights of linear CNN, initialized with random weights sampled zero mean and covariance $\sigma^2 I$ and trained with GD, is equal to the following:

$$e_i := \frac{1}{M} \sum_{j=1}^{M} \langle f_j, p_i \rangle^2 = \bar{w}_i^2 + \sigma^2$$

where $\bar{w} = (\bar{K}^T \bar{K} + \Lambda)^{-1} \bar{K}^T y$ is the solution to a regularized regression problem in the PC basis, that regresses the average patch in an image with its label, with $\Lambda = \Lambda(K^T K, t, \eta)$ a matrix depending on the eigenvalues of $K^T K$, the iteration of GD and the step size.

Theorem 3.3 (Effect of Labels). Let $W_{\text{True}}^t$ be the weights of the first layer of a linear CNN with a single hidden layer and any width, trained for $t$ steps on a binary classification task with MSE loss and gradient descent, and let $W_{\text{Random}}$ be the weights of the first layer of the same CNN trained with random labels drawn from a Bernoulli distribution. If the average patch of both classes is identical, and the dataset is balanced between them, then at any training iteration:

$$\mathbb{E}_{y \sim \text{Bernoulli}(\frac{1}{2})} [W_{\text{Random}}^t] = W_{\text{True}}^t$$

In other words, if the average patch is the same in the two classes, then CNNs trained with random labels will learn the same filters as CNNs with true labels. Fig. 5 shows that the assumption on the average patch is quite reasonable. We computed the average patch in 10 different CIFAR categories and measured the energy profile of these average patches. The correlation coefficient between average patches corresponding to different classes is consistently above 0.9. This is true even for two classes that are visually very different (e.g. Frog and Boat).

Theorem A.5 and figure 5 partially explains the fact that real world CNNS trained with random labels learn the same energy profile as true labels (figure 3). Apparently the average patches in the
different classes are similar enough that the patch statistics overcome the influence of labels. To see whether our analytical results can quantitatively predict the label dependence of learned filters in real world CNNs, we created synthetic datasets where we systematically varied the difference in the average patch between the two classes. Specifically, we took two CIFAR categories ("dog", "frog") and modified the patches in one of the categories so that the average patches will differ by a specified $\epsilon$ in a specified direction in PCA space. This gave us 28 datasets and for each dataset we measured the predicted correlation between the energy profiles with random labels and true labels according to equation 14 and the actual correlation between the energy profiles when we train VGG on the two classes. Figure 6 shows that the prediction is excellent (correlation coefficient above 0.98).

### 4 Linear CNN and non linear CNNs

We have seen that real world CNNs learn an energy profile that is independent of architecture, initial conditions and labels, and we have also shown that some of these behaviors can be understood by analyzing the energy profile a linear CNN with a single layer. This does not, however, imply that the energy profile predicted by our analytical formula for linear, single-layer CNNs matches the energy profile of real world CNNs. As can be seen in table 3, the correlation between the energy profiles of pretrained CNNs and the best fit of the analytical formula is better than the fit with the random initialization but far from perfect. Since the analytical formula depends on various hyperparameters (learning rate, number of iterations etc.) we searched numerically for the best hyperparameters that would give the best fit. Even with this search over hyperparameters the correlations are on average around 0.5. How can we explain the poor fit?
Table 3: Correlation between the analytic formula described in Theorem A.6 and different pretrained models downloaded from the web. This simplified linear formula captures some of the pattern learned by the highly non-linear models with $\sim 10^3$ times more parameters in the first layer. Although this is far from perfect, the formula still fails in comparison to capture random initializations.

| DATASET | RESNET44    | RESNET56    | VGG13    | VGG19    | RANDOM INITIALIZATION |
|---------|-------------|-------------|----------|----------|------------------------|
| CIFAR10 | 0.474 ± 0.022 | 0.459 ± 0.001 | 0.421 ± 0.008 | 0.42 ± 0.019 | 0.23 ± 0.074 |
| CIFAR100| 0.432 ± 0.008 | 0.568 ± 0.039 | 0.49 ± 0.028 | 0.49 ± 0.01 | 0.29 ± 0.094 |

One possibility is the fact that the analytical formula is based on MSE loss while the pretrained networks were trained using cross-entropy. As we show in the supplementary material, the choice of loss actually has very little influence on the energy profile. We trained VGG with MSE and cross-entropy on the various datasets and calculated the correlation coefficient of the first layers’ energy profiles. Due to optimization challenges when training with MSE, we subtracted the initialization before calculating the correlation between the two energy profiles. After doing so, correlations can reach up to 0.98 between the models trained with the two losses and are visually similar.

Another possibility is the fact that real-world CNNs are highly nonlinear. Surprisingly, we see that these nonlinearities also have very little influence on the energy profiles. Figure 7 shows that purely linear VGG (no ReLU, max-pool replaced by average pool) have very similar energy profiles to the original, nonlinear VGGs after we account for optimization differences (we choose the number of iterations and initialization scheme for the linear network to best fit the nonlinear VGG). A promising direction for future work is to develop an analytical formula for the filters in the first layer of a linear VGG.

![Figure 7: Fitting Linear VGG for different datasets (correlation coefficient between linear VGG and VGG in parenthesis). We can see that the first layer of a linear VGG provides a highly correlated prediction of the first layer of a VGG, when finding the suitable epoch for which to stop the linear VGG’s training. We hypothesise this is due to numerical differences when training with ReLU.](a) CIFAR10 (0.81) (b) Faces (0.9)

5 Related Work

The fact that different CNNs tend to learn similar filters in the first layer has been reported previously (e.g. [26, 23, 18, 2]), and follows from a line of work of visualizing representations in deep CNNs [27, 9]. Our work extends this finding by showing that the overall representation in the first layer is not only qualitatively but also is quantitatively similar - different CNNs not only learn to recognize spatial frequencies in their first layer but also the same distribution of frequencies.
The work most closely related to our own is the recent work of Maennel et al. [19] which found that CNNs trained with random labels learn a representation whose singular vectors align with the PCAs of the input patches and suggested to investigate the "transfer function" that maps the singular values. This "transfer function" is the same as the energy profile in our work, but note that Maennel et al.'s proposition 1 merely states that when training with random labels, a transfer function will exist at any iteration (including the random, initial state). Our theorem, in contrast, gives an explicit formula for the transfer function in a linear CNN with true or random labels and we use this formula to explain why the transfer function is invariant to architecture, initialization and labels.

While other works previously tackled the problem of comparing representations in CNNs [17, 14, 20, 4], much of their focus was on the comparison methods themselves and the difference of representation in different depths of the same models. Despite focusing on only the first layer, in this paper, we attempted to conduct a comprehensive study on a wide range of architectures trained on benchmark datasets, and on the difference in training with true and random labels.

Results we have shown on training with random labels are reminiscent of works in the field of semi-supervised training. Here, the goal is to find auto-generated "pretext tasks" such that training on them lead to the model learning good representations which are useful for downstream tasks [22, 16, 5]. This is often done with auto-generated labels such as predicting image rotations [7]. While [19] reported on the benefits of pretraining with random labels, our work expands on this aspect by showing the similarity to true label training, therefore contributing a possible explanation to the reported benefits.

Our analysis method follows closely on the methods used by [15, 11] which analyze the dynamics of weights in a fully connected network during learning with gradient descent. We use a similar technique but our focus is on the first layer of a CNN.

Many recent works use linear networks to gain insight into the behavior of nonlinear networks [11, 8, 10]. Our work differs in that we are interested in the representation learned in the first layer, whereas most works focus on the classifier that is learned by the linear network. Furthermore, we theoretically and empirically study the affect of training with true and random labels, a comparison which to the best of our knowledge has not been done this comprehensively.

6 Discussion

This paper was based on our empirical observation that deep CNNs learn a highly consistent representation in their first layer. We introduced a quantitative measure of similarity between learned filters and used our measure to demonstrate a remarkable degree of similarity between the learned filters regardless of initialization and architecture. Perhaps more surprisingly, we also demonstrated very high similarity of the learned filters when trained with the true labels and random labels.

In order to understand this empirical observation we analyzed the sensitivity of filters in the first layer of a linear CNN and showed that under a reasonable assumption the energy profile will be the same with true and random labels. This allowed us to derive an analytical prediction for how much influence the labels will have on the energy profile and we saw that this predicts the behavior of nonlinear VGGs quite well.

As we emphasized repeatedly in the paper, real CNNs are very different from single layer, linear CNNs. While the first layer of these simplified CNNs exhibit some of the same qualitative behavior of the first layer of nonlinear CNNs, the quantitative fit is not perfect. We are optimistic that the tools we used to analyze shallow, linear CNNs can be extended to the deep case and to linear VGGs which we have shown behave similarly to realistic models.

Additionally, the surprising results on the similarity in representations in the first layer could prove a motivation for various applications. These can range from using random label training as a pretext task in a semi-supervised scheme to finding good representations based on patch statistics, and applying these in downstream tasks.
7 Acknowledgements

We would like to thank the Israeli Ministry of Science and Technology and the Gatsby Foundation for their support in funding this research.
References

[1] Jarmo Hurri Aapo Hyvärinen and Patrick O. Hoyer, editors. Natural Image Statistics. Springer-Verlag London, London, UK, 2009.

[2] Andrey Alekseev and Anatoly Bobe. Gabornet: Gabor filters with learnable parameters in deep convolutional neural network. In 2019 International Conference on Engineering and Telecommunication (EnT), pages 1–4, 2019. doi: 10.1109/EnT47717.2019.9030571.

[3] Dan Amir and Yair Weiss. Understanding and simplifying perceptual distances. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 12226–12235, June 2021.

[4] Adrián Csiszárik, Péter Körösi-Szabó, Ákos K. Matszangosz, Gergely Papp, and Dániel Varga. Similarity and matching of neural network representations, 2021.

[5] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. CoRR, abs/1505.05192, 2015. URL http://arxiv.org/abs/1505.05192.

[6] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. CoRR, abs/1508.06576, 2015. URL http://arxiv.org/abs/1508.06576.

[7] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. CoRR, abs/1803.07728, 2018. URL http://arxiv.org/abs/1803.07728.

[8] Gauthier Gidel, Francis R. Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in deep linear neural networks. CoRR, abs/1904.13262, 2019. URL http://arxiv.org/abs/1904.13262.

[9] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR, abs/1311.2524, 2013. URL http://arxiv.org/abs/1311.2524.

[10] Daniel Gissin, Shai Shalev-Shwartz, and Amit Daniely. The implicit bias of depth: How incremental learning drives generalization. CoRR, abs/1909.12051, 2019. URL http://arxiv.org/abs/1909.12051.

[11] Guy Hacohen and Daphna Weinshall. Convergence analysis of over-parameterized deep linear networks, and the principal components bias. CoRR, abs/2105.05553, 2021. URL https://arxiv.org/abs/2105.05553.

[12] Hui Ji, Zuowei Shen, and Yufei Zhao. Directional frames for image recovery: Multi-scale discrete gabor frames. Journal of Fourier Analysis and Applications, 23:729–757, 2017.

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. CoRR, abs/1603.08155, 2016. URL http://arxiv.org/abs/1603.08155.

[14] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey E. Hinton. Similarity of neural network representations revisited. CoRR, abs/1905.00414, 2019. URL http://arxiv.org/abs/1905.00414.

[15] Yann LeCun, Idos Kanter, and Sara Solla. Second order properties of error surfaces: Learning time and generalization. In R. P. Lippmann, J. Moody, and D. Touretzky, editors, Advances in Neural Information Processing Systems, volume 3. Morgan-Kaufmann, 1991. URL https://proceedings.neurips.cc/paper/1990/file/758874998f5bd0c393da094e1967a72b-Paper.pdf.

[16] Yazhe Li, Roman Pogodin, Danica J. Sutherland, and Arthur Gretton. Self-supervised learning with kernel dependence maximization, 2021.

[17] Yixuan Li, Jason Yosinski, Jeff Clune, Hod Lipson, and John Hopcroft. Convergent learning: Do different neural networks learn the same representations?, 2016.
[18] Shangzhen Luan, Baochang Zhang, Chen Chen, Xianbin Cao, Jungong Han, and Jianzhuang Liu. Gabor convolutional networks. CoRR, abs/1705.01450, 2017. URL http://arxiv.org/abs/1705.01450.

[19] Hartmut Maennel, Ibrahim Alabdulmohsin, Ilya Tolstikhin, Robert J. N. Ballock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? 2020. URL https://arxiv.org/abs/2006.10455.

[20] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. CoRR, abs/2010.15327, 2020. URL https://arxiv.org/abs/2010.15327.

[21] Tomaso Poggio, Kenji Kawaguchi, Qianli Liao, Brando Miranda, Lorenzo Rosasco, Xavier Boix, Jack Hidary, and Hrushikesh Mhaskar. Theory of deep learning iii: explaining the non-overfitting puzzle, 2017.

[22] Attaullah Sahito, Eibe Frank, and Bernhard Pfahringer. Better self-training for image classification through self-supervision. CoRR, abs/2109.00778, 2021. URL https://arxiv.org/abs/2109.00778.

[23] Syed Shakib Sarwar, Priyadarshini Panda, and Kaushik Roy. Gabor filter assisted energy efficient fast learning convolutional neural networks. CoRR, abs/1705.04748, 2017. URL http://arxiv.org/abs/1705.04748.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1409.1556.

[25] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. J. Mach. Learn. Res., 19:70:1–70:57, 2018. URL http://jmlr.org/papers/v19/18-188.html.

[26] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? CoRR, abs/1411.1792, 2014. URL http://arxiv.org/abs/1411.1792.

[27] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013. URL http://arxiv.org/abs/1311.2901.
A Proofs of Theorems on Convolutional Networks

A.1 Non-Linear Convolutional Networks

**Theorem A.1** (First Layer in CNNs). Let \( \{(X_i, y_i)\}_{i=1}^N \) be a dataset of images and labels, let \( P_i = \{p_i^1, \ldots, p_i^K\} \) be the patches of the \( i \)'th image. Consider a CNN of any architecture whose first layer is a convolutional layer with filters \( f_1, \ldots, f_M \). If the filters are initialized sufficiently close to zero, then at any step of training with GD and loss \( L \), any filter \( f_j \) (for \( j \in [M] \)) is equal to a weighted average of patches from the input:

\[
\bar{f}_j = \sum_{i \in [N]} \sum_{p \in P_i} \delta_p \cdot p
\]

**Proof.** Denote \( P_i \) as the matrix whose \( k \)'th row is the patch \( p_i^k \) flattened. With this notation, the convolution operation of \( f_j \) with \( X_i \) can be written as a matrix multiplication:

\[
X_i \ast f_j = P_i f_j
\]

Denote \( g_0 \) all layers except the first. Therefore by the chain rule, the gradient w.r.t. to a filter is:

\[
\sum_{i \in [N]} \nabla f_j L(g_0(X_i \ast f_j)) = \sum_{i \in [N]} \nabla f_j L(g_0(P_i f_j)) = \sum_{i \in [N]} P_i^T \nabla g_0(P_i f_j) L(g_0(P_i f_j))
\]

meaning that the derivative is some linear combination of the columns of \( P_i^T \) which are the \( i \)'th image’s patches. Summing over \( i \) results in a sum over all the training images’ patches. Since the filter \( f_j \) is initialized sufficiently close to zero, at any iteration \( t \) and with any step size \( \eta \):

\[
f_j^t = f_j^0 + \eta \sum_{s=1}^{t-1} \sum_{i \in [N]} \nabla f_j L(g_0(X_i \ast f_j^s)) \approx \sum_{s=1}^{t-1} \sum_{i \in [N]} P_i^T \nabla g_0(P_i f_j^s) L(g_0(P_i f_j^s))
\]

meaning that at any iteration \( f_j \) is still a linear combination of the input image patches. \( \square \)

A.2 Linear Convolutional Networks

We first begin with a basic claim on the model composed of a hidden convolutional layer and followed by a global average pool.

**Lemma A.2.** A linear CNN of depth 1 (followed by a global average pool) trained with MSE loss, is equivalent to linear regression on the average image patch.

**Proof.** Let \( \{X_i\}_{i=1}^N \) with \( X_i \in \mathbb{R}^{c \times w \times h} \) be the set of training images, \( \{y_i\}_{i=1}^N \) their binary labels (\( y_i \in \{0, 1\} \)) and the weights of the first layer be \( W \in \mathbb{R}^{k \times c \times d \times d} \) - \( k \) filters of dimension \( d \times d \). Denote the output dimensions of the convolution as \( w', h' \), then:

\[
\mathcal{L}(W; \{(X_i, y_i)\}_{i=1}^N) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} \left\| \frac{1}{k} \sum_{k, w', h'} X_i \ast w' \right\| _2^2 - y_i \right\| ^2
\]

Summing over the output dimensions is equivalent to summing over a dot product of the patches with a single filter, therefore denoting \( K_i \in \mathbb{R}^{w' \times c \times d^2} \) as the patch matrix of the \( i \)'th image and \( \bar{W} \in \mathbb{R}^{c \times d^2 \times k} \) the reshaped weights matrix:

\[
\mathcal{L}(W; \{(X_i, y_i)\}_{i=1}^N) = \frac{1}{N} \sum_{i=1}^N \frac{1}{2} \left\| \left( \frac{1}{w'} \right) K_i \bar{W} \left( \frac{1}{k} \right) - y_i \right\| ^2
\]

And noting that \( \left( \frac{1}{w'} \right) K_i \) is the average patch of the \( i \)'th image and \( \bar{W} \left( \frac{1}{k} \right) \) the average filter concludes the proof. \( \square \)
Another lemma we’ll use further on claims that during training of the linear CNN model, only the average filter changes while the filter covariance remains as during initialization. Therefore proofs from one filter to multiple filters are easily extendable.

**Lemma A.3.** In a linear CNN of depth 1 followed by a global average pool, of any width, trained with GD and MSE loss, the average filter changes during iterations while the covariance of filters remains as during initialization.

**Proof.** Following the notation of Lemma A.2, denote $K \in \mathbb{R}^{N \times c \times d^2}$ as the average image patch matrix - the image whose $i$’th row is the average patch of the image $X_i$, and the network consists of filters $w_1, \ldots, w_m$, therefore trained with the following loss:

$$
\mathcal{L}(w_1 \ldots w_m, K, y) = \left\| \frac{1}{m} \sum_{i=1}^{m} K w_i - y \right\|^2
$$

(9)

$$
= \left\| K \left( \frac{1}{m} \sum_{i=1}^{m} w_i \right) - y \right\|^2 = \left\| Kw - y \right\|^2
$$

(10)

Where $\bar{w}$ is the average filter. The dynamics of a single filter in this layer:

$$
\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{2m} K^T \left( K \left( \frac{1}{m} \sum_{i=1}^{m} w_i \right) - y \right) = \frac{1}{2m} K^T (K \bar{w} - y)
$$

(11)

Meaning that the gradients w.r.t to all filters are equal and depend only on the average filter at the current iteration.

By recursion we can see that the change in the average filter is as follows, for learning rate $\eta$:

$$
\bar{w}^t = \frac{1}{m} \sum_{i=1}^{m} (w_i^{t-1} - \eta \nabla \mathcal{L}^{t-1}(w_i)) = \frac{1}{m} \sum_{i=1}^{m} (w_i^{t-1} - \eta \nabla \mathcal{L}^{t-1}(\bar{w}))
$$

(12)

$$
= \left( \frac{1}{m} \sum_{i=1}^{m} w_i^{t-1} \right) - \eta \nabla \mathcal{L}^{t-1}(\bar{w}) = \bar{w}^{t-1} - \eta \nabla \mathcal{L}^{t-1}(\bar{w})
$$

(13)

Concluding that the gradients for all filters are equal, and depend only on the average filter.

**Theorem A.4.** Let $K$ be a matrix whose $i$’th row is the average image patch of the $i$’th image and $y$ is a vector with the labels of all images, and let $\bar{K} = KU$ be the same matrix in the PC basis (with $U$ being the PCA eigenvector matrix). The squared energy profile of weights of linear CNN, initialized with random weights sampled zero mean and covariance $\sigma^2 I$ and trained with GD, is equal to the following:

$$
e_i := \frac{1}{M} \sum_{j=1}^{M} (f_j, p_i)^2 = \bar{w}^2_i + \sigma^2
$$

(14)

where $\bar{w} = (\bar{K}^T \bar{K} + \Lambda)^{-1} \bar{K}^T y$ is the solution to a regularized regression problem in the PC basis, that regresses the average patch in an image with its label, with $\Lambda = \Lambda(K^T K, t, \eta)$ a matrix depending on the eigenvalues of $K^T K$, the iteration of GD and the step size.

**Proof.** It follows from Lemma A.3 that during training all filters change by the average filter. We’ll show that a single filter (at iteration $t$ of GD) corresponds to a solution to ridge regression with some matrix $\Lambda = \Lambda(t, \eta, K^T K)$ with $\eta$ being the step size. Opening the recursion of GD updates, and assuming $w$ is initialized at $w = 0$:

$$
w_t = w_{t-1} - \eta K^T (Kw_{t-1} - y) = w_{t-1} - \eta K^T K w_{t-1} + \eta K^T y
$$

$$
w_t = \eta \sum_{j=0}^{t-1} (I - \eta K^T K)^j K^T y
$$

(15)

In this coordinate system, $\bar{K}^T \bar{K}$ is a diagonal matrix with the empirical variances $\hat{\sigma}^2$ on the diagonal if it is centered. If the matrix isn’t centered, then $\bar{K}^T \bar{K} = \Sigma + \bar{\mu} \bar{\mu}^T$ where $\Sigma$ is a diagonal matrix
with the empirical variances on the diagonal and \( \bar{\mu}_i \) is the empirical mean estimating \( \mathbb{E}_x [\langle x, p_i \rangle] \).

This is because in PCA coordinates, \( \tilde{K} = KU \), where \( U \) contains the eigenvectors as columns. Since \( \tilde{K} \) isn’t centered, \( \tilde{K} = K_0 + 1\bar{K}^T \) with \( K_0 \) being zero meaned and \( \bar{K} \) being the average row. Therefore, \( \hat{K}^T \tilde{K} = (K_0U + 1\bar{K}^T U)^T (K_0U + 1\bar{K}^T U) = \hat{\Sigma} + \hat{\mu} \hat{\mu}^T \), where the phrase \( K_0^T (1\bar{K}^T) \) disappears since \( K_0 \) has zero mean. Therefore:

\[
\hat{w}_t = \eta \sum_{j=0}^{t-1} \left( I - \eta \hat{K}^T \hat{K} \right)^j \hat{K}^T y = \eta \sum_{j=0}^{t-1} \left( I - \eta \left( \hat{\Sigma} + \hat{\mu} \hat{\mu}^T \right) \right)^j \hat{K}^T y \quad (16)
\]

Notice that \( \left( I - \eta \left( \hat{\Sigma} + \hat{\mu} \hat{\mu}^T \right) \right)^j \) can be decomposed in the following manner using the binomial theorem:

\[
\left( I - \eta \left( \hat{\Sigma} + \hat{\mu} \hat{\mu}^T \right) \right)^j = \left( I - \eta \hat{\Sigma} \right)^j + \sum_{k=1}^{j} \binom{j}{k} (-\eta)^k \| \hat{\mu} \|^2 (k-1) \left( I - \eta \hat{\Sigma} \right)^{j-k} \hat{\mu} \hat{\mu}^T \quad (17)
\]

Putting it back into Eq. (16):

\[
\hat{w}_t = \eta \sum_{j=0}^{t-1} \left( I - \eta \left( \hat{\Sigma} + \hat{\mu} \hat{\mu}^T \right) \right)^j \hat{K}^T y
\]

\[
= \eta \left( \sum_{j=0}^{t-1} \left( I - \eta \hat{\Sigma} \right)^j + \sum_{k=1}^{j} \binom{j}{k} (-\eta)^k \| \hat{\mu} \|^2 (k-1) \left( I - \eta \hat{\Sigma} \right)^{j-k} \right) \hat{K}^T y \quad (18)
\]

Looking at the \( i \)’th coordinate, with \( \lambda_i \) being the \( i \)’th eigenvalue on the diagonal of \( \hat{\Sigma} \):

\[
\hat{w}_t(i) = \left( \hat{K}^T y \right)(i) \sum_{j=0}^{t-1} (1 - \eta \lambda_i)^j
\]

\[
+ \left( \hat{\mu} \hat{\mu}^T \hat{K}^T y \right)(i) \sum_{j=1}^{t-1} \frac{1}{\| \hat{\mu} \|^2} \sum_{k=1}^{j} \binom{j}{k} (-\eta \| \hat{\mu} \|^2)^k (1 - \eta \lambda_i)^{j-k} \quad (19)
\]

After some algebra:

\[
\hat{w}_t(i) = \left( \frac{1 - (1 - \eta \lambda_i + \| \hat{\mu} \|^2)^t}{\| \hat{\mu} \|^2 (\lambda_i + \| \hat{\mu} \|^2)} - \frac{1 - (1 - \eta \lambda_i)^t}{\lambda_i} \right) \left( \hat{K}^T y \right)(i)
\]

\[
+ \left( \frac{1 - (1 - \eta \lambda_i)^t}{\lambda_i} \right) \left( \hat{K}^T y \right)(i) \quad (20)
\]

And in matrix notation, define the diagonal matrix \( A \) with \( \frac{1 - (1 - \eta \lambda_i)^t}{\lambda_i} \) as the \( i \)’th element on the diagonal, and the diagonal matrix \( B \) with \( \frac{1 - (1 - \eta \lambda_i + \| \hat{\mu} \|^2)^t}{\| \hat{\mu} \|^2 (\lambda_i + \| \hat{\mu} \|^2)} \) the \( i \)’th element on the diagonal and we get that:

\[
\hat{w}_t = (B - A) \hat{\mu} \hat{\mu}^T \hat{K}^T y + A \hat{K}^T y \quad (21)
\]

Solving the following:

\[
\hat{w}_t = \left( \hat{K}^T \hat{K} + \Lambda \right)^{-1} \hat{K}^T y = \left( \hat{\Sigma} + \hat{\mu} \hat{\mu}^T + \Lambda \right)^{-1} \hat{K}^T y \quad (22)
\]

we get that:

\[
\Lambda = (B + (A - B) \hat{\mu} \hat{\mu}^T)^{-1} - \hat{\Sigma} - \hat{\mu} \hat{\mu}^T \quad (23)
\]

and we got a definition for the regularization matrix.

Since the filter covariance stays constant throughout training due to Lemma A.3, treating the filters as a random variable initialized with covariance \( \sigma^2 I \) (in PCA basis) means that their empirical second
moment is equal to the sum of the squared mean and variance. Therefore denoting the filters in PCA basis as $\tilde{f}_j$, we get that in the $i$th coordinate:

$$
\frac{1}{M} \sum_{j=1}^{M} \langle f_j, p_i \rangle^2 = \left( \frac{1}{M} \sum_{j=1}^{M} \langle f_j, p_i \rangle \right)^2 + \frac{1}{M} \sum_{j=1}^{M} \left( \langle f_j, p_i \rangle - \left( \frac{1}{M} \sum_{j=1}^{M} \langle f_j, p_i \rangle \right) \right)^2 = \tilde{w}^2(i) + \sigma^2
$$

(24)

**Theorem A.5** (Effect of Labels). Let $W^t_{\text{True}}$ be the weights of the first layer of a linear CNN with a single hidden layer and any width, trained for $t$ steps on a binary classification task with MSE loss and gradient descent, and let $W_{\text{Random}}$ be the weights of the first layer of the same CNN trained with random labels drawn from a Bernoulli distribution. If the average patch of both classes is identical, and the dataset is balanced between them, then at any training iteration:

$$
\mathbb{E}_{y \sim \text{Bernoulli}(\frac{1}{2})} [W^t_{\text{Random}}] = W^t_{\text{True}}
$$

(25)

**Proof.** Let $K \in \mathbb{R}^{N \times c \times d^2}$ be the average image patch matrix and $y \in \{0, 1\}^N$ the image labels. From Lemma A.2, training a linear CNN with 1 layer followed by a global average pool is equivalent to solving the following linear regression problem for weights matrix $W \in \mathbb{R}^{c \times d^2 \times 1}$:

$$
\mathcal{L}(W; K, y) = \frac{1}{N} \|KW - y\|^2
$$

Using gradient descent with learning rate $\eta$, the update rule for $W$ is:

$$
W_t = W_{t-1} - \frac{\eta}{N} (K^T(KW_{t-1}y)) = \left( I - \frac{\eta}{N} K^T K \right) W_{t-1} + \frac{\eta}{N} K^T y
$$

(26)

Notice that in expectation, $\mathbb{E}_{y \sim \text{Bernoulli}(\frac{1}{2})} [y] = \frac{1}{2} 1$, therefore $\mathbb{E}_{y \sim \text{Bernoulli}(\frac{1}{2})} [K^T y]$ is (half) the sum of all average image patches. From our assumption, the average image is equal between classes. Denote this average patch as $z$, and since $K$ is the average patch matrix $z = \frac{2}{N} K y$. Combining this observation with the above:

$$
\mathbb{E}_{y \sim \text{Bernoulli}(\frac{1}{2})} \left[ \frac{\eta}{N} K^T y \right] = \frac{\eta}{N} \frac{1}{2} K^T 1 = \frac{\eta}{2N} K^T y
$$

(27)

And that concludes the proof. Note that we assumed that the CNN is of width 1, but using Lemma A.3 is enough for generalizing to any width. $\square$

**Theorem A.6** (Solution in PCA Basis). Let $\tilde{w} = \left( \tilde{K}^T \tilde{K} + \Lambda \right)^{-1} \tilde{K}^T y$ be as described in Theorem A.4, for $\tilde{K}$ the average image patch matrix in the PCA basis and $\Lambda = \Lambda(\tilde{K}^T \tilde{K}, t, \eta)$. Denote $\hat{\mu}$ as the empirical mean projection onto the PCA basis and $\hat{\Sigma}$ as the the uncentered data covariance in PCA basis. If the labels are drawn randomly from a Bernoulli distribution, then in expectation, $\tilde{w}$ can be calculated at any iteration $t$ and for any step size $\eta$ with the following formula:

$$
\mathbb{E}_{y \sim \text{Bernoulli}(\frac{1}{2})} [\tilde{w}] \propto \left( I - \frac{\hat{\Sigma}^{-1} \hat{\mu} \mu^T}{1 + \mu^T \hat{\Sigma}^{-1} \hat{\Sigma}^T} \right) \hat{\Sigma}^{T-1} \hat{\mu}
$$

(28)

with $\hat{\Sigma} = \hat{\Sigma} + \Lambda$.

**Proof.** Following the notation from before, denote $K \in \mathbb{R}^{N \times c \times d^2}$ as the average patch matrix, and $\tilde{K}$ as the same matrix in the PCA basis coordinates. From Theorem A.4 $\tilde{K}^T \tilde{K} = \hat{\Sigma} + \hat{\mu} \hat{\mu}^T$. Solving the linear ridge regression problem in this coordinate system as described in Theorem A.4:

$$
L(w; \tilde{K}, y) = \frac{1}{2} \| \tilde{K} w - y \|^2 + \frac{1}{2} w^T \Lambda w \Rightarrow \tilde{w} = \left( \tilde{K}^T \tilde{K} + \Lambda \right)^{-1} \tilde{K}^T y
$$

(29)
In expectation over a random $y$, as described in Theorem A.5: $\mathbb{E} [y] = \frac{1}{2} I$, therefore $\mathbb{E} \left[ \tilde{K}^T y \right] = \frac{N}{2} \mu$.

As mentioned before, $\tilde{K}^T \tilde{K} = \hat{\Sigma} + \mu \mu^T$. Define $\hat{\Sigma}' = \hat{\Sigma} + \Lambda$ a matrix summing the PCA variances and the regularization coefficients. Now using Woodbury matrix identity:

$$\left( \hat{\Sigma}' + \mu \mu^T \right)^{-1} = \hat{\Sigma}'^{-1} - \hat{\Sigma}'^{-1} \mu (I + \mu^T \hat{\Sigma}'^{-1} \mu) \mu^T \hat{\Sigma}'^{-1} = (I - \frac{\hat{\Sigma}'^{-1} \mu \mu^T}{1 + \mu^T \hat{\Sigma}'^{-1} \mu}) \hat{\Sigma}'^{-1}$$

and we get that:

$$\tilde{w} \propto (I - \frac{\hat{\Sigma}'^{-1} \mu \mu^T}{1 + \mu^T \hat{\Sigma}'^{-1} \mu}) \hat{\Sigma}'^{-1} \mu$$

\[\square\]

### B Expanded Results on Similarity Between Pretrained CNNs

#### B.1 Comparison of Pretrained CNNs on CIFAR and ImageNet

To expand on the similarity between first layers of different architectures, we present correlation plots emphasizing the difference between a random initialization and the learned weights of different networks on different datasets. Presented are figures comparing pretrained models on ImageNet (Figs. 8 and 9), CIFAR10 (Fig. 10) CIFAR100 (Fig. 11), and ResNets trained on different datasets (Fig. 12). All models were downloaded through the Pytorch Model Hub.

![Figure 8: The energy profiles of networks with different architectures and first layer with kernel size 7, trained on Imagenet, are correlated and differ much from a random initialization.](image)

Although it might seem odd that correlation on Imagenet is much higher than on the CIFAR datasets, we believe this is due to resolution - while on the CIFAR datasets correlation is calculated over an energy profile in $\mathbb{R}^{27}$, the Imagenet example contains profiles in $\mathbb{R}^{147}$, making the calculated correlation smoother and less sensitive to noise. This is demonstrated in Fig. 13 which presents the correlation between 27 components of the Imagenet profiles. When looking in higher resolution the correlation coefficients between the different models drop and are relatively equal to those between the different models on the CIFAR datasets.
Figure 9: The energy profiles of networks with different architectures and first layer with kernel size 3, trained on Imagenet, are correlated and differ much from a random initialization.

Figure 10: The energy profiles of networks with different architectures and first layer with kernel size 3, trained on CIFAR10, are correlated and differ much from a random initialization.
Figure 11: The energy profiles of networks with different architectures and first layer with kernel size 3, trained on CIFAR100, are correlated. Although it is possible to sample a random initialization that correlates with some models (ResNet) as good as others correlate with them (VGG 11), most models still differ from such a random initialization.

Figure 12: An expansion of the result shown in Figure 2: ResNets of different depths trained on different datasets have highly correlated energy profiles.
Figure 13: Correlation between different model energy profiles on Imagenet when zooming in on components 25-52. The higher correlation between the models relative to models on CIFAR is due to a higher dimension of the energy profiles.
B.2 Comparison of Linear CNN on CIFAR10

To empirically test our theorems, we trained a linear CNN of different first layer widths and depths on binary subsets of CIFAR10. Examples of the results are depicted in Fig. 14. Due to optimization challenges in training linear networks, networks weren't initialized with exactly zero mean and diagonal covariance. That being said, networks of different widths, depths and trained with different labels are still highly correlated, even without theoretical guarantee.

![Graph](image)

(a) Dog vs Frog

(b) Car vs Truck

Figure 14: Linear CNN of depth 10, with different first layer widths and trained with true and random labels on different binary datasets constructed from CIFAR10. As depicted in the graphs, even at depth larger than 1, for which we have no theoretical guarantees by Theorems A.4 and A.6, linear models have highly correlated energy profiles when trained with true and random labels and different widths.

B.3 Comparison of VGG with Different Losses

Although A.5 and all other theorems are proved on a linear network using MSE loss (as customary in theoretical works on linear networks e.g. [11, 15, 8]), in practice most CNNs for multi-class classification are trained with crossentropy loss. To test the effect on the energy profile of a real
network, we trained VGG with both crossentropy and MSE, and with true and random labels, the results are displayed 15 and correlations in 4. As can be seen in the figure, even in this case the networks’ energy profiles are highly correlated, thus supporting our hypothesis that the main difference between the formula A.6 and the pretrained networks is due to the oversimplification of the linear model, and not for example the loss used in theory vs practice.

Figure 15: Comparison between VGG11 trained with MSE loss and Cross Entropy loss on different datasets. Models are highly correlated, and learn similar components in their first layer. The phenomena is consistent even when training with random labels (Fig. 15d). Initialization was subtracted from the first layer prior to calculation of energy profile due to the challenging optimization of networks trained with MSE.
Table 4: Correlation between energy profiles of VGG11 when trained with MSE loss and Cross Entropy (CE) loss. Due to optimization challenges, we subtracted the initialization from the first layer prior to calculating the correlation, now comparing the accumulated gradients.

| Dataset      | MSE vs Cross Entropy |
|--------------|-----------------------|
| CIFAR10      | 0.96 ± 0.01           |
| CIFAR100     | 0.67 ± 0.03           |
| FACES        | 0.90 ± 0.025          |

B.4 Experimental Details

All models - linear and non linear were trained with SGD and a constant learning rate of 0.1. No preprocessing was applied to the data except when stated otherwise. All models were trained for 100 epochs, with minibatches of size 256. All results are averaged over at least 3 different random seeds. When referring to models “trained with random labels”, we trained models until they overfit the training data. While ResNet and VGG can reach >99% accuracy on CIFAR10, experiments on Linear models where kept only if they achieved at least 60% train accuracy (as these models can’t reach the performance of non-linear models).

All pretrained models were downloaded from the Pytorch Model Hub.

C Filter Visualization

As noted, filters of first layers of CNNs trained on natural images are visually similar to Gabor filters Gabor filters, i.e. localized filters that are sensitive to different spatial frequencies [26, 23, 18]. In 16, it can be seen that filters learned by VGG trained on CIFAR10 with true and random labels have visually similar learned filters. Although the dimensions are $3 \times 3$, it can be seen that some of the filters correspond to spatial frequencies. Furthermore, it is clear that the PCA components are composed mainly of filters sensitive to spatial frequencies.

Figure 16: The filters of a first layer of VGGs trained with true and random on CIFAR10, and the PCA components used for analysis.