DETECTING MEMORIZATION IN RELU NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a new notion of 'non-linearity' of a network layer with respect to an input batch that is based on its proximity to a linear system, which is reflected in the non-negative rank of the activation matrix. We measure this non-linearity by applying non-negative factorization to the activation matrix. Considering batches of similar samples, we find that high non-linearity in deep layers is indicative of memorization. Furthermore, by applying our approach layer-by-layer, we find that the mechanism for memorization consists of distinct phases. We perform experiments on fully-connected and convolutional neural networks trained on several image and audio datasets. Our results demonstrate that as an indicator for memorization, our technique can be used to perform early stopping.

1 INTRODUCTION

A fundamental challenge in machine learning is balancing the bias-variance tradeoff, where overly simple learning models underfit the data (suboptimal performance on the training data) and overly complex models are expected to overfit or *memorize* the data (perfect training set performance, but suboptimal test set performance). The latter direction of this tradeoff has come into question with the observation that deep neural networks do not memorize their training data despite having sufficient capacity to do so (Zhang et al., 2016), the explanation of which is a matter of much interest.

Due to their convenient gradient properties and excellent performance in practice, rectified-linear units (ReLU) have been widely adopted and are now ubiquitous in the field of deep learning. In addition, the relative simplicity of this function ($\max(\cdot, 0)$) makes the analysis of ReLU networks more straight-forward than networks with other activation functions.

We propose a new notion of 'non-linearity' of a ReLU layer with respect to an input batch. We show that networks that networks that generalize well have deep layers that are approximately linear with respect to batches of similar inputs. In contrast, networks that memorize their training data are highly non-linear with respect to similar inputs, even in deep layers.

Our method is based on the fact that the main source of non-linearity in ReLU networks is the threshold at zero. This thresholding determines the *support* of the resulting activation matrix, which plays an important role in the analysis of non-negative matrices. As we discuss in Section 3, the *non-negative rank* of a matrix is constrained by the shape of the support, and is therefore indicative of the degree of non-linearity in a ReLU activation matrix with respect to the input.

Although computing the non-negative rank is NP-hard (Vavasis, 2009), we can restrict it with approximate *non-negative matrix factorization* (NMF) (Lee & Seung, 1999). Consequently, we propose to estimate the 'non-linearity' of a ReLU layer with respect to an input batch by performing NMF on a grid over the approximation rank k, and measuring the impact on network performance.

This procedure can be seen as measuring the robustness of a neural network to increasing compression of its activations. We therefore compare our NMF-based approach to two additional dimensionality reduction techniques, namely principal component analysis (PCA) and random ablations.

We report results for a variety of neural network architectures trained on several image and audio datasets. We show that our NMF-based approach is extremely sensitive to memorization in neural networks. We conduct a layer-by-layer analysis and our results reveal interesting details on the internal mechanism of memorization in neural networks. Finally, as an indicator for memorization, we use our proposed measure to perform early stopping.

2 RELATED WORK

The study of factors involved in the bias-variance tradeoff in learning models goes back several decades. Classical results in statistical learning consider properties of learning models such as the VC-dimension (Vapnik, 1998) and Rademacher complexity (Bartlett & Mendelson, 2002). These properties give generalization bounds in terms of the capacity model to (over)fit data. When considering the vast capacity of deep neural networks, such bounds become irrelevant and fail to explain their ability to generalize well in practice (Zhang et al., 2016; Bartlett et al., 2017).

More direct analyses have been done with respect to a specific setting of model parameters. For instance, Bartlett (1998) showed that the number of weights in a network is less important compared to their scalar value (e.g. ℓ_2-norm), and more recently Bartlett et al. (2017) presented a bound for deep neural networks based on the product of spectral norms of the network's weight matrices. Achille & Soatto (2017) showed that memorizing networks contain more information in their weights.

Methods to explain generalization have been proposed that examine a network's robustness to perturbations (Hochreiter & Schmidhuber, 1997; Chaudhari et al., 2016; Keskar et al., 2016; Neyshabur et al., 2017; Li et al., 2017). These methods propose the notion of *flatness of minima* on the loss surface, assuming that perturbing the parameters without dramatically changing performance is an indicator of the generalization of a network.

However, any reversible transformation, such as simple scaling, can arbitrarily manipulate the local flatness without affecting generalization (Dinh et al., 2017). The procedure we propose can be viewed as applying perturbations, albeit to *activations* and not parameters, and must address this concern. The perturbations we apply to activations account for magnitude, since they depend on a change of rank or non-negative rank of the activation matrix, a property which is robust to rescaling and similar reversible transformations.

In contrast to the methods described thus far, which deal exclusively with the parameters of the model, methods have been developed that account for the role of the data distribution. Liang et al. (2017) proposed to use the Fisher-Rao norm, which uses the geometry of the data distribution to weigh the contribution of different model parameters. The empirical studies of Morcos et al. (2018) and Novak et al. (2018) explore robustness to specific types of noise. The former uses Gaussian noise and masking noise injected into hidden activations, while the latter interpolates between input samples to study network behavior on and off the data manifold. In both cases, robustness to noise proved a reliable indicator for good generalization. Additionally, Arora et al. (2018) derive generalization bounds in terms of robustness to noise.

Our experimental setup is reminiscent of Morcos et al. (2018) in that both methods apply a form of compression to hidden activations and test for robustness to this type of noise. Specifically, they set random axis-aligned directions in feature space to zero which can be viewed as a crude form of dimensionality reduction, i.e., by simply removing canonical dimensions. In our experiments we refer to this method as random ablations. Our results show that robustness to NMF compression is much more correlated with low memorization/high generalization than robustness to random ablations. Arpit et al. (2017) have also studied various empirical aspect of memorization.

As a dimensionality reduction technique, NMF has gained popularity due to its producing meaningful factorizations that lend themselves to qualitative interpretation across various domains such as document clustering (Xu et al., 2003), audio source separation (Grais & Erdogan, 2011), and face recognition (Guillamet & Vitria, 2002). In the context of deep convolutional neural networks, Collins et al. (2018) applied NMF to the activations of an image classifier and showed that the result gives a decomposition into semantic parts, which benefits from the transformation invariance learned by the neural network.

3 METHOD

Consider a ReLU layer of a neural network, parameterized by a weight matrix $\boldsymbol{W} \in \mathbb{R}^{m \times q}$. For a batch of n inputs $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, we compute the layer activation matrix $\boldsymbol{A}$ as follows:

$$\boldsymbol{A} = \max\left(\boldsymbol{X}\boldsymbol{W}, 0\right) \in \mathbb{R}_{+}^{n \times q}, \tag{1}$$

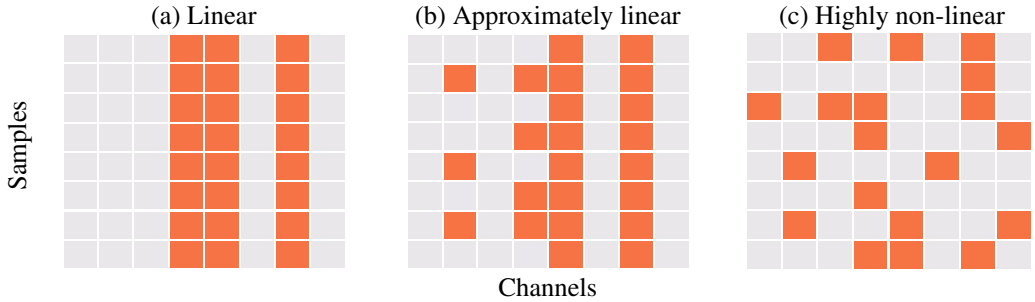

Figure 1: **The support of the activation matrix** is determined by ReLU threshold. (a) When all the rows of the support are identical, there is a sub-weight-matrix such that the layer is fully linear with respect to the input batch. (b, c) As the support becomes more complex, which we characterize by the increase in its rectangle cover number, the layer becomes more non-linear.

where $\mathbb{R}_+$ are the non-negative reals. We omit the bias term for notational convenience.

3.1 ReLU-induced support

The processing of a single input $\boldsymbol{x}$ by a ReLU network is equivalent to sampling a sub-network that is linear with respect to the sample (Wang et al., 2016). This could be accomplished by simply setting to zero the columns of each $\boldsymbol{W}$ whose dot product with the input is negative (and would thus be set to zero by ReLU), and then removing the thresholding.[1]

Extending this notion to a batch of several input samples to a ReLU layer, suppose the samples are sufficiently close to each other such that they all share the same ReLU mask $\boldsymbol{m} \in \{0,1\}^q$. In this case, we may say that *the layer is linear with respect to its input batch*. This is because, for the entire batch, instead of using ReLU, we could zero out a subset of columns and have have a linear system, i.e., $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{W}\text{diag}(\boldsymbol{m})$.

For an activation matrix $\boldsymbol{A}$ (Equation 1), we consider the support $\boldsymbol{M} = \text{supp}(\boldsymbol{A})$, which we describe as a binary 0/1 matrix where $\boldsymbol{M}_{i,j} = 1$ where $\boldsymbol{A}_{i,j} > 0$. Because $\boldsymbol{A}$ is a ReLU activation matrix, the structure of $\boldsymbol{M}$ is mainly determined by the thresholding at zero.[2] Because threshold is the main source of non-linearity in a ReLU network, the support takes on a special meaning in this case.

3.2 Rectangle cover number and non-negative rank

We want to characterize how close to being linear a layer is with respect to its input $\boldsymbol{X}$ by examining the support of the resulting activations $\boldsymbol{M}$. If all the rows of $\boldsymbol{M}$ are identical to a unique vector $\boldsymbol{m}$, we can say the layer is completely linear with respect to $\boldsymbol{X}$. In general, the 'simpler' the support $\boldsymbol{M}$, the closer to linearity the layer.

One measure that captures this idea is the *rectangle cover number* of a matrix, $rc(\boldsymbol{M})$, an important quantity in the study of communication complexity (Klauck, 2003). Also known as the *Boolean rank*, $rc(\boldsymbol{M})$ is the smallest number r for which there exist binary matrices $\boldsymbol{U}_B \in \{0,1\}^{n\times r}$, $\boldsymbol{V}_B \in \{0,1\}^{r\times q}$ such that $\boldsymbol{M} = \boldsymbol{U}_B\boldsymbol{V}_B$. As a complexity measure for ReLU activations, $rc(\boldsymbol{M}) = 1$ means the layer is linear with respect to its input, and higher values $rc(\boldsymbol{M})$ imply increasing non-linearity. This is visualized in Figure 1.

Intuitively, imagine having to fit a layer with 'ReLU switches', each of which controls a subset of weight matrix columns. In the linear case, one switch would control all the columns. In the most non-linear case we would require a switch for every column, which is also the maximal value of $rc(\boldsymbol{M})$.

[1] This is a similar intuition to that of viewing dropout as an approximation to a model ensemble, where the dropout mask is seen to sample a sub-network (Srivastava et al., 2014).

[2] The probability of an activation value being exactly zero prior to thresholding is negligible.

Because computing the rectangle cover number $rc(\boldsymbol{M})$ is complex, several approximations and bounds to it have been studied (Fiorini et al., 2013). For the support of a non-negative matrix, a well-known upper-bound is:

$$rc(\text{supp}(\boldsymbol{A})) \leq rank_+(\boldsymbol{A}), \tag{2}$$

where $rank_+(A)$ is the *non-negative rank* of $\boldsymbol{A}$ (Gillis & Glineur, 2012) that is defined as the smallest number k such that there exist non-negative matrices $\boldsymbol{U}_+ \in \mathbb{R}_+^{n\times k}$, $\boldsymbol{V}_+ \in \mathbb{R}_+^{k\times q}$ such that $A = U_+V_+$. Similar to the rectangle cover number, the non-negative rank is hard-constrained by the combinatorial arrangement of supp($\boldsymbol{A}$), but additionally accounts for the actual value in the non-zero entries of $\boldsymbol{A}$.

While computing $rank_+(\boldsymbol{A})$ is not easier than computing $rc(\text{supp}(\boldsymbol{A}))$, we can *restrict it* by performing approximate non-negative matrix factorization (NMF).

3.3 Non-negative matrix factorization

For a given non-negative rank constraint k, NMF solves for:

$$\underset{\boldsymbol{U}_+,\boldsymbol{V}_+}{\arg\min} \|\boldsymbol{A} - \boldsymbol{U}_+\boldsymbol{V}_+\|_2^2, \tag{3}$$

with $\boldsymbol{U}_+, \boldsymbol{V}_+$ as defined above. The result $\boldsymbol{U}_+\boldsymbol{V}_+ = \tilde{\boldsymbol{A}}_k \approx \boldsymbol{A}$ is the closest matrix to $\boldsymbol{A}$ under the Frobenius norm that has $rank_+$ at most k.

Consequently, we propose to estimate the 'linearity' of a ReLU layer with respect to a batch of similar inputs by performing NMF on a grid over the non-negative rank k, and measuring the impact on network performance by observing the change in the prediction (output layer) as we change k. This procedure also addresses the fact that in practice network activations tend to be noisy, whereas supp($\boldsymbol{A}$) is not robust to noise, i.e., $\boldsymbol{A}_{i,j} = \epsilon > 0 \rightarrow \boldsymbol{M}_{i,j} = 1$ even for very small ϵ.

Concretely, if we let $\boldsymbol{A}_i$ be the activation matrix at layer i, during the forward pass, we replace the feature activations of one or several layers with their rank k NMF approximations:

$$\boldsymbol{A}_{i+1} = \max\left(\tilde{\boldsymbol{A}}_k W_{i+1}, 0\right) \tag{4}$$

For convolutional networks, we first reshape the tensor of feature maps from $n \times q \times h \times w$ to $(n \cdot h \cdot w) \times q$, i.e., we flatten the batch (n) and spatial dimensions (h, w) to form a matrix with q columns, where q is the number of channels in that layer. We then inversely reshape the approximated features to continue forward propagation through the network.

3.4 Single class batches

We now characterize the input batch, with respect to which we would like to measure layer linearity. Informally, the goal of training is to cluster together input samples that have similar (or identical) output labels,while separating them from samples of other labels. In the context of classification then, we expect therefore that from a certain depth and onward, samples of the same class will have similar activations, and thus a simpler support. In other words, while a network may exploit flexible non-linear structure to separate *different classes*, we expect that with respect to a *single class*, deep layers are approximately linear.

When single-class batches are *not* approximately linear, even in deep layers, we take this as evidence of memorization, i.e., the samples are not tightly clustered. In the next section we present empirical results that support this view.

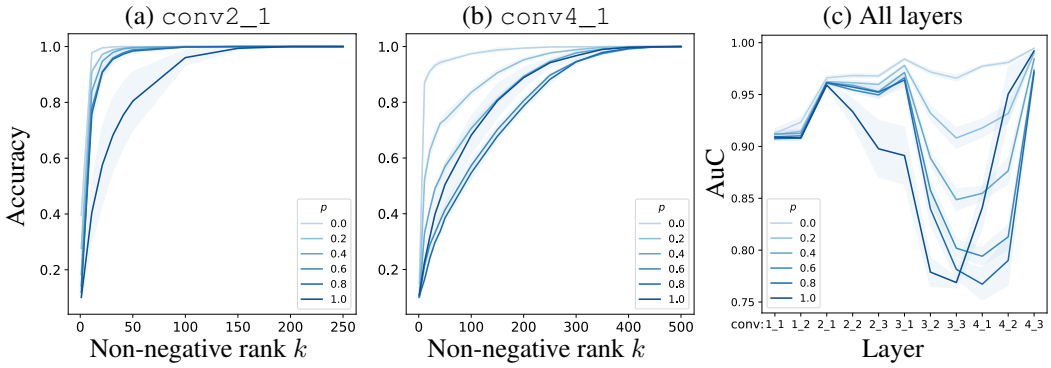

Figure 2: **Memorization mechanism** of CNNs trained on CIFAR-10, with increasing level of label randomization p (i.e., $p = 0$ is the unmodified dataset). We analyze each layer by applying NMF compression to its activation matrix with increasing rank k, while observing the impact on classification performance. In (a) and (b) we show the k vs. accuracy curves at an early layer and a deep layer, respectively. We can immediately see that in deep layers, networks with high memorization are significantly less robust to NMF compression, indicating higher degrees of non-linearity. Furthermore, networks trained on fully randomized labels ($p = 1$) behave differently than networks with partial or no randomization. By summarizing each curve in (a) and (b) by its area under the curve (AuC), we show in (c) a birds-eye view over all layers. All networks with $p < 1$ pass through distinct phases consisting of a *feature extraction* phase until `conv3_1`, followed by a *memorization* phase until `conv4_2`, followed by a final *clustering* phase. Interestingly, the case $p = 1$ shifts the process into earlier layers, explaining why layer-by-layer it appears as an outlier.

4 Experiments

4.1 Feature compression and memorization

We start by studying networks that have been forced into different level of memorization due to label randomization on their training set (Zhang et al., 2016). The level of induced memorization is controlled by setting a probability p for a training label to be randomized, i.e., $p = 0$ is the unmodified dataset and $p = 1$ gives fully random labels. Note that the capacity of these networks is sufficiently large such that the training accuracy is 1 in all cases, regardless of the value of p.

We perform experimental evaluations on several image datasets, namely CIFAR-10 (Krizhevsky & Hinton, 2009), Fashion-MNIST (Xiao et al., 2017), SVHN(Netzer et al., 2011), and ImageNet (Russakovsky et al., 2015), as well as on the Urban Sounds audio classification dataset (Salamon et al., 2014). We use a fully-connected network for Fashion-MNIST and various CNN architectures for the others, which we describe in more detail the appendix.

4.1.1 Layer by layer analysis

We start by analyzing the layers of an 11-layer CNN trained on CIFAR-10. We sampled 10 batches (one batch per class) of 50 images, and compressed the activation matrix at each layer individually down to various values of the non-negative rank. We then measured classification accuracy of the prediction. In this analysis we report average results for 60 neural networks, ten networks (with different random initializations) trained per randomization level p.

In Figure 2 (a) and (b) we show k vs. accuracy curves of networks trained with increasing levels of label randomization, at an early layer (`conv2_1`) and a deep layer (`conv4_1`) respectively. We can immediately see that networks trained on fully randomized labels ($p = 1$) behave differently than networks with partial or no randomization. Furthermore, note that in deep layers, memorizing networks are significantly less robust to NMF compression, i.e., their activations posses a high non-negative rank, which indicates high non-linearity with respect to the input, as discussed in Section 3.2. We can characterize each curve in (a) and (b) with a single number, its area under the curve (AuC). This allows us in Figure 2 (c) to generate a single figure for all layers. Networks with $p < 1$ display a similar feed-forward trend up until layer `conv3_1`. Since these networks differ from each

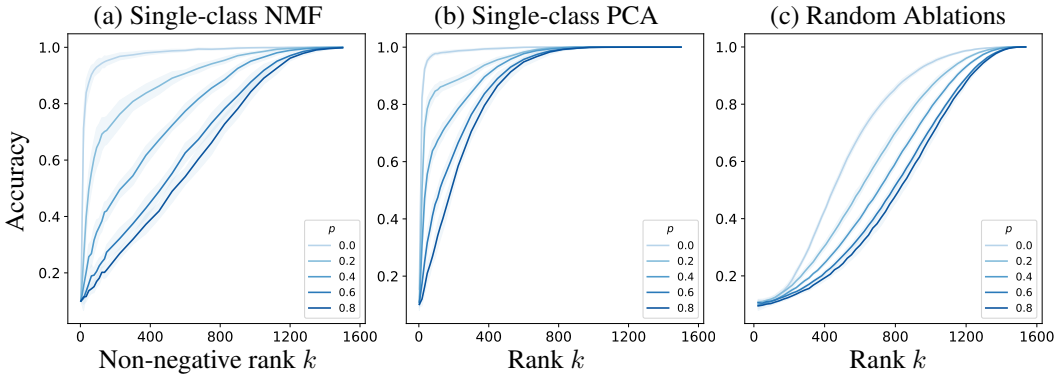

Figure 3: **Detecting memorization via compression**. We demonstrate this on networks trained with different levels of label randomization (p), and hence of memorization. (a) Due to its sensitivity to the non-linearity of ReLU activations, NMF successfully captures the level of memorization present in neural networks. (b) PCA compression is able to regain sufficient variance for good accuracy even with small values of k, which renders it less effective for detecting memorization. (c) Though not taking into account batch statistics, random ablations can distinguish between different levels of memorization, albeit less significantly. Compression was applied to the final three (convolutional) layers of CNNs trained on CIFAR-10.

other in no way other than the level of label randomization on the training data, we take this to be a generic *feature extraction* phase common to all of them. In the next phase, until `conv4_2`, we see a big difference between networks, such that more memorization (higher p) is correlated with lower AuC, i.e., higher non-negative rank and hence non-linearity of those layers with respect to single-class batches. We therefore localize memorization to these layers. Lastly, the phase only of `conv4_3` is where samples of the same class are clustered together, right before the final 10-dimensional classification layer (which is not shown). This final phase is in accordance with the premise that regardless of randomization level p, all of these networks achieve perfect training set accuracy.Interestingly, setting $p = 1$ shifts the process to earlier layers, explaining why layer-by-layer this case appears as an outlier.

4.1.2 Compression techniques

We compare different compression techniques with regards to detecting memorization. Given our choice of solving NMF under the Frobenius norm (Equation 3), a natural method to compare against is principal component analysis (PCA), which optimizes under the same norm but without the non-negativity constraint on its factors. We also consider random ablations, i.e., setting a random subset of columns in the activation matrix to zero, since this technique has been used previously to detect memorization (Morcos et al., 2018).

Rather than choosing a single layer, we sequentially apply compression to several layers. We target the final convolutional blocks of our CNNs, all of which contain three layers, each of which consists of 512 channels. In fully-connected networks, we applied compression to all layers.

In Figure 3 we give results for the CIFAR-10 dataset, which confirm that NMF compression is indeed more sensitive to memorization, due to the properties of the non-negative rank discussed in Section 3.2. PCA, which is less constrained, is more 'efficient' at compressing the activations, but is in turn less discriminative with respect to the level of memorization. Finally, we confirm that robustness to random ablations correlates with less memorization, however less so than NMF.

In Figure 4 we show additional results for single-class NMF on three additional datasets and network architectures (described in the appendix), including a fully-connected network for Fashion-MNIST. The results in (d) and (e), of applying PCA and NMF to multi-class batches, show that such batches produce activations with higher rank or non-negative rank compared to single-class batches. This is a result of the network trying to separate samples of different labels.

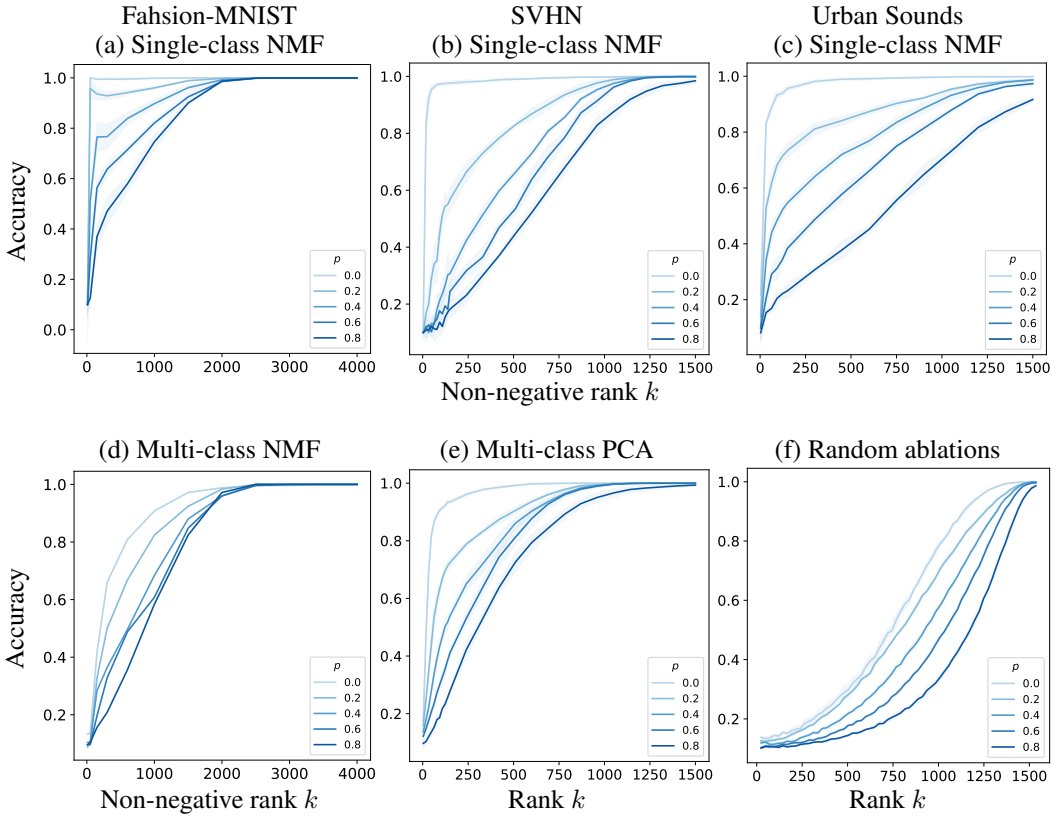

Figure 4: **Memorization across various datasets and network architectures**. We show that NMF-based compression is sensitive to memorization in diverse settings. Each column shows results for a specific dataset and network architecture. (a, b, d, e) We show that network layers are considerably more linear with respect to single-class batched than with respect to multi-class batches. (b,c,e,f) PCA and random ablations show less sensitivity to memorization compared with NMF.

4.2 Feature compression and generalization

We have shown results for networks forced into memorization due to label randomization. In this section we show our technique is useful for predicting good generalization in a more realistic setting, without artificial noise.

In addition to the experiments below, we refer the reader to Section 6.3 in the appendix, where predict *per-class* generalization of a pre-trained VGG-19 network on a ImageNet classes.

4.2.1 Compression techniques

We trained 96 CNN classifiers on CIFAR-10, over a grid of hyper-parameter values for the batch size, weight decay and optimization algorithm, SGD vs. ADAM (Kingma & Ba, 2015). Following the same procedure as above, for each of the three methods, NMF, PCA, or random ablations, we computed the k vs. accuracy curves for each network, targeting its final convolutional block. In Figure 5 we compare the area under the curve (AuC) of each curve with the average generalization error on the test set.

While all three methods show correlation with generalization error, NMF is most correlated with a Pearson correlation of -0.82, followed by PCA with -.064 and random ablation with -0.61.

4.2.2 Early stopping

We test whether our method can detect memorization during training. Doing so would allow us to perform early stopping, i.e., stop training as memorization begins to decrease generalization.

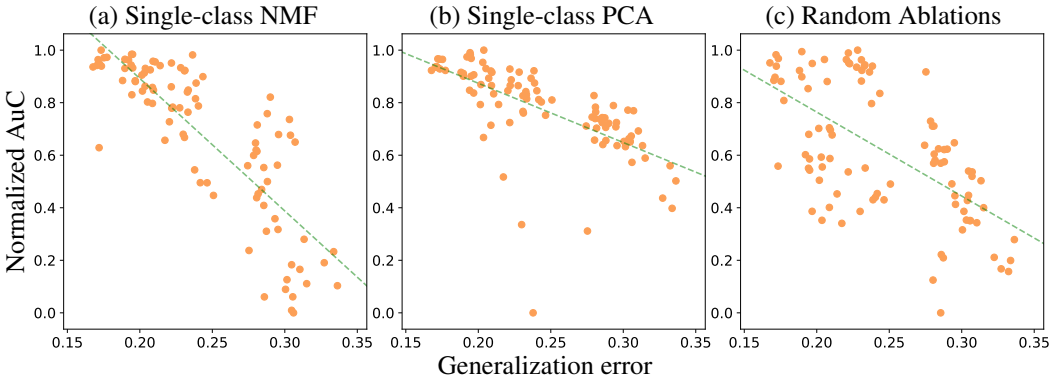

Figure 5: **Detecting generalization via compression**. While all three methods show correlation with generalization error, NMF is most correlated with a Pearson correlation of -0.82, followed by PCA with -.064 and random ablation with -0.61.

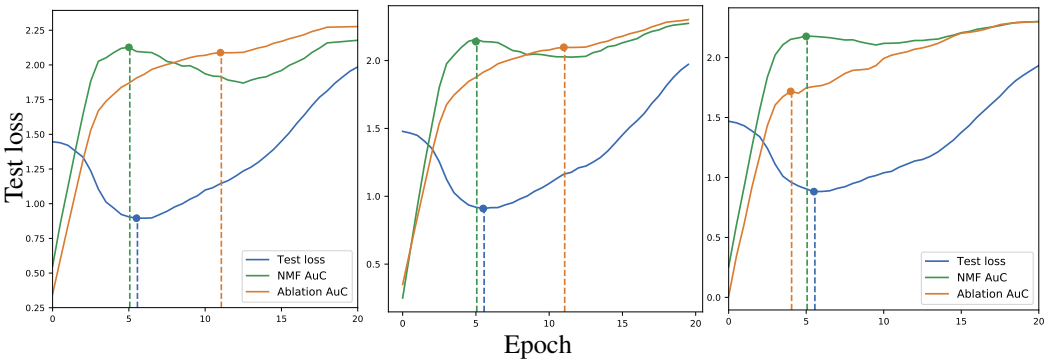

Figure 6: **Early stopping** for CNN training on CIFAR-10. The test error is (in blue) starts to increase after about the 5th epochs, indicating the start of overfitting. Using our proposed single-class NMF approach, we can detect the test error turning point. We show the area under the curve (AuC) for the single-class NMF approach (in green) for the accuracy measures as discussed in Section 4.1.2. Similarly, we show the AuC when performing random ablations (in orange). The NMF AuC curve correlates with the test error curve, as seen by the proximity of their extrema.

We trained CNNs on CIFAR-10 with the original labels. Each network was trained for 10K batches with a batch size of 100. We recorded the test set error every 250 batches, and tested the non-linearity of the deepest three convolutional layers using our NMF-based approach with a coarse grid on k. As before, we compute the area under each k vs. accuracy curve as in Figures 3 (c). Finally, we also computed the area under the curve produced by random ablations.

Results of three instances with different initializations are shown in Figure 6. These include the test error (in blue), our single-class NMF method (in green), and random ablations (in orange). We smooth the plots using a radius of two epochs to reduce the noise. The blue dashed lines mark the local minima of the test loss in Figure 6. The green and orange dashed lines indicate the location of the first local maxima of the NMF and random ablation AuC curves after smoothing has been applied. We notice that the test loss minima align almost precisely with the maximum NMF AuC.

5 Conclusion

We have introduced a notion of a ReLU layer's non-linearity with respect to an input batch, which is based on its proximity to a linear system. We measure this property indirectly via NMF applied to deep activations of single-class batches. While more analysis is required before definite guarantees could be given, we find that our approach is successful in detecting memorization and generalization across a variety of neural network architectures and datasets.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentangling in deep representations. *arXiv preprint arXiv:1706.01350*, 2017.

Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *International Conference on Machine Learning (ICML)*, 2017.

Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 1998.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016.

Edo Collins, Radhakrishna Achanta, and Sabine Susstrunk. Deep feature factorization for concept discovery. In *The European Conference on Computer Vision (ECCV)*, 2018.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. *arXiv preprint arXiv:1703.04933*, 2017.

Samuel Fiorini, Volker Kaibel, Kanstantsin Pashkovich, and Dirk Oliver Theis. Combinatorial bounds on nonnegative rank and extended formulations. *Discrete Mathematics*, 313(1):67 – 83, 2013.

Nicolas Gillis and FranÃğois Glineur. On the geometric interpretation of the nonnegative rank. *Linear Algebra and its Applications*, 437(11):2685 – 2712, 2012. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2012.06.038. URL http://www.sciencedirect.com/science/article/pii/S0024379512005022.

Emad M Grais and Hakan Erdogan. Single channel speech music separation using nonnegative matrix factorization and spectral masks. In *Digital Signal Processing (DSP)*, pp. 1–6. IEEE, 2011.

David Guillamet and Jordi Vitria. Non-negative matrix factorization for face recognition. In *Topics in artificial intelligence*, pp. 336–344. Springer, 2002.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Computational Complexity, 2003. Proceedings. 18th IEEE Annual Conference on*, pp. 118–134. IEEE, 2003.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

Tengyuan Liang, Tomaso Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. *arXiv preprint arXiv:1711.01530*, 2017.

Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, pp. 5, 2011.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pp. 5947–5956, 2017.

Roman Novak, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. *ICLR*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research. In *22nd ACM International Conference on Multimedia (ACM-MM'14)*, pp. 1041–1044, Orlando, FL, USA, Nov. 2014.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Vladimir Vapnik. *Statistical learning theory. 1998*. Wiley, New York, 1998.

Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

Shengjie Wang, Abdel-rahman Mohamed, Rich Caruana, Jeff Bilmes, Matthai Plilipose, Matthew Richardson, Krzysztof Geras, Gregor Urban, and Ozlem Aslan. Analysis of deep neural networks with extended data jacobian matrix. In *International Conference on Machine Learning*, pp. 718–726, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pp. 267–273. ACM, 2003.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

6 APPENDIX

6.1 NEURAL NETWORK ARCHITECTURES

CIFAR-10

Type	Out dim.	Kernel	Padding	Stride
Conv_+	64	3	1	1
Conv_+	64	3	1	1
Conv_+	128	3	1	2
Conv_+	128	3	1	1
Conv_+	128	3	1	1
Conv_+	256	3	1	2
Conv_+	256	3	1	1
Conv_+	256	3	1	1
Conv_+	512	3	1	2
Conv_+	512	3	1	1
Conv	512	3	1	1
Linear	10	-	-	-

Urban Sounds

Type	Out dim.	Kernel	Padding	Stride
Conv_+	64	3	1	1
MaxPool	-	2	-	1
Conv_+	128	3	1	1
Conv_+	128	3	1	1
MaxPool	-	2	-	1
Conv_+	256	3	1	1
Conv_+	256	3	1	1
MaxPool	-	2	-	1
Conv_+	512	3	1	1
Conv_+	512	3	1	1
MaxPool	-	2	-	1
Linear_+	4096	-	-	-
Linear_+	4096	-	-	-
Linear	10	-	-	-

SVHN

Type	Out dim.	Kernel	Padding	Stride
Conv_+	64	3	1	1
Conv_+	64	3	1	1
Conv_+	128	3	1	2
Conv_+	128	3	1	1
Conv_+	256	3	1	2
Conv_+	256	3	1	1
Conv_+	512	3	1	2
Conv	512	3	1	1
Linear	10	-	-	-

Fashion-MNIST

Type	Out dim.	Kernel	Padding	Stride
Linear_+	128	-	-	-
Linear_+	512	-	-	-
Linear_+	2048	-	-	-
Linear_+	2048	-	-	-
Linear	10	-	-	-

Table 1: Neural architecture used for each dataset in Section 4.

The exact architectures we used for each dataset are given in Table 1. We denote a linear or convolutional layer followed by a ReLU as Linear_+ and Conv_+, respectively.

6.2 ABLATING NMF AND PCA DIRECTIONS

It is interesting to study the impact of *ablating* the activation in the directions found by NMF and PCA by forward propagating the residual, i.e.,

$$\boldsymbol{A}_{i+1} = \max\Big((\boldsymbol{A}_k - \tilde{\boldsymbol{A}}_k)W_{i+1}, 0\Big) \tag{5}$$

This is interesting because in the case of PCA, for instance, the top k directions are those that capture most of the variance in the activation matrix, and presumably the k directions found by NMF are of similar importance. This is not true for the random ablations, where the ablated directions are of no special importance.

In Figure 7 we see that networks with no induced memorization that are *most vulnerable* to ablation of NMF and PCA direction. In other words, while non-memorizing networks are more robust to *random* ablations, they are not robust to ablations of specific important directions. This is in contrast to the interpretation of Morcos et al. (2018) that non-memorizing networks are more robust to ablations of single directions.

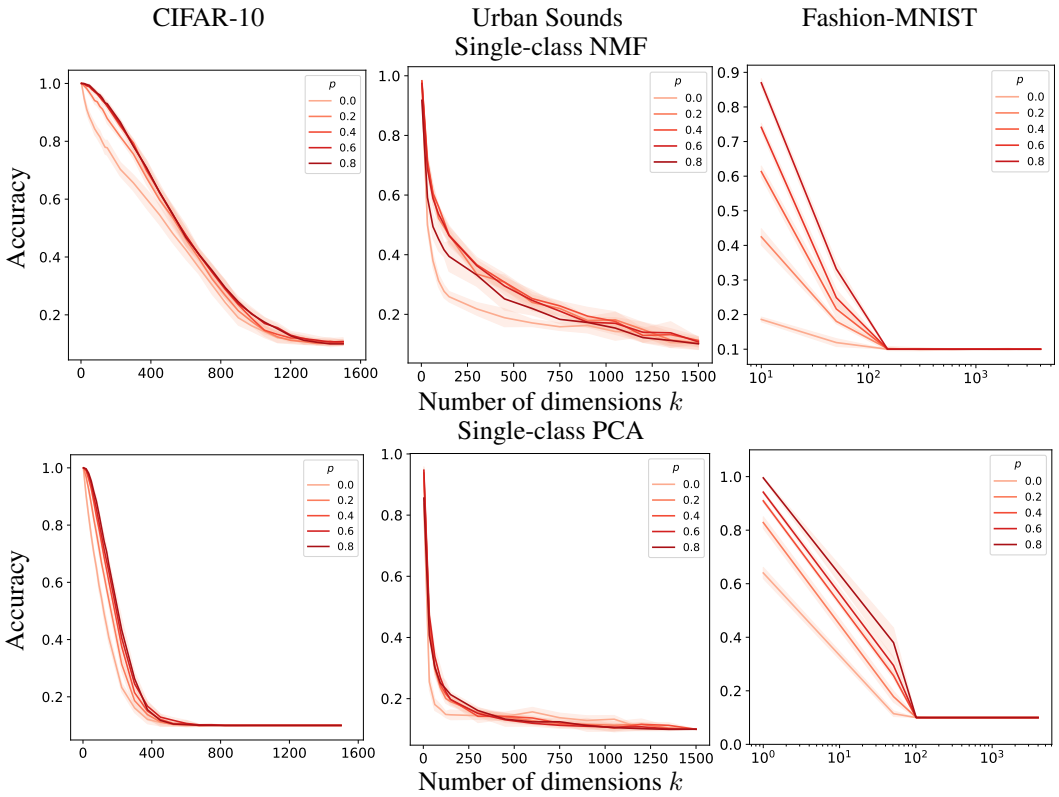

Figure 7: **NMF and PCA directions are more important** for networks with induced-memorization. Each column shows results for a specific dataset and network architecture. Compared to random ablation as in Figures 3 and 4, where networks with induced-memorization are most robust, in all cases we see that ablation in NMF and PCA directions hurts their performance more, compared to memorizing networks.

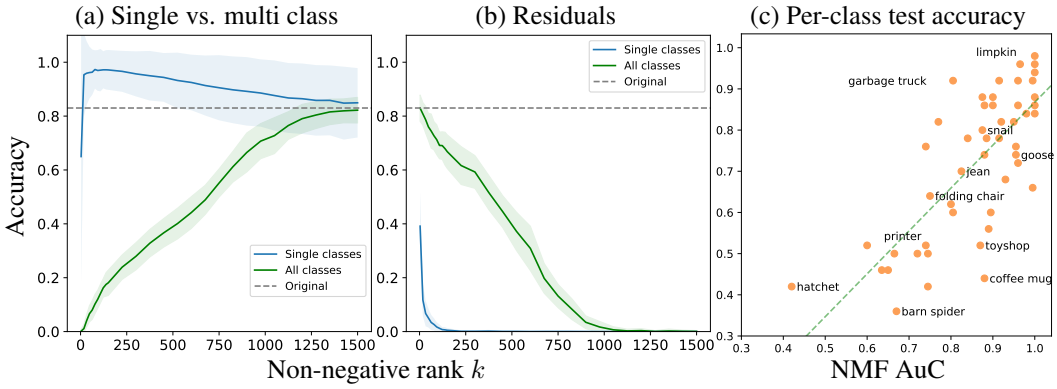

Figure 8: **NMF compression on VGG-19**. (a) Deep VGG layers are highly linear with respect to single-class batches, as indicated by high accuracy for small dimensions of k. Compression can have a denoising effect and improve upon the baseline accuracy of the batch (dashed line). (b) Removing NMF directions causes a dramatic drop in accuracy, more so on single-class batches. (c) Per-class test set accuracy is significantly correlated with the area under the k vs. accuracy curve (NMF AuC).

6.3 NMF ON VGG-19

The VGG-19 model (Simonyan & Zisserman, 2014), trained on ImageNet (Russakovsky et al., 2015), is known for its good generalization ability, as evident by its widespread use as a general feature extractor. We use a pre-trained model here as an example of a well-generalizing network and analyze it with our method.

We apply NMF compression to the three deepest convolutional layer, on activations of both single-class batches and multi-class batches. We select 50 random classes from ImageNet and gather batches of 50 training samples from each class.

In Figure 8, shown in blue, NMF applied to single-class batches, has a denoising effect and improves over the baseline accuracy of the batch, shown as a dashed line. As the constraint on k is relaxed, that accuracy drops back to its baseline level.

We contrast this behavior with the one shown in green when using multi-class batches. Here, only when k is large do we regain baseline accuracy, and sensitivity to ablation is similarly diminished. This is due to the critical role non-linearity plays in separating the different classes. Ablating the NMF directions dramatically reduces classification accuracy. Finally, in Figure 8 (c) we show there is a significant per-class correlation (Pearson $r = 0.78$) between NMF AuC and accuracy on test accuracy on batches from the ImageNet test set.