Appendixes to *Is my Neural Net driven by the MDL Principle?* ⋆

Eduardo Brandao(✉)[1][0000−0002−7146−8255], Stefan Duffner[2][0000−0003−0374−3814], Rémi Emonet[1][0000−0002−1870−1329], Amaury Habrard[1,3][0000−0003−3038−9347], François Jacquenet[1][0000−0002−0653−0710], and Marc Sebban[1][0000−0001−6851−169X]

[1] Université Jean Monnet Saint-Etienne, CNRS, Institut d Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, SAINT-ETIENNE, France
Eduardo.Brandao@univ-st-etienne.fr
[2] CNRS, INSA-Lyon, LIRIS, UMR5205, Université de Lyon, F-69621 Villeurbanne, France
Stefan.Duffner@liris.cnrs.fr
[3] Institut Universitaire de France (IUF)

Abstract. The Minimum Description Length principle (MDL) is a formalization of Occam's razor for model selection, which states that a good model is one that can losslessly compress the data while including the cost of describing the model itself. While MDL can naturally express the behavior of certain models such as autoencoders (that inherently compress data) most representation learning techniques do not rely on such models. Instead, they learn representations by training on general or, for self-supervised learning, pretext tasks. In this paper, we propose a new formulation of the MDL principle that relies on the concept of signal and noise, which are implicitly defined by the learning task at hand. Additionally, we introduce ways to empirically measure the complexity of the learned representations by analyzing the spectra of the point Jacobians. Under certain assumptions, we show that the singular values of the point Jacobians of Neural Networks driven by the MDL principle should follow either a power law or a lognormal distribution. Finally, we conduct experiments to evaluate the behavior of the proposed measure applied to deep neural networks on different datasets, with respect to several types of noise. We observe that the experimental spectral distribution is in agreement with the spectral distribution predicted by our MDL principle, which suggests that neural networks trained with gradient descent on noisy data implicitly abide the MDL principle.

Keywords: Neural Networks · MDL · Signal-Noise · Point Jacobians.

⋆ This work has been funded by a public grant from the French National Research Agency (ANR) under the "France 2030" investment plan, which has the reference EUR MANUTECH SLEIGHT - ANR-17-EURE-0026. This work has also been partly funded and by a PhD grant from the French Ministry of Higher Education and Research.

1 Appendixes

1.1 Finding a finite precision network that overfits

Memorizing data with Neural Networks Two-layer ReLU feed forward neural networks can do this for given, arbitrary sample size n, data in arbitrary dimension d with surprising ease: as stated in [7], Theorem 1, within the set of such networks $\mathcal{N}$ with at least $2n + d$ parameters, there is at least one $N \in \mathcal{N}$ that will be able to compress y perfectly, by expressing it in terms of x. This is *not* at odds with the MDL principle (i) since MDL states that compressibility of data without a rule is *unlikely*, rather than impossible (ii) the description lenght of the network is not taken into account (see[4]).

The proof of theorem 1 in [7], implicitly assumes infinite precision in the weights of the network, which would take up infinite, and thus unavailable, space. For completeness, we briefly describe the gist of the proof in [7], before adapting it to the finite precision case.

The proof rests on a Lemma that constructs a matrix A that is lower-triangular and has non-zero and distinct real diagonal elements: the first differences of an increasing sequence. A is hence non-singular, since the diagonal elements of a triangular matrix are its singular values. The authors then proceed to stating the overfitting problem, for a 2-layer ReLU network, as the solution of linear system in a matrix B. This matrix can be made of type A via a judicious choice of network parameters a and b. Precisely, one needs to chose a, b such that for every sample x_j we have $a^\top x_j < a^\top x_{j+1}$. This can always be done for distinct x_i, by the Archimedian property of the reals. It remains to select b_j such that $a^\top x_j < b_j < a^\top x_{j+1}$, which can be done because $\mathbb{R}$ is complete. The remaining parameters of the network w are precisely the solutions of the system in B.

Memorizing data with finite-precision Neural Networks In finite precision, this cannot be done in general. The number of significant figures of $a^\top x_j$ is at most that of x_j. The problem of finding constants a and b such that we can place a b between every two $a^\top x_j$ can be done surely by picking b with one more significant figure than x_j. The number of significant figures in w, on the other hand, depends on that of y as well: it is the minimum between the number of significant figures of x plus one, and the number of significant figures of y.

We thus have the following proposition:

Proposition 1. *In order to be able to overfit data x, y with s_x, s_y significant figures, it suffices a neural network with n parameters with one more significant digit than x, d parameters with the same number of significant digits as x, and n parameters with the same precision as y for an expected number of bits of* $(n(s_x + 1) + ds_x + ns_y) \log_2 10$.

Memorizing cifar-10 Typically, the number of significant figures in ML pipelines is fixed, and it is the same for data and for weights (a 32 bit float). In the unlikely

event that data is too closely packed (some data only differs by one in the last significant digit), then there is no guaranteed overfit.

Incidentally, this precisely gives a lower bound to the parametric complexity of the model "2-layer ReLU networks": they will be able to perfectly fit data with precision s_x, s_y if the number of parameters satisfies the constraints above.

Note that the number of significant digits in 8 bit images is 3, and the number of significant digits in 10 classification choices is 1. If the model can be compressed to less than $\left(6 \times 10^4 \times (3+1) + 32^2 \times 3 + 6 \times 10^4 \times 1\right) \log_2 10$, which amounts to about 125 kB, it cannot thus be expected to overfit.

1.2 Deriving the local approximation, alternative derivation

Within each local patch the Jacobian J has a singular vector decomposition $J = U\Sigma V^\top$. Assume without loss of generality that $J \in \mathbb{R}^{n\times m}$. Then $U \in \mathbb{R}^{m\times m}$ is orthogonal, $\Sigma \in \mathbb{R}^{m\times n}$ is a diagonal matrix and $V \in \mathbb{R}^{n\times n}$ is orthogonal as well. The entropy $H(V^\top \delta X) = H(\delta X)$ since V^T has determinant one everywhere. In the same way, $H(J\delta X) = H(U^\top J\delta X) = H(\Sigma V^\top \delta X)$. In the case of an embedding, the matrix Σ has a number of zero components along its diagonal, which will, upon multiplying by $V^\top \delta X$ produce a vector that has a number of zero components independently of the other components. Hence, the entropy of the zero part and the nonzero part is the same as the entropy of the nonzero part. So the entropy above is just the entropy of the non-zero components after acting on data with $V^\top$. Explicitly, $H(\sigma_1 v_1^\top \delta X, \ldots, \sigma_k v_k^\top \delta X, 0, \ldots, 0) = H(\sigma_1 v_1^\top \delta X, \ldots, \sigma_k v_k^\top \delta X)$. Plugging into our objective we obtain

$$\begin{aligned}\max_J \lambda H(J\delta X) - H(J\Delta) &= \max_\sigma \lambda H(\sigma_1 v_1^\top \delta X, \ldots, \sigma_k v_k^\top \delta X) - H(\sigma_1 v_1^\top \Delta, \ldots, \sigma_k v_k^\top \Delta) \\ &\geq \max_\sigma \left\{ \max_i \lambda H(\sigma_i v_i^\top \delta X) - H(\sigma_1 v_1^\top \Delta, \ldots, \sigma_k v_k^\top \Delta) \right\}\end{aligned}$$

Noting that

$$\begin{aligned}H(\sigma_1 v_1^\top \Delta, \ldots, \sigma_k v_k^\top \Delta) &\leq \sum_i H(\sigma_i v_i^\top \Delta) \\ &= \sum_i H(\Delta_i) + \log \sigma_i\end{aligned}$$

where we called $v_i^\top \Delta := \Delta_i$. Doing the same for δX and plugging in our objective above, we obtain

$$\begin{aligned}\max_J \lambda H(J\delta X) - H(J\Delta) &\geq \max_\sigma \left\{ \max_i \lambda H(\sigma_i v_i^\top \delta X) - H(\sigma_1 v_1^\top \Delta, \ldots, \sigma_k v_k^\top \Delta) \right\} \\ &= \max_\sigma \left\{ \lambda \max_i \{\log \sigma_i + H(\delta X_i)\} - H(\sigma_1 v_1^\top \Delta, \ldots, \sigma_k v_k^\top \Delta) \right\} \\ &\geq \max_\sigma \left\{ \lambda \max_i \{\log \sigma_i + H(\delta X_i)\} - \left(\sum_i H(\Delta_i) + \log \sigma_i \right) \right\}\end{aligned}$$

Note now that we can do three things to maximize this local lower bound: imagine that we start training and there is one component of the data that happens to have an image with larger entropy; assuming that the singular values at start are the same, then this is because we are more aligned with the data. And so we will promote this alignment by increasing both the singular value along that direction and rotating it in order to improve the alignment.

The last terms are interesting as well: in order to reduce them and thus increase the lower bound, we can do two things: reduce the mean entropy of the projections of noise onto the singular directions and reduce the mean logarithm of the singular values of the Jacobian. Although the latter can be done without restriction, the former depends on the local shape of noise. Without further assumptions, the only sure way to reduce the mean entropy is to remove dimensions altogether.

But note that since the logarithm can be very negative, it's even better to *keep* the dimensions and just focus on decreasing the singular values to as close to zero as possible.

1.3 Controlling the approximation error of the local objective

We now show that a given $x_1, \ldots, x_N$ and an error budget E, a set of radii can be chosen such that the maximum linear approximation error does not exceed it, and these radii are inversely proportional to the largest principal singular value of the point Hessian matrices. Conversely, for a compact domain, a set of radii can be chosen such that every point is inside one of the neighborhoods of the $x_1, \ldots, x_N$ that minimizes the total approximation error.

Intuitively, since the Hessian matrix at a point controls the curvature, the curvature along the maximum curvature direction controls how far we are able to go away from the point while not changing the Jacobian too much.

Proposition 2. *Let $f : A \subseteq \mathbb{R}^n \to B \subseteq \mathbb{R}^m$ be analytical, with A compact and $x_1, \ldots, x_N \subseteq A$. Then given $E^k > 0$, a set of balls $\{V_k\}_{k=1\cdots N}$ centered at x_k and with radius r_k can be chosen such that the approximation error is upper bounded by*

$$\forall_{k=1\cdots N},\ \sup_{\substack{w\in V_k \\ i=1\cdots m}} \frac{1}{2}\sigma_1(\nabla^2 f^i|_w) \cdot \|r_k\|^2 = E^k$$

where $\sigma_1(\nabla^2 f^i|_w)$ is the first singular value of the Hessian matrix of the component f^i calculated at $w \in V_k$.

Proof. For each component f^i of f, Taylor's theorem states that the approximation error of $f^i(x_k + r_k) - f^i(x_k) \approx \left(\nabla f^i|_{x_k}\right) r_k$, along a radius r_k, in Lagrangean form, is $\frac{1}{2} {r_k}^\top \left(\nabla^2 f^i|_w\right) r_k$, where w is a point between $x_k, x_k + r_k$. The approx-

imation error is thus

$$\begin{aligned}\frac{1}{2}{r_k}^\top \left(\nabla^2 f^i|_w\right) r_k &= \frac{1}{2}\frac{{r_k}^\top \left(\nabla^2 f^i|_w\right) r_k}{{r_k}^\top r_k} \cdot \|r_k\|^2 \\ &\leq \frac{1}{2}\sigma_1(\nabla^2 f^i|_w) \cdot \|r_k\|^2 \\ &\sup_{\substack{w\in V_k \\ i=1\cdots m}} \frac{1}{2}\sigma_1(\nabla^2 f^i|_w) \cdot \|r_k\|^2\end{aligned}$$

, where we used the definition of the first singular value in terms of the Rayleigh quotient to establish this result on the sup norm. A result that holds for other norms follows from convexity of the norms and the bound on each of the components of the vector of the Hessian matrices.

To see the converse, consider that given a compact set and a point, there is always a ball that contains it. Hence so would a union of such balls. Since each of the radii sets an upper bound for the local approximation error, with $\sup_{\substack{w\in V_k \\ i=1\cdots m}} \frac{1}{2}\sigma_1(\nabla^2 f^i|_w) := \sigma_1^k$, we can write the total error as

$$\min_{r_k} \sum_{k=1}^{N} \sigma_1^k \cdot r_k^2$$

with the constraint that $A \subseteq V_1 \cup \cdots \cup V_N$.

1.4 Combining local objectives

In order to maximize the combined local objective

$$\max_{\sigma} \left\{\lambda M \mathbb{E}\left[\log \sigma\right] + \lambda H(X) - H(\Delta) - \bar{N} M \mathbb{E}\left[\log \sigma\right]\right\}$$

and aiming at an expression for the spectral distribution, we follow the strategy highlighted in **??**, which we repeat below for completeness, for the combined objective

- aligning J with δX and then maximizing the logarithm of the singular values in the non-zero dimensions: if δX is locally low-dimensional, the singular values that get maximized are few.
- aligning J with Δ and then minimizing the logarithm of the singular values in the non-zero dimensions: since Δ tends to be relatively high-dimensional, all singular values of J tend to be minimized.

Aligning the Jacobian and δX implies that the entropy $H(\delta X^{i_k})$ is maximal. Using the manifold hypothesis, we assume that the maximal entropy component accounts for most of the entropy, that is $\max_{i_k} H(\delta X_k^{i_k}) \approx H(\delta X_k)$. As explained in **??**, the maximal entropy component does not necessarily correspond to the

maximal singular value: during training, singular spaces corresponding to higher-order singular values will also be "selected". Taking this "selection" as a one-sample estimate of the mean justifies replacing the maximization over i_k in the expression above with $\mathbb{E}[\log \sigma_k] + H(\delta X_k)$. Under the assumption of cross-patch independence, we have $\sum_{i=1}^{M} H(\delta X_k) = H(X)$ and similarly for Δ. The expression below follows from linearity of expectation:

$$\max_{\sigma} \left\{ \lambda M \mathbb{E}\left[\log \sigma\right] + \lambda H(X) - H(\Delta) - \bar{N} M \mathbb{E}\left[\log \sigma\right] \right\}$$

Since each of the terms in the sum above is negative, this expression is non-positive. At the maximum, we thus have

$$\mathbb{E}\left[\log \sigma\right] = \frac{H(\Delta) - \lambda H(X)}{M(\lambda - \bar{N})}$$

1.5 Experimental setup

The experimental setup follows [7] closely. We investigate two image classification datasets, namely the MNIST dataset [3] and the CIFAR10 dataset [2]. Both datasets are composed of 50,000 training and 10,000 validation images, distributed across 10 different classes. In CIFAR10, each image in the dataset has dimensions of 32x32, with 3 color channels. To scale the pixel values into the range of [0, 1], we normalize them by dividing each value by 255. Additionally, we center crop the images to obtain a size of 28x28, and normalize them by subtracting the mean and dividing the adjusted standard deviation independently for each image, adapting the per_image_whitening function in Tensorflow [1], as presented in[7]. The same procedure adapted to one channel, except center cropping which is unnecessary since MNIST images are 28x28, is performed on MNIST.

On both datasets, we use two common deep architectures, which were adapted to smaller image sizes/single-channel images: a simplified Inception model [5] and Alexnet [2]. As in [7], the simplified Inception model uses a combination of 1x1 and 3x3 convolution pathways, while the simplified Alexnet is constructed using two (convolution 5x5 $\rightarrow$ max-pool 3x3 $\rightarrow$ local-response-normalization) modules followed by two fully connected layers with 384 and 192 hidden units, respectively. We utilize a 10-unit linear layer for prediction, and to calculate the point Jacobians. All architectures employ the standard rectified linear activation functions (ReLU).

We also study two fully connected multi-layer perceptrons (MLPs): one having a hidden layer with 512 units, the other having three hidden layers of the same size.

For all experiments, we train the models using SGD with a momentum of 0.9, using an initial learning rate of 0.01. We apply a decay factor of 0.95 per epoch to adjust the learning rate, and train the models without weight decay, dropout, or any other explicit regularization techniques.

In all experiments, we calculate the Jacobian at the linear layer, using automatic differentiation with Pytorch's torch.autograd.functional.jacobian method. The point Jacobian spectrum is calculated at all training and test examples using Pytorch's torch.linalg.svdvals, which is a port of Numpy's.

The experimental spectral distributions were split at the first deepest trough. The lognormal fit of each modality, the probabilty plots and line of best fit were calculated using scipy [6].

1.6 A few fundamental results in Information theory

We repeat a few results in the main paper for readability, while extending some of the arguments.

Preliminaries and notation source code $C(X)$ (simply "code" from now, and often denoted C when there is no risk of ambiguity) for a r.v. X is a function from $\mathcal{X}$ the range of X to $\mathcal{D}^*$ the set of finite strings of a d-ary alphabet $\mathcal{D}$, associating $x \in \mathcal{X}$ to a codeword $C(x)$. The *length* of the codeword $l(x)$ is the number of elements in $C(x)$, and the expected code length is $L(X) := \mathbb{E}_X[l(x)]$. A code is said to be non-singular if every $x \in \mathcal{X}$ maps to an unique element of $\mathcal{D}^*$. An extension $C^*$of code C codes sequences $x_1x_2\cdots x_n$of elements of $\mathcal{X}$ as the concatenation of $C(x_1)C(x_2)\cdots C(x_n)$. A code is said to be uniquely encoded if its extension is non-singular. Since it every element in $\mathcal{X}$ is unambiguously encoded with a unique string, non-singular codes allow us to losslessly compress data.

Optimal codelength and irregular data Results (ii) and (iii) crucially rest on the Kraft-Macmillan inequality:

Theorem 1 (Kraft-Macmillan inequality). *For any uniquely decodable code C over an alphabet of size D, the codeword lengths $l_1, l_2, \ldots, l_m$ must satisfy the inequality*

$$\sum_i D^{-l_i} \leq 1 \tag{1}$$

Conversely, given a set of codeword lengths that satisfy 1, there exists a uniquely decodable code with these word lengths.

The idea of the proof for prefix codes (known as the Kraft inequality) is that prefix codes from a D - adic alphabet can be seen as the childless nodes on a rooted tree. No prefix code can then be among the descendants of another. Hence, the sum of the descendants of prefix codes cannot exceed the number of leaves of the tree: $\sum_i D^{l_{max}-l_i} \leq D^{l_{max}}$. The converse is established simply by noting that lengths that satisfy 1 can be placed on rooted tree. If they couldn't there would be more words of length l_i than descendants of non-used codes; but since the total mass at every level is constant, this cannot happen.

Theorem 2 (Optimal code length). *The expected code length for any uniquely decodable code C of a r.v. X over an alphabet of size D is greater than or equal to $H_D()$ the entropy calculated in base D, with equality holding iff $D^{-l_i} = p_i$*

To establish this, consider the difference between the entropy and the expected length. The result then follows from theorem 1 and non-negativity of relative entropy, which is a consequence of the concavity of logarithm (Jensen inequality). An optimal prefix code always exists (e.g. Huffman code), but for our purposes, it suffices that the Shannon-Fano code, which sets codeword lengths $l(x) = \lceil -\log p(x) \rceil$ is competitive in the sense that the probability that the expected length exceeds another code's by c bits does not exceed 2^{1-c}.

Finally, we give an informal argument to justify the statement that it is extremely unlikely that data with no regularities (with maximal entropy) can be compressed. By the Kraft-Macmillan inequality 1, for every prefix code of a r.v. X over an alphabet of size D, the expected codeword length is no greater than the entropy, with equality iff the $l_i = -\log_D p_i$. Assuming X is discrete, all n events have the same probability $\frac{1}{n}$. Hence, the expected code length (per symbol) is $L \geq -\sum_{i=1}^{n} p_i \log_D p_i = \log_D n$. The lower bound is what we can achieve simply by assigning each codeword to the leaves of a D-nary tree: the best code coincides with the worst possible code, and so data cannot be compressed.

1.7 Additional figures

Point Jacobian spectrum, full spectra For the figures detailing the full spectrum, i.e. figs. 3 through 11, note that train and test distributions are the same, since the underlying distributions are the same for this relatively simple dataset. We also note that the overall shape of each distribution does not change significantly with the addition of noise. As discussed in the main text, we note the clear bimodality in all spectral distributions, comparatively higher "lognormality" of Inception, and that the addition of noise increases the mean spectrum.

Finally, note that at the beginning of training, MLP and Alexnet's predictions are very local, since there is a great number of relatively small singular values (high peak), and more so with the addition of noise. This effect is also observed in Inception, but to much less extent. Using fig. 12 as illustration, MLP and Alexnet are more *conservative* than Inception, keeping close to the image of the training examples for small perturbations. This is similar to the strategies of generalization that we commonly use (e.g. maxent). Inception, on the other hand, which generalizes better, while not being conservative at all, which suggests that it does so by focusing on the signal.

Point Jacobian spectrum MNIST Figures 1 and 2 parallel those in the main text for the MNIST dataset.

Dataset noise illustration Figure 12 provides a graphical illustration of the representation building leading to bimodality. The example is for a classifier

$f : \mathbb{R}^3 \to \mathbb{R}^2$ and an autoencder g for visualization, but the overall idea extends to higher dimensions.

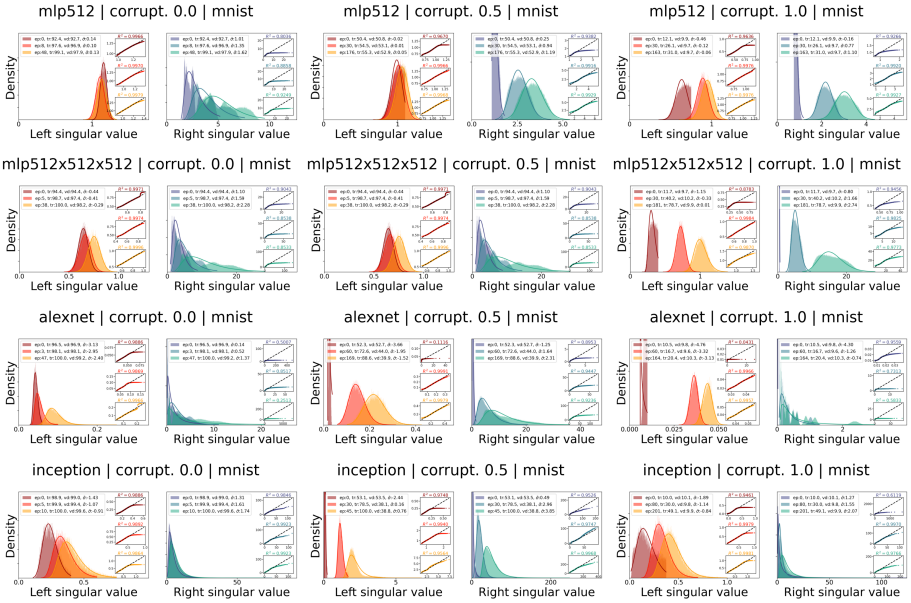

Fig. 1: Point Jacobian spectral distribution for *model* | ***label noise*** | *MNIST*, from first epoch to overfit. "Left" and "right" distributions (cf. 1.7) are represented separately for each triplet for clarity. The best fit lognormal plot is superimposed on each histogram, with the corresponding probability plot on the right, with the line of best fit (R^2 displayed on top). Legend elements, in order: epoch, training and validation accuracy, and the mean log spectrum.

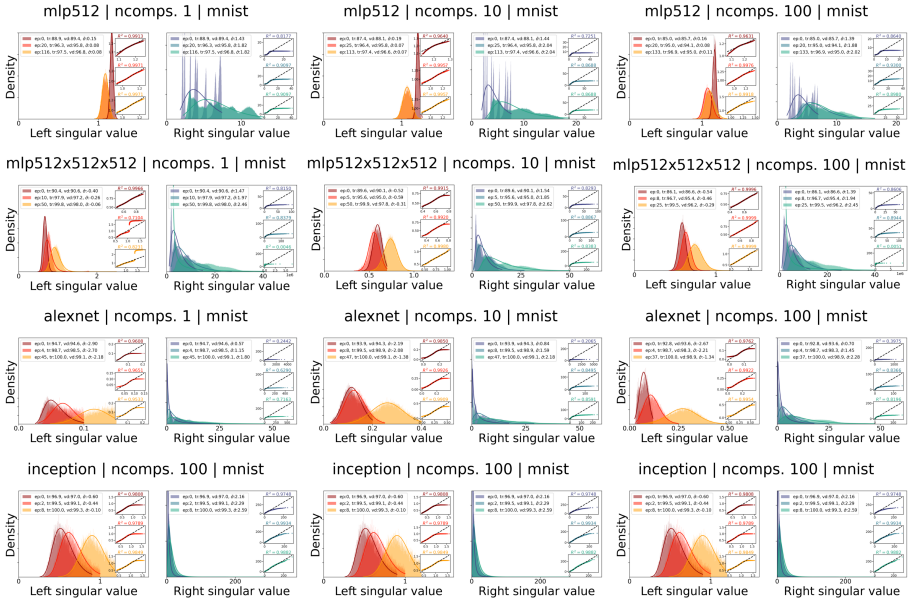

Fig. 2: Point Jacobian spectral distribution for *model* | ***nbr. pca comp.*** | *MNIST*, from first epoch to overfit. "Left" and "right" distributions (cf. 1.7) are represented separately for each triplet for clarity. The best fit lognormal plot is superimposed on each histogram, with the corresponding probability plot on the right, with the line of best fit (R^2 displayed on top). Legend elements, in order: epoch, training and validation accuracy, and the mean log spectrum.

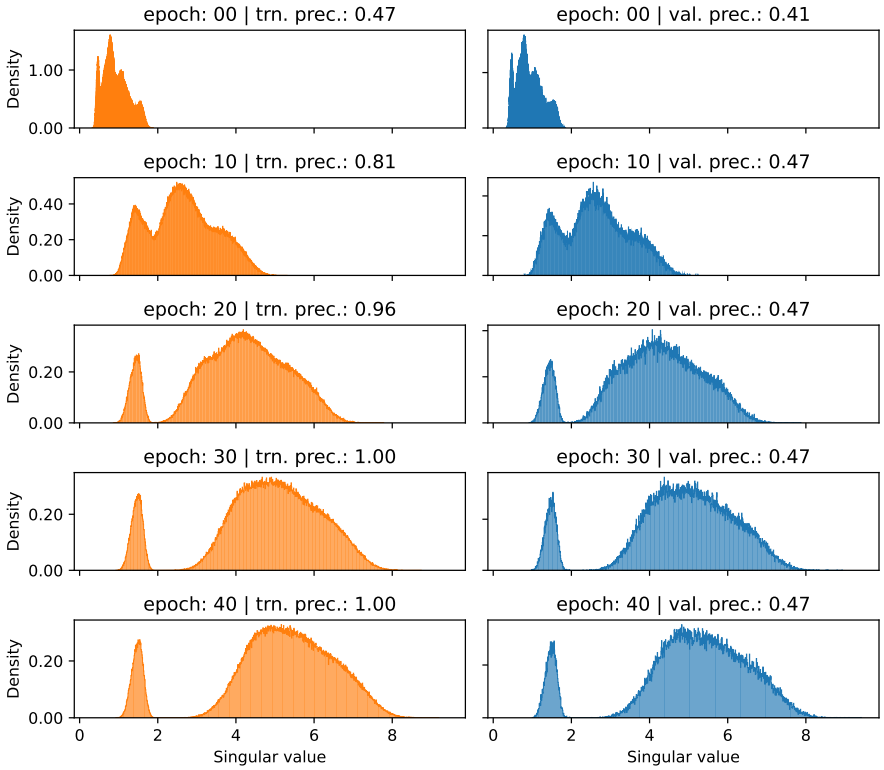

Fig. 3: Full point train (left) and validation (right) Jacobian spectrum for MLP trained on cifar-10, from first epoch to overfit, with no label noise. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

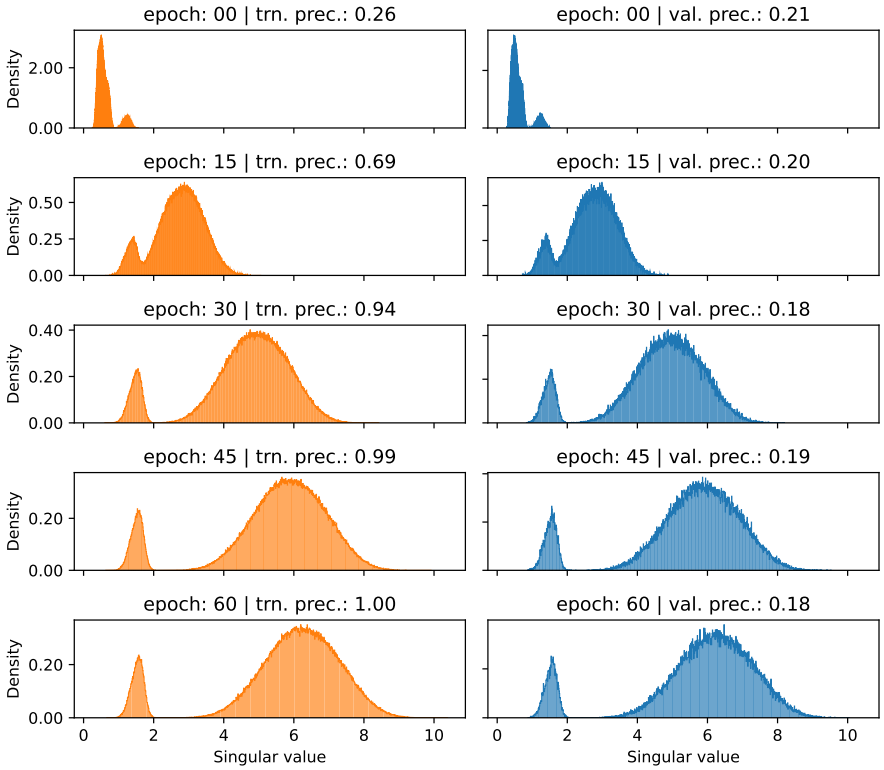

Fig. 4: Full point train (left) and validation (right) Jacobian spectrum for MLP trained on cifar-10, from first epoch to overfit, with label noise $p = 0.5$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

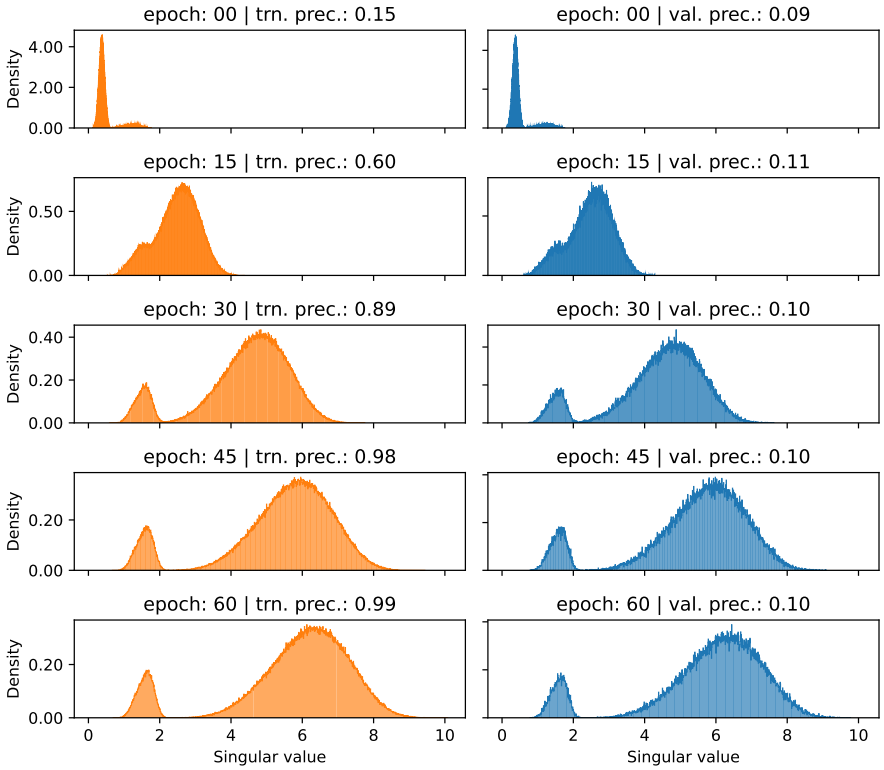

Fig. 5: Full point train (left) and validation (right) Jacobian spectrum for MLP trained on cifar-10, from first epoch to overfit, with label noise $p = 1.0$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

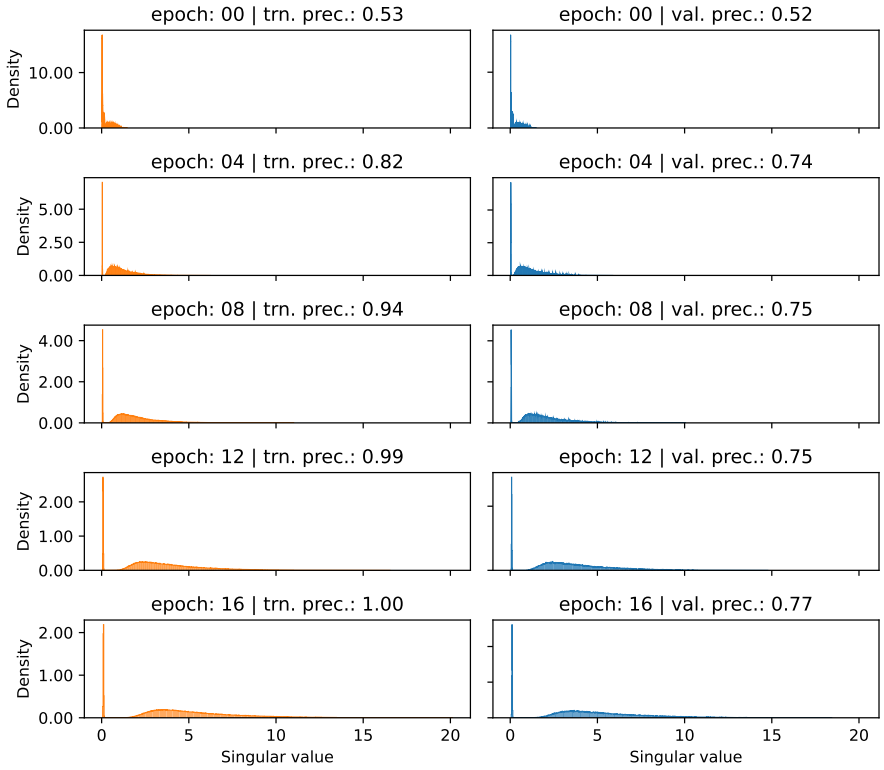

Fig. 6: Full point train (left) and validation (right) Jacobian spectrum for Alexnet trained on cifar-10, from first epoch to overfit, with label noise $p = 0.0$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

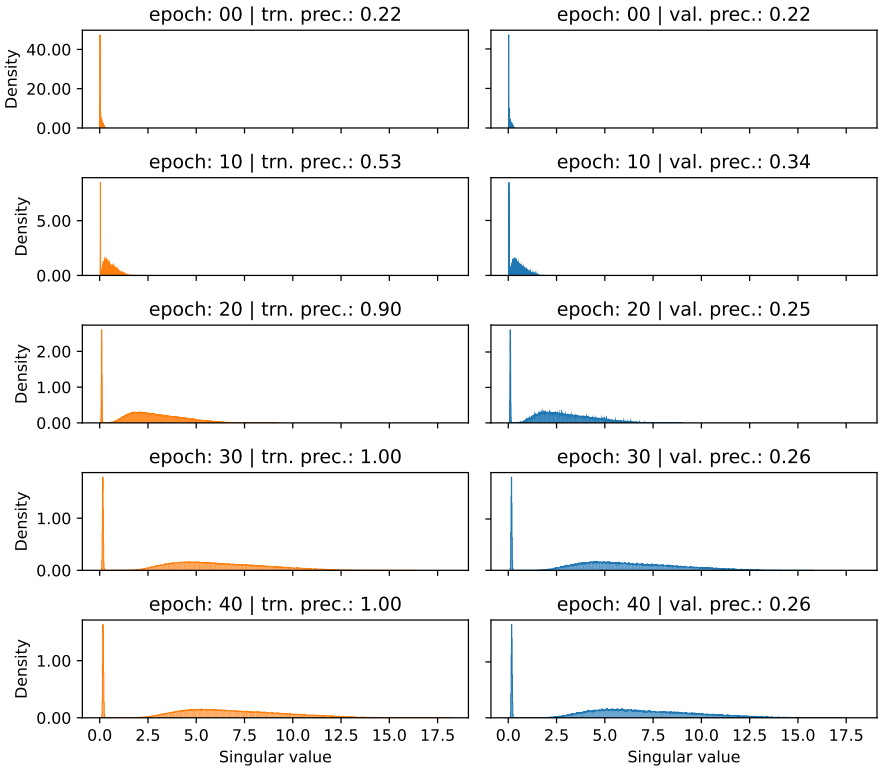

Fig. 7: Full point train (left) and validation (right) Jacobian spectrum for Alexnet trained on cifar-10, from first epoch to overfit, with label noise $p = 0.5$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

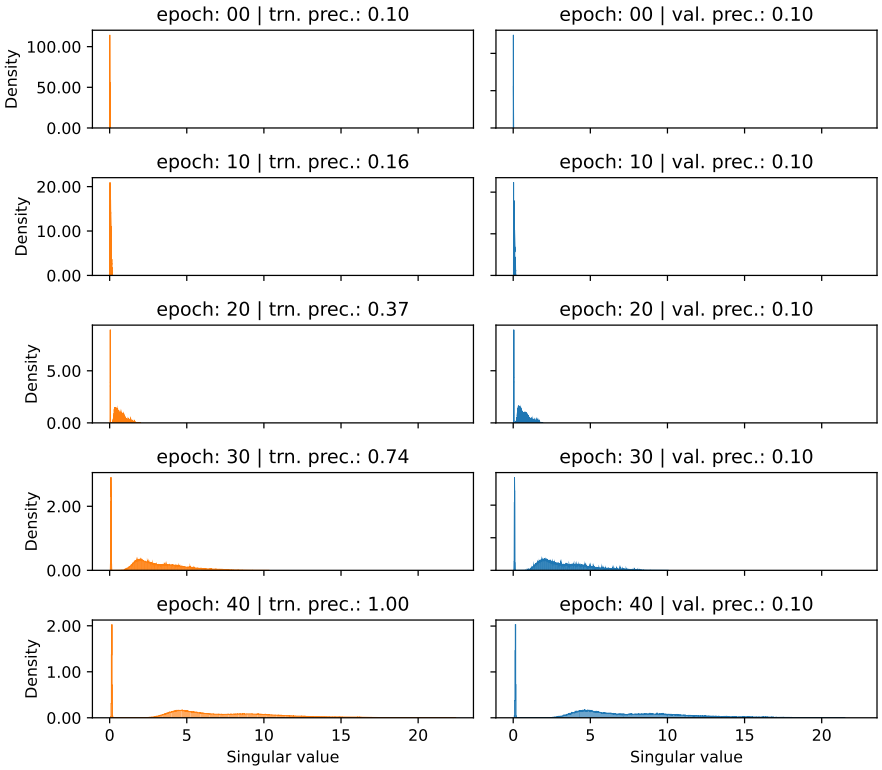

Fig. 8: Full point train (left) and validation (right) Jacobian spectrum for Alexnet trained on cifar-10, from first epoch to overfit, with label noise $p = 1.0$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

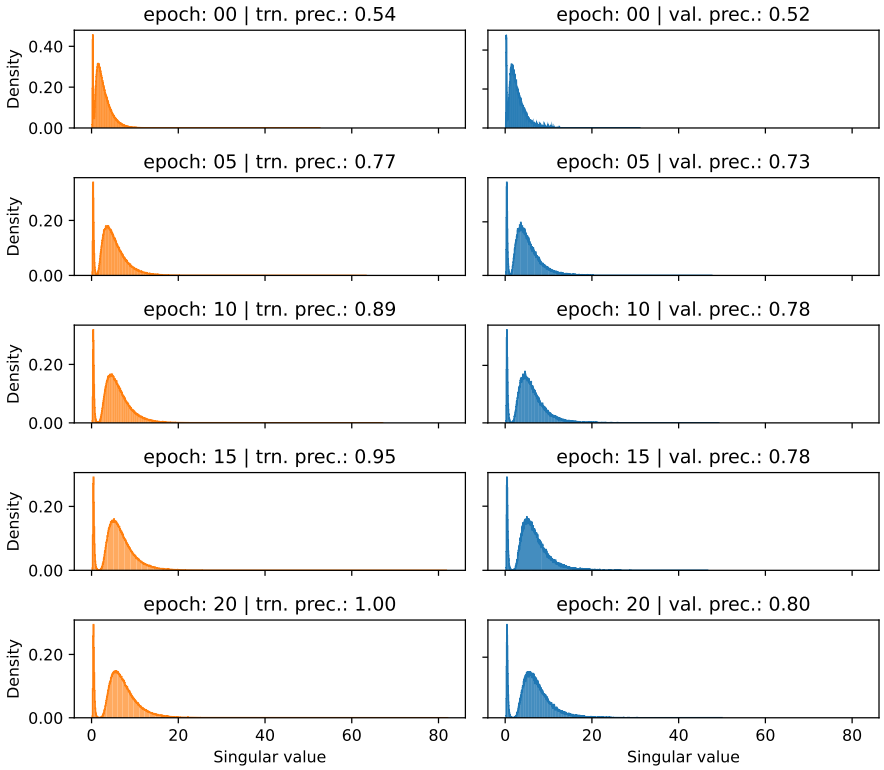

Fig. 9: Full point train (left) and validation (right) Jacobian spectrum for Inception trained on cifar-10, from first epoch to overfit, with label noise $p = 0.0$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

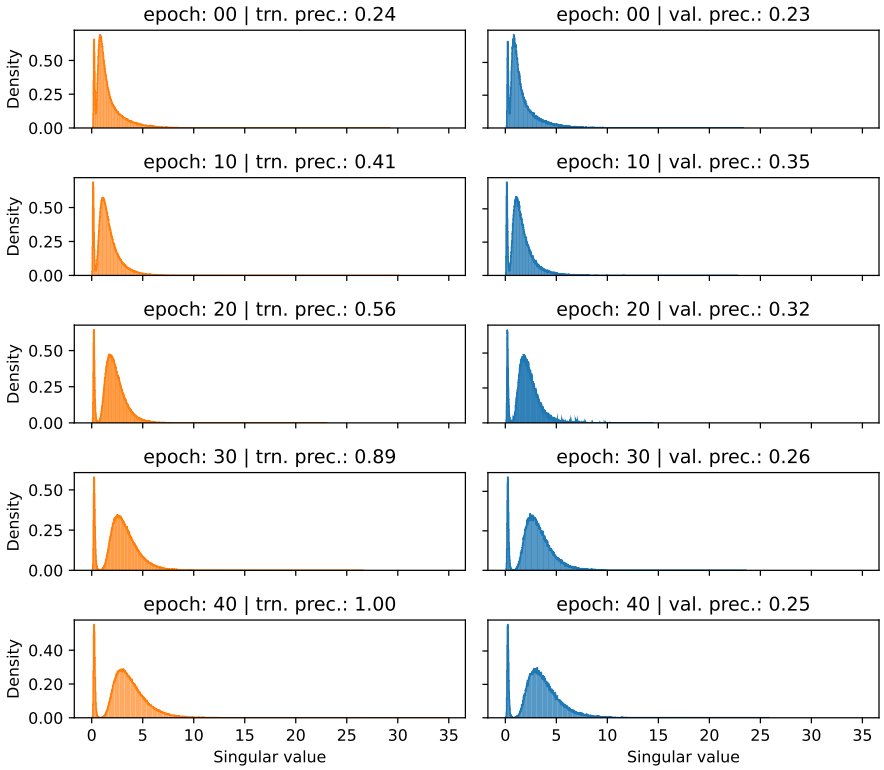

Fig. 10: Full point train (left) and validation (right) Jacobian spectrum for Inception trained on cifar-10, from first epoch to overfit, with label noise $p = 0.5$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

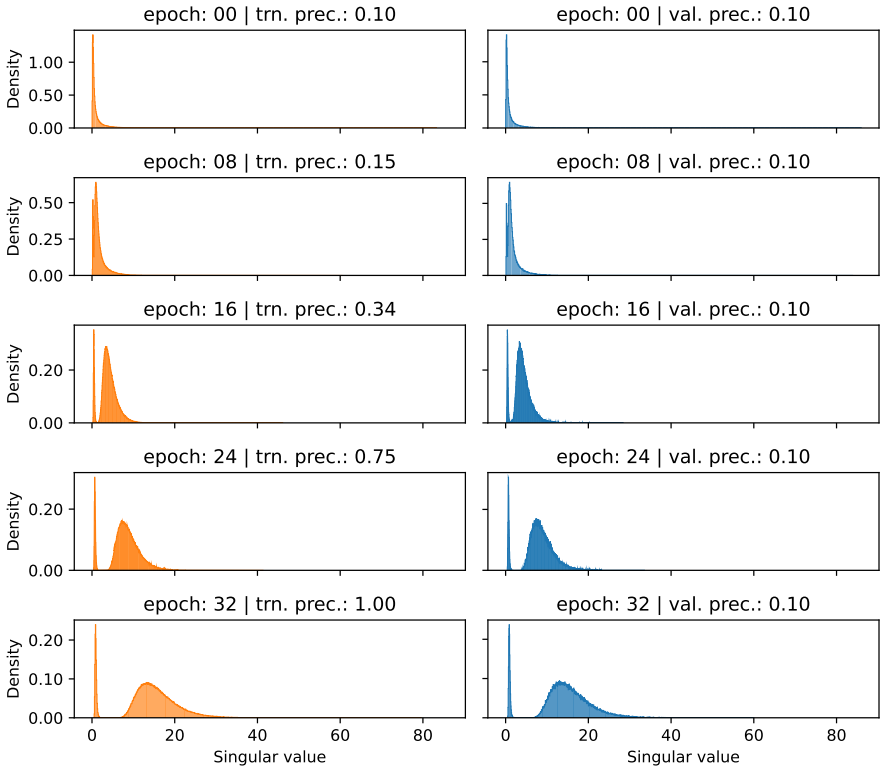

Fig. 11: Full point train (left) and validation (right) Jacobian spectrum for Inception trained on cifar-10, from first epoch to overfit, with label noise $p = 1.0$. Horizontal scale is the same across all plot in the same column. Epochs and performance are indicated on top of each graph.

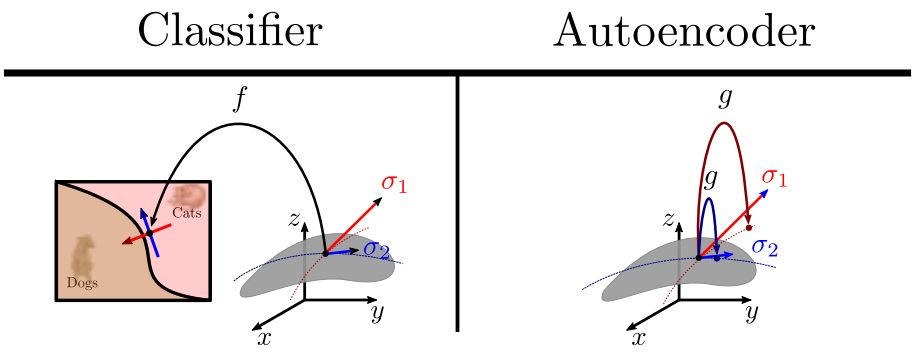

Fig. 12: Principal directions of the point Jacobian for a classifier f and an autoencoder g for three-pixel pictures of cats and dogs on the neighborhood of a given point. By definition, the norm of the change of the image through f for perturbations in first singular direction σ_1 is maximal among all directions, and similarly for the second direction in the orthogonal space to the first. Note that since the destination space is in $\mathbb{R}^2$, there are only two singular directions in the original $\mathbb{R}^3$. For each point P the two directions are the directions of respectively maximum and minimum change with respect to "cat-dog". As for the autoencoder, reconstruction is much more sensitive to perturbations along σ_1 than σ_2: changes along the latter are reconstructed as being the same image, which means that the model considers them as being noise.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `https://www.tensorflow.org/`, software available from tensorflow.org
2. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
3. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
4. Morcos, A.S., Barrett, D.G.T., Rabinowitz, N.C., Botvinick, M.: On the importance of single directions for generalization. CoRR (2018), `http://arxiv.org/abs/1803.06959v4`
5. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31 (2017)
6. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al.: Scipy 1.0: fundamental algorithms for scientific computing in python. Nature methods **17**(3), 261–272 (2020)
7. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (2017), `https://openreview.net/forum?id=Sy8gdB9xx`