The Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)

Understanding and Leveraging the Learning Phases of Neural Networks[*]

Johannes Schneider[1†], **Mohit Prabhushankar**[2]

[1]University of Liechtenstein, Vaduz, Liechtenstein
[2]Georgia Institute of Technology, Atlanta, USA
johannes.schneider@uni.li, mohit.p@gatech.edu

Abstract

The learning dynamics of deep neural networks are not well understood. The information bottleneck (IB) theory proclaimed separate fitting and compression phases. But they have since been heavily debated. We comprehensively analyze the learning dynamics by investigating a layer's reconstruction ability of the input and prediction performance based on the evolution of parameters during training. We empirically show the existence of three phases using common datasets and architectures such as ResNet and VGG: (i) near constant reconstruction loss, (ii) decrease, and (iii) increase. We also derive an empirically grounded data model and prove the existence of phases for single-layer networks. Technically, our approach leverages classical complexity analysis. It differs from IB by relying on measuring reconstruction loss rather than information theoretic measures to relate information of intermediate layers and inputs. Our work implies a new best practice for transfer learning: We show empirically that the pre-training of a classifier should stop well before its performance is optimal.

Introduction

Deep neural networks are arguably the key driver of the current boom in artificial intelligence(AI) in academia and industry. They achieve superior performance in a variety of domains. Still, they suffer from poor understanding, leading to an entire branch of research, i.e., explainability(XAI) (Meske et al. 2022), and to widespread debates on trust in AI within society. Thus, enhancing our understanding of how deep neural networks work is arguably one of key problems in ongoing machine learning research (Poggio, Banburski, and Liao 2020). Unfortunately, the relatively few theoretical findings and reasonings are often subject to rich controversy.

One debate surrounds the core of machine learning: learning behavior. Shwartz-Ziv and Tishby (2017) leveraged the information bottleneck(IB) framework to investigate the learning dynamics of neural networks. IB relies on measuring mutual information between activations of a hidden layer and the input as well as the output. A key qualitative, experimental finding was the existence of a fitting and compression phase during the training process. Compression is conjectured a reason for good generalization performance. It is frequently discussed in the literature (Geiger 2021; Jakubovitz, Giryes, and Rodrigues 2019). Such a finding can be considered a breakthrough in understanding deep neural networks. However, its validity has been disputed, i.e., Saxe et al. (2019) claimed that statements by Shwartz-Ziv and Tishby (2017) do not generalize to common activation functions. Today, the debate is still ongoing (Lorenzen, Igel, and Nielsen 2021). A key challenge is approximating the IB. This, makes a rigorous mathematical treatment very hard – even empirical analysis is non-trivial.

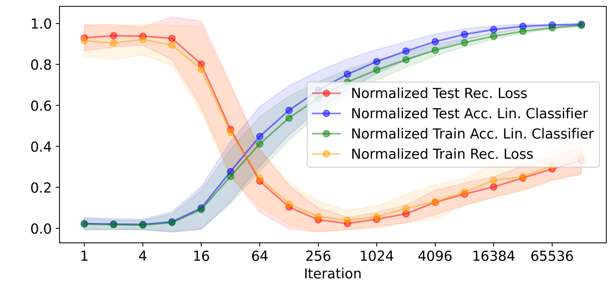


Figure 1: Normalized accuracy and reconstruction loss for a linear classifier and the FashionMNIST dataset

In this work, we study the learning behavior with a similar focus, i.e., classification and reconstruction capability of layers. We propose a different lens for investigation rooted in classical complexity analysis that leads to more precise statements. We perform a rigorous analysis of a linear classifier and a data model built on empirical evidence. First, to the best of our knowledge prior work relies on general statements that lack mathematical proof. Instead, we show bounds on the duration of phases, also highlighting interplays among data characteristics, e.g., between the number of samples, the number of classes and the number of input attributes. Second, we utilize different measures from IB. IB measures information of a layer with respect to input and output (Shwartz-Ziv and Tishby 2017). We measure (i) accuracy, e.g., how well the network classifies samples, and (ii) the reconstruction error of the input given the layer by utilizing a decoder.Third, we provide a data model and prove for single-layer networks, i.e., linear networks, that it can cause the three learning phases, i.e., the reconstruction error is likely to decrease early in training after a phase of near constant behavior and increases

[*]Code/Proofs: https://github.com/JohnTailor/LearnPhase
[†]Work conducted during holidays.

later during training.

As practical implication, we show that pre-training of classifiers that are later fine-tuned should stop well before the performance is optimal for the pre-training task. That is, we argue and show that it is decisive how much information on the original dataset is kept.

The paper begins with an empirical analysis showing that the alleged phases can be observed for multiple classifiers, datasets and layers followed by a theoretical analysis drawing on our empirical findings to model the problem and rigorously analyze a linear multi-class model. Finally, we elaborate on transfer learning, state related work, discuss our work, and conclude.

Empirical Analysis

For a model $M = (L_0, L_1, ...)$ consisting of a sequence of layers L_i, we investigate the reconstruction loss of inputs of a decoder $DE^{(t)}$ trained for each iteration t. The decoder is trained to output a reconstruction $\hat{x}$ from a layer activation $L_i(x)$, i.e., $\hat{X} = DE(L_i(x))$. The reconstruction error $Rec^{(t)}$ is $||\hat{x} - x||^2$. We compare this error against the model's accuracy $Acc^{(t)}$. (In the full version, we also discuss accuracy of a linear classifier CL trained on layer activations L_i.)

Datasets, Networks and Setup

As networks M we used VGG-11 (Simonyan and Zisserman 2014), Resnet-10 (He et al. 2016) and a fully connected network, i.e., we employed networks $F0$, which equals the theoretical setup, i.e., one dense layer followed by a softmax activation. After each hidden layer we applied the ReLU activation and batch-normalization. We used a fixed learning rate of 0.002 and stochastic gradient descent with batches of size 128 training for 256 epochs.

We computed evaluation metrics $Acc^{(t)}$ and $Rec^{(t)}$ at iterations 2^i. For the decoder DE we used the same decoder architecture as in (Schneider and Vlachos 2021), where a decoder from a (standard) auto-encoder was used. For each computation of the metrics, we trained the decoder for 30 epochs using the Adam optimizer with a learning rate of 0.0003.We reconstructed from the network outputs, i.e., the last dense layer (index -1), the second last layer (index -2), i.e., the one prior to the (last) dense layer and layer with index -3, which indicates using outputs of the second last conv layer for VGG and the second last block for ResNet.[1] We used CIFAR-10/100 (Krizhevsky and Hinton 2009), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), and MNIST (Deng 2012), all scaled to 32x32, available under the MIT (first 3 datasets) and GNU 3.0 license. We trained each model M five times. All figures show standard deviations. We report normalized metrics to better compare $Acc^{(t)}$ and $Rec^{(t)}$.

Observations

Figure 1 shows results for FashionMNIST using a linear classifier $F0$. Figure 2 shows the outputs for a ResNet for multiple layers for the MNIST and FashionMNIST datasets. Additional datasets and classifiers are in the full version. The behavior of all three classifiers across datasets is qualitatively identical. As expected, accuracy increases throughout training. The reconstruction loss remains stable for the first few iterations before decreasing and increasing towards the end, highlighting the existence of multiple phases. For layers closer to the input, phases become less pronounced and reconstruction loss overall is less. That is, the phases are well-observable when normalizing each line (left panel in 2, but less so for lower layers as seen in the right panel, where lines are not normalized separately. Put differently, the closer to the output, the more easily the phases are observable. For layers close to the input they are not noticeable. This is aligned with existing knowledge that lower layers in classifiers converge faster and are more generic(Zeiler and Fergus 2014), i.e., fit less to input data (especially, its labels). Thus, empirically, we have observed the existence of the phases, which we investigate more profoundly in our theoretical analysis.

Theoretical Analysis

We analyze the reconstruction loss over time of a linear decoder taking outputs of a simple classifier as input. We follow standard complexity analysis from computer science using O-notation deriving bounds regarding the number of samples n, dimension d of the inputs, and number of classes l. As common, we assume the quantities in the bounds such as n, d, and l are large, allowing us to discard lower order terms in n, d, and l and constants.

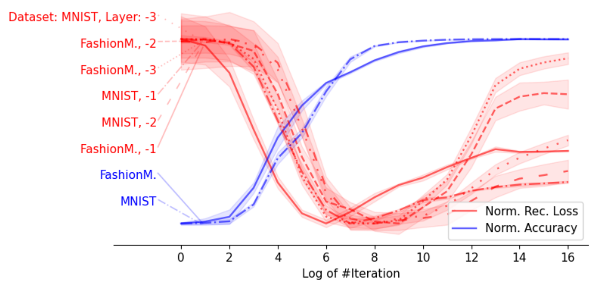
Definitions and Assumptions

Data: We consider a labeled dataset $D = \{(x, y)\}$ consisting of pairs (x, y) with d-dimensional input $x = (x_0, x_1, .., x_{d-1})$ and label $y \in [0, l-1]$, i.e., we have l classes and each input has d attributes. We denote $n = |D|$ as the number of samples. The set $C^y = \{(x|(x, y') \in D) \wedge (y' = y)\}$ comprises of all inputs of class y. We assume balanced classes, i.e., $|C^i| = |C^j| = n/l$. We denote the set of indexes for all d input attributes as $A := (0, 1, ..., d-1)$. For a sample $(x, y) \in D$ of a class y, only the subset of attributes $A_y := [y \cdot d/l, (y+1) \cdot d/l] \subset A$ are non-zero. This builds on the assumption that features are only present for at most a few classes.

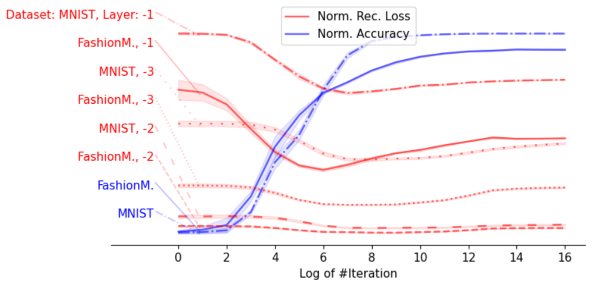
Our data builds on the natural assumption that some samples are easier and others are more difficult to recognize, i.e., input features have different strengths. That is, some samples might appear like prototypical samples with well-notable characteristics of that class, i.e., *strong features* and others might appear more ambiguous, i.e., have *weak features*. Rather than using a continuous distribution for feature strengths, we employ a discrete approximation, where a feature strength is either 0, 1 (weak) or k (strong). That is, feature strengths differ by a factor k. We assume $k \geq 2$.[2] We treat it as a constant. That is, although we do not subsume it in our asymptotic analysis, it should be seen as a constant, i.e., $\Theta(k) = \Theta(1)$. We say that samples C^y of class y can be split into two equal-sized disjoint subsets, i.e., *weak samples*

[1]We don't show reconstructions from softmax outputs as they follow the pattern even more strongly.

[2]$k > 1$ suffices with a more complex analysis.



(a) ResNet-10 (each line normalized separately)



(b) ResNet-10

Figure 2: Accuracy and reconstruction loss for a ResNet for the FashionMNIST and the MNIST dataset. (Negative) layer indices indicate skipped layers from the output as described in text. Other datasets and classifiers are in the extended version.

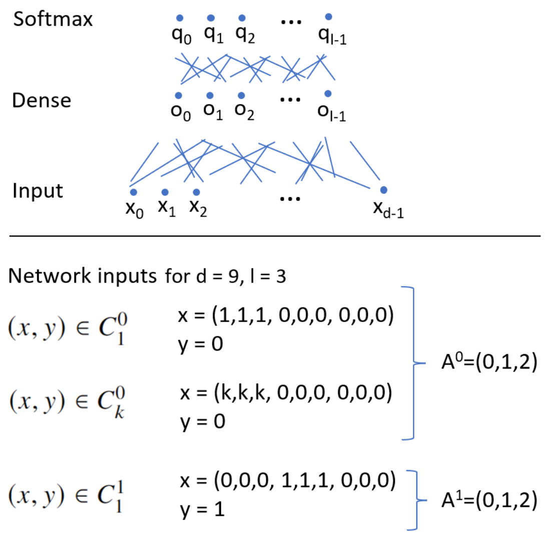

Softmax
Dense
Input
q_0 q_1 q_2 ... q_{l-1}
o_0 o_1 o_2 ... o_{l-1}
x_0 x_1 x_2 ... x_{d-1}
Network inputs for d = 9, l = 3
$(x, y) \in C_1^0$
x = (1,1,1, 0,0,0, 0,0,0)
y = 0
$(x, y) \in C_k^0$
x = (k,k,k, 0,0,0, 0,0,0)
y = 0
A⁰=(0,1,2)
$(x, y) \in C_1^1$
x = (0,0,0, 1,1,1, 0,0,0)
y = 1
A¹=(0,1,2)

Figure 3: Illustration of network with all to all connections and inputs

with only *weak features* C_1^y, and *strong samples* with only *strong features* C_k^y.[3]

The data samples (x, y) are defined as:

$$x := \begin{cases} x_{i \in A_y} = 1, & \text{if } x \in C_1^y \text{ (weak features for weak samples)} \\ x_{i \in A_y} = k, & \text{if } x \in C_k^y \text{ (strong features for strong samples)} \\ x_i = 0 & \text{otherwise (non-present features for both)} \end{cases} \tag{1}$$

The data are illustrated in Figure 3. In our data model, the value of an attribute (or feature) is zero for most samples. It is non-zero, i.e., either 1 or k, for samples of a single class. Also, the input attributes do not change their values throughout training. Real data, i.e., inputs to the last layer throughout training, is shown for comparison in Figure 4. For real data, the input distribution of one layer evolves during training as weights of lower layers change. For VGG-16, inputs stem from a ReLU layer and, thus, the majority of outputs is 0 as proclaimed in our model. For Resnet-10, the inputs stem from a 4x4 average pooling, which averages outcomes of a ReLU layer. Thus, there is no strong peak at 0, but rather inputs are small positive values. Despite the averaging after a few iterations, well-noticeable, class-dependent differences between attributes emerge. Thus, the essence of our model is also captured for Resnet: Most input attributes of (samples of) a class are small. An input attribute is only large for samples of one or a few classes.

Network: The network is illustrated in Figure 3. All parameters (and parameter-dependent entities such as outputs) are time-dependent, i.e., with the superscript $^{(t)}$. For readability, we omit the superscript in expressions, where all entities share the same time t.

The network output stems from a single dense layer followed by a softmax activation.

Definition 1 (Linear Layer Output)**.** The output for a sample x is $o(x) = (o_0(x), o_1(x), ..., o_{l-1}(x))$. The scalar o_y is the output for class y defined as $o_y := o_y(x) := w_y \cdot x = \sum_{i<d} w_{y,i} \cdot x_i$.

The softmax function to compute the class probability for a class y given the output $o(x)$ for a sample (x, y'') is:

$$q(y|x) = \frac{e^{o_y}}{\sum_{y'<l} e^{o_{y'}}} \tag{2}$$

If all elements x in a set C are identical we write as an abbreviation:

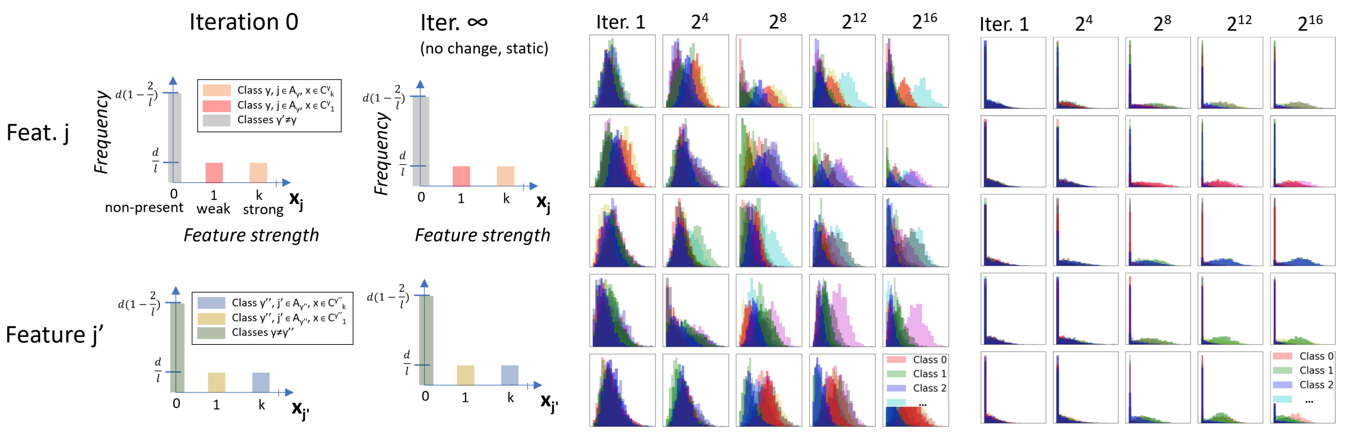
$$q(y|C) := q(y|x \in C) \tag{3}$$
$$o(C) := o(x \in C) \tag{4}$$

Assumption 2 (Weight Initialization)**.** $w_{y,j}^{(0)} \sim N(0, 1/d)$ (random), $w_{y,j}^{(0)} = 1/d$ (deterministic)

For random initialization, initial noise becomes much smaller in magnitude relative to the changes in weights during training. This diminishes its impact over time. Thus, we first assume deterministic initialization for simplicity and analyze random initialization in the extended version. The random initialization follows common initialization schemes (He et al. 2015; Schneider 2022).

Cross-entropy Loss and Optimization: We employ the cross-entropy loss for a sample (x, y) defined as $L(x, y) = \sum_{y'<l} -\mathbb{1}_{y=y'} \cdot \log(q(y'|x))$. The indicator variable $\mathbb{1}_{cond}$

[3]Samples having a mix of both do not change outcomes, but add to notational complexity



(a) Data model (b) Real data: ResNet-10 on Fashion-MNIST. (c) Real data: VGG-16 on CIFAR-10

Figure 4: Class-dependent distribution of an input attribute fed into the last dense layer. Rows show features, columns iterations. Initially, inputs are equally distributed for all classes, but already after a few iterations samples of one or a few classes have larger values. ResNet is less concentrated due to 4x4 average pooling.

evaluates to 1 if the condition *cond* holds and otherwise to 0. We perform gradient descent. The update of weight $w^{(t)}_{y,j}$ at iteration t is

$$w^{(t+1)}_{y,j} = w^{(t)}_{y,j} - \frac{\lambda}{|D|} \sum_{(x,y')\in D} \nabla_{w_{y,j}} L(x,y') \tag{5}$$

A linear classifier can perfectly classify the data (Def. 1): Weights $w_{y,i}$ for $i \in A_y$ should be very large since the loss tends to 0 as these weights tend towards ∞. All other weights should be very small since the loss tends to 0 as these weights tend towards -∞.

Reconstruction: A linear reconstruction (or decoding) function g_j is defined for each input attribute j.

Definition 3 (Reconstruction Function)**.** The reconstruction function g_j takes as input the output vector $o(x)$ for a sample (x, y) and learnable parameters are a bias b and a slope $s_{y'}$ for each class y', i.e., each scalar $o_{y'}$ in vector $o(x)$:

$$g_j(o(x)) := b_j + \sum_{y'<l} s_{y',j} \cdot o_{y'} \tag{6}$$

Note, like the parameters $w^{(t)}$ of the classifier, the parameters $b^{(t)}$ and $s^{(t)}$ of the reconstruction function are not fixed throughout training. They are fit for each iteration t. That is, $g^{(t)}_j$ also depends on time. The reconstruction loss for an attribute x_j of a sample (x, y) is the squared difference between x and the reconstructed inputs g_j. The total reconstruction loss $R^{(t)}$ is just the average of the individual losses.

$$R^{(t)} := \sum_{(x,y)\in D} \frac{(x_j - g^{(t)}_j(o(x)))^2}{|D|} \tag{7}$$

Analysis Outline

Full details are given in the extended version. Here we only provide an outline. We formally show the existence of two phases, i.e., that the reconstruction loss decreases and then increases again. To this end, we split the analysis into three stages based on sums of weights f, which are linked to iterations t. We show that the error decreases in Stage 1, starts to increase again in Stage 2 and stabilizes in Stage 3. Due to symmetry, it suffices to focus on one class y and aim to fit four (weighted) points as shown in Figure 5. In fact, we use an approximate reconstruction function and show that it is not far from the optimal. Our approach relies on heavy calculus, in particular (Taylor) series expansions, and classical O-notation to keep the complexity of the analysis manageable.

To get a flavor, we derive an expression for the change of weights in each iteration, which is further expanded during the analysis. To this end, we leverage symmetries arising from the data definition and the deterministic initialization. We compute derivatives for weights $w_{y,j}$ for a sample (x, y') separately for the class $y = y'$ and $y \neq y'$, and for samples of different strengths, i.e., $(x, y') \in C^y_v$ with $v \in \{1, k\}$. The derivative $\frac{\partial L(x,y)}{\partial w_{y',j}}$ of the loss with respect to network parameters is non-zero if inputs are non-zero, i.e., $x_j \neq 0$, which only holds for $j \in A_y$ (Def. 1 and Figure 3). For derivative of loss with a softmax holds (see Section 5.10 in (Jurafsky and Martin 2021)):

$$\frac{\partial L(x,y)}{\partial w_{y',j}} = \frac{\partial L(x,y)}{\partial o_{y'}} \cdot \frac{\partial o_{y'}}{\partial w_{y',j}} = (q_{y'|x} - \mathbb{1}_{y'=y}) \cdot x_j \tag{8}$$

$$= \begin{cases} v(q(y'|x) - \mathbb{1}_{y=y'}) & \text{if } j \in A_y, x \in C^y_v \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Next, we rephrase the reconstruction loss (Eq. 7), which sums the error across all samples. Due to symmetry, it suffices to focus on one attribute x_j with $j \in A_y$ for some y. There are also only four different reconstruction errors due to symmetry:

1. R_1 states the error for reconstructing $x_j = 1$ with $j \in A_y$ given the output for a sample $x \in C^y_1$

2. R_k for $x_j = k$ with $x \in C_k^y$
3. $R^{(t)}_{0,C_1^{y'}}$ for $x_j = 0$ with $x \in C_1^{y \neq y'}$
4. $R^{(t)}_{0,C_k^{y'}}$ for $x_j = 0$ with $x \in C_k^{y \neq y'}$

Formally, these errors and the total error R are given by:

$$R^{(t)}_{v,C} := \sum_{x \in C, x_j = v} \frac{(x_j - g_j^{(t)}(o(x)))^2}{|C|} \tag{10}$$

$$R_1^{(t)} := R^{(t)}_{1,C_1^y} \qquad R_k^{(t)} := R^{(t)}_{k,C_k^y} \qquad \text{(Abbreviation)} \tag{11}$$

$$R^{(t)}_{0,C_1^{y'}} := R^{(t)}_{1,C_1^{y' \neq y}} \qquad R^{(t)}_{0,C_k^{y'}} := R^{(t)}_{k,C_k^{y' \neq y}} \qquad \text{(Abbreviation)} \tag{12}$$

$$R^{(t)} = \big((1 - \frac{1}{2l}) \cdot (R^{(t)}_{0,C_1^{y'}} + R^{(t)}_{0,C_k^{y'}}) + \frac{1}{l} \cdot (R_1^{(t)} + R_k^{(t)})\big) \tag{13}$$

That is, the total reconstruction error of an attribute $j \in A_y$ is the weighted sum of errors for $x_j = 0$, which are most prevalent with a weight of $1 - \frac{1}{2l}$, and the reconstruction errors for $x_j = 1$ and $x_j = k$, which are less common, i.e., only a fraction $\frac{1}{l}$ of samples has $x_j = 1$ or $x_j = k$ for $j \in A_y$.

We aim to reconstruct input attributes $x_j \in \{0, 1, k\}$ from output probabilities $q^{(t)}(y'|x)$ optimally for each time t. We choose non-optimal but simpler reconstruction functions depending on t and bound the error due to the approximation. We show that these functions approximate the true reconstruction error asymptotically optimally. As approximation, we fit the reconstruction function g_j using two of the four output values q, i.e., the outputs for $x_j = k$ with $j \in A_y$ for $x \in C_k^y$ and for $x_j = 0$ for $x \in C_k^{y' \neq y}$ (see Figure 5 and the four errors listed in Def. 13). These two points are fitted optimally, i.e., without error, while the other two can have larger errors than the optimal reconstruction function g_j^{opt}. The motivation for the selection of the two specific points is as follows. We use $x_j = 0$ since most attributes x_j are 0, i.e., out of d attributes only a fraction $2/l$ are non-zero. Therefore, as d is assumed to be large, even small errors in reconstructing attributes $x_j = 0$ can yield large overall errors. The choice of $x_j = k$ rather than $x_j = 1$ is to have a larger spread between points used to fit the reconstruction function.

For $t > 0$, we use:

$$g_j(o) = \frac{k}{q(y|C_k^y) - q(y|C_k^{y' \neq y})} \cdot (o_j - q(y|C_k^{y' \neq y})) \tag{14}$$

Put differently, geometrically, we fit a line g_j without error through two points given as tuple (output q,value x_j to reconstruct), i.e., $(q(y|C_k^{y' \neq y}), 0)$ and $(q(y|C_k^y), k)$ as illustrated in Figure 5. Thus, it follows from the construction of g_j that the reconstruction errors R_k (i.e., for $x_j = k$ and $x \in C_k^y$) and $R_{0,C_k^{y'}}$ (i.e., for $x_j = 0$ and $x \in C_k^{y' \neq y}$) are both zero.

For $t = 0$, where outputs q are identical for all inputs, we use a constant reconstruction function being a weighted average:

$$g_j^{(t)}(o) = \frac{\frac{n}{l} + \frac{n}{l}k + n(1 - \frac{1}{2l}) \cdot 0}{n} = \frac{1+k}{l} \tag{15}$$

Reconstruction Error: Next, we bound the error of the reconstruction function. We use different stages suitable for analysis. We first express them not using time t but in terms of sums of weights. Figure 5 illustrates the stages: Initially, all probabilities q are equal and the reconstruction error is large. During the first stage $q(y|C_k^y)$ increases rapidly for (strong) samples with strong features, reducing the initial reconstruction error. In the second phase, $q(y|C_k^y)$ (for strong samples) changes much less, while $q(y|C_1^y)$ for (weak) samples grows fast and catches up. Within this stage the reconstruction error increases again. In the third stage both $q(y|C_1^y)$ and $q(y|C_k^y)$ have roughly the same magnitude and converge towards 1. The reconstruction error still slowly increases but also converges. The fact that differences between the two diminish, worsens the reconstruction as shown in the rightmost panel in Figure 5, since the best reconstruction is in the middle of both well-separated points rather than being close to each of them.

We formally derive each reconstruction error for each stage separately. The derivation for each error and stage follows the same schema. We simplify the reconstruction errors R_1 and $R_{0,C_1^{y' \neq y}}$ using series expansions. We do not derive a single expression for a reconstruction error for all iterations t but rather split the analysis by looking at intervals of weight values, i.e., their sums f. These are then linked to iterations t. Considering intervals constraining the sum of weights allows to further simplify expressions. Still, a significant amount of calculus is required to obtain closed-form expressions. We obtain bounds for each individual reconstruction error (see Eq. 13), combining all individual reconstruction errors yielding the total error based on our approximate reconstruction function g_j:

Corollary 4. *The reconstruction error for g_j decreases from $R^{(0)} = \Omega(k^2/l)$ to $R^{(t)} = O(k^2/l^2)$ with $t \in [1, \Theta(\frac{2l(c_f - 1)}{\lambda d(k+1)})]$ for an arbitrary constant $c_f > 1$ and increases to $R^{(t \to \infty)} = \Omega(k^2/l)$.*

But our proclaimed reconstruction function g_j is not optimal. Thus, finally, we need to bound the deviation of our reconstruction error based on g_j from the optimal reconstruction error. This can be best understood based on our illustration Figure 5, where the optimal reconstruction g^{opt} line might not go through two points exactly as g_j does, but rather be more in the "middle" of all points.

Theorem 5. *For the approximation error of the reconstruction function g_j of the optimal function g^{opt} holds $||g_j - g_j^{opt}|| < 2$.*

As proof strategy, we consider the individual terms R_i that sum to the total reconstruction error R. Each R_i represents the error of one set of points to reconstruct. We show that changing the points for reconstruction allows reducing the most dominant error R_i only up to a constant factor before another error R_j becomes dominant. Thus, asymptotically our approximation is optimal.

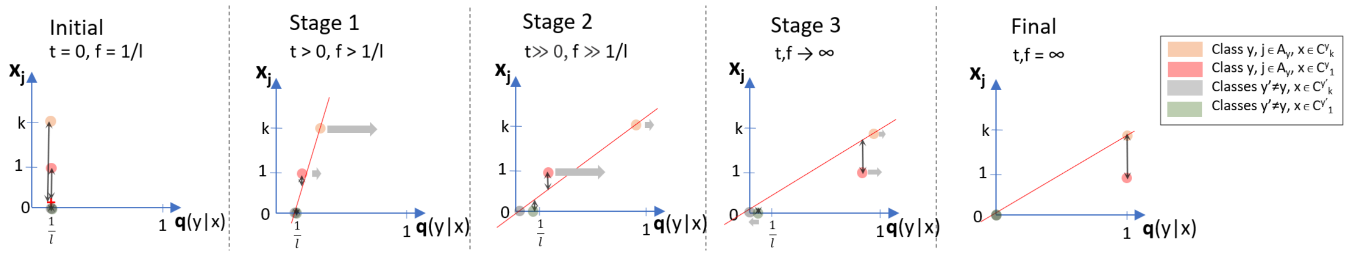


Figure 5: Stages distinguished for analysis. Stages are separated based on sums of weights (f), which are linked to iterations t. Each panel shows network outputs q (blue points) versus attributes x_i for samples $(x, y) \in D$ and the (approximate) reconstruction function of x_i from q (red line) over time t. The reconstruction approximates the optimal reconstruction. Vertical bars indicate error bars. Grey horizontal arrows shows how outputs q change compared to the panel on the left.

Improved Transfer Learning

Our work shows that while the discriminative performance of networks increases, their ability to accurately describe the data (as measured by reconstruction ability) decreases. Thus, if a classifier is used for fine-tuning on a dataset D' or as feature extractor that should describe the data well, it might be better to stop training of the classifier C on the (original, large) dataset D before the cross-entropy loss is minimal, i.e., before the classifier performance is maximized on D. When exactly to stop depends on the similarity of the datasets D and D', i.e., it might not be necessarily when the reconstruction loss is minimal.
To assess this hypothesis, we proceed analogously as for reconstruction (see Section for details). We train a classifier C, i.e., VGG or Resnet, on a dataset D. We then freeze the classifier C and train a linear classifier on a dataset D' taking as input the output of a layer (last (-1), second last(-2), etc.) of the classifier C. More precisely, for dataset D being FashionMNIST, we use D' being MNIST and for D being CIFAR-10 we use CIFAR-100 as D'. We also assess the scenarios with D and D' switched. Furthermore, we also assess to predict, which color channel, i.e., 'red','green','blue', has largest average value used across all pixels. Figures 6 and 7 show two exemplary outputs. It can be seen that accuracy on the downstream have a clear maximum, which tends to be roughly after the same iterations when the reconstruction loss is minimal. Thus, we see that some maintaining more "information" on the original dataset (as observed due to lower reconstruction loss) is helpful for downstream task, since these tasks might exactly rely on this information.

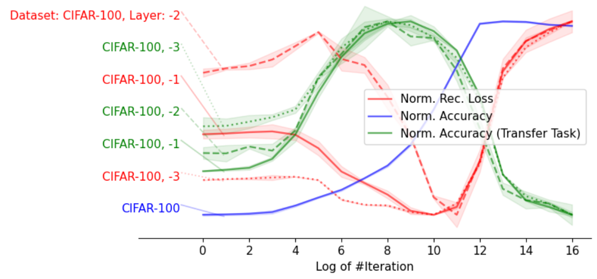


Figure 6: Transfer Learning of a Resnet from Cifar-100 to Cifar-10. Lines are normalized.

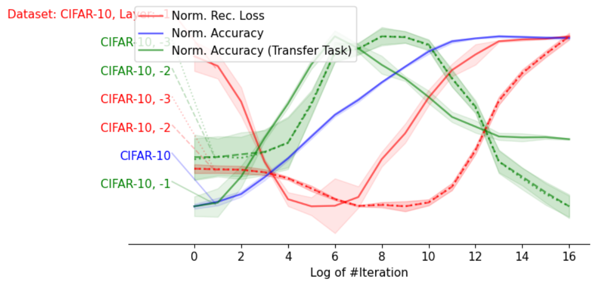


Figure 7: Transfer Learning of VGG from predict Cifar-10 classes to predict the most dominant color channel. Lines are normalized.

Figure 6 shows the normalized performance of the original classifier, the reconstruction loss and the accuracy for the fine-tuning task.

Discussion

We have shown empirically and analytically the existence of different phases for the reconstruction error. Our analysis is fairly different to prior works and, as such, interesting in its own right. That is, we aim to focus on approximating the governing optimization equations, which allows to capture the reconstruction dynamics over time. An essential part of our work are the underlying assumptions on which the theoretical analysis builds. While we verified our assumptions with empirical analysis (Figure 4), any assumption is a limitation. Furthermore, our analysis relies on shallow networks, i.e., we essentially analyze the last dense layer and the softmax layer, which already requires multiple pages of calculus. Approaches as described in (Maier et al. 2019) could help in extending it to deeper networks, but it is unclear, whether the analysis would reamin tractable. On the positive side, we derive concise bounds including multiple network parameters and covering also the number of iterations. This is often foregone in other works that focus on learning dynamics, e.g., they might require that width of layers tend to infinity to approximate layers with distribution(Jacot, Gabriel, and Hongler 2018; Huang and Yau 2020).

While we have shown empirically that the three phases of the reconstruction loss can occur for multiple datasets, networks, and layers, they are most notable for the last layers of a network. Features in the last layers are the least shared among classes, since the softmax layer paired with the cross-entropy loss of the classifier enforces discriminative features, i.e., features that strongly correlate with one or few classes only. That is, during training outputs of the last linear layer become more and more discriminative, forgoing information on the sample. The softmax also reduces the nuanced differences between classes. Thus, information content on the appearance is lost or difficult to retrieve in upper layers. This effect has also been observed in (Saxe et al. 2019) using mutual information through simulations rather than mathematical proofs. This behavior can also be observed for other loss functions like the hinge loss as shown empirically in the extended version. We anticipate that these phases are also prevalent in other architectures like RNNs and transformers, since the key assumptions such as the existence of weak samples (weak feature strenghts) and strong samples is not architecture dependent.

Related Work

The information bottleneck (Tishby, Pereira, and Bialek 2000) was used for analysis of deep learning (Tishby and Zaslavsky 2015; Shwartz-Ziv and Tishby 2017). It suggests a principled way to "find a maximally compressed mapping of the input variable that preserves as much as possible the information on the output variable" (Shwartz-Ziv and Tishby 2017). To this end, layers h_i of a network are viewed as a Markov chain for which holds given $i \geq j$ using the data processing inequality: $I(Y;X) \geq I(Y;h_j) \geq I(Y;h_i) \geq I(Y;\hat{Y})$ Learning is seen as the process that maximizes $I(Y;h_i)$ while minimizing $I(h_{i-1};h_i)$, where the latter can be interpreted as the minimal description length of the layer. In our view, we agree on the former (at least on a qualitative level), but we do not see minimizing the description length as a goal of learning. In our perspective, it can be a consequence of the first objective, i.e. to discriminate among classes, and existing learning algorithms, i.e., gradient descent. From a generalization perspective, it seems preferable to cling onto even the smallest bit of information of the input X, even if it is highly redundant and, as long as it *could* be useful for classification. This statement is also supported by (Saxe et al. 2019) who show that compression is not necessary for generalization behavior and that fitting and compression happen in parallel rather than sequentially. The review (Geiger 2021) also concludes that the absence of compression is more likely to hold. In contrast to our work, their analysis is within the IB framework. Still, it remedies an assumption of Shwartz-Ziv and Tishby (2017). Namely, Saxe et al. (2019) investigates different non-linearities, i.e., the more common ReLU activations rather than sigmoid activations. Lorenzen, Igel, and Nielsen (2021) argues that compression is only observed consistently in the output layer. The IB framework has also been used to show that neural networks must lose information (Liu et al. 2020) irrespective of the data it is trained on. From our perspective, the alleged information loss measured in terms of the reconstruction capability is a fact though it can be small. In particular, it is evident that at least initially reconstruction is almost perfect for wide networks following theory on random projections, i.e., the Johnson-Lindenstrauss Lemma (Johnson and Lindenstrauss 1984) proved that random projections allow embedding n points into an $O(\log n/\epsilon^2)$ dimensional space while preserving distances within a factor of $1 \pm \epsilon$. This bound is also tight according to Larsen and Nelson (2017), and it can be extended to cases where we apply non-linearities, i.e., $ReLU$. Wang et al. (2021) used an information measure based on weights to show empirically that fitting and compression phase exist. Achille, Rovere, and Soatto (2018) suggest two phases based on empirical analysis. They call the second phase "forgetting".

Geiger (2021) also discussed the idea of geometric compression based on prior works on IB analysis. However, the literature was inconclusive according to Geiger (2021) on whether compression occurs due to scaling or clustering. Our analysis is inherently geometry (rather than information) focused.

Alain and Bengio (2017) used a single linear layer to analyze networks empirically. However, their focus was to understand the suitability of intermediate layers for classification rather than learning dynamics. As the IB has also been applied to other types of tasks, i.e., autoencoding (Tapia and Estévez 2020), we believe that our approach might also be extended to such tasks. The idea to reconstruct inputs from layer activations has been outlined in the context of explainability (Schneider and Vlachos 2021, 2022). The idea is to compare reconstructions using a decoder with original inputs to assess what information (or concepts) are "maintained" in a model. Our work also touches upon linear decoders that have been studied extensively, e.g., (Kunin et al. 2019). It also estimates reconstruction errors from noisy inputs $x_i + \epsilon$ (Carroll, Delaigle, and Hall 2009).

Dynamics of learning have been studied using Neural Tangent Kernel (NTK) (Jacot, Gabriel, and Hongler 2018; Huang and Yau 2020). NTK focuses on infinite-width networks by showing that they are equal to Gaussian processes. For example, it has been shown that fully connected networks of a particular size show linear rate convergence towards zero training error (Corollary 2.5 in (Huang and Yau 2020)). However, they do not discuss the relationship between reconstruction and classification, which is the focus of our work.

The impact of loss functions on transfer learning was studied in (Kornblith et al. 2021). Aligned with our work it was found that models performing better on the pre-training task can perform worse on downstream tasks. Diversity of features(Nayman et al. 2022) has also been shown to lead to better downstream performance. This is also aligned with our work, since more diversity is also likely meaning that more information is captured, i.e., reconstructions are better. In contrast to these works, we proclaim that it is essential how much information on the input is maintained.

Conclusions

Theory of deep learning is limited. This work focused on a very pressing problem, i.e., understanding the learning process. To this end, it rigorously analyzed a simple dataset modeling empirical observations of common datasets and classifiers. Our results highlight the existence of multiple phases for the reconstruction loss. This insight can be used to improve transfer-learning using early stopping of pre-training.

References

Achille, A.; Rovere, M.; and Soatto, S. 2018. Critical learning periods in deep networks. In *International Conference on Learning Representations*.

Alain, G.; and Bengio, Y. 2017. Understanding intermediate layers using linear classifier probes. In *International Conference on Learning Representations ICLR (Workshop)*.

Carroll, R. J.; Delaigle, A.; and Hall, P. 2009. Nonparametric prediction in measurement error models. *Journal of the American Statistical Association*, 104(487): 993–1003.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6): 141–142.

Geiger, B. C. 2021. On Information Plane Analyses of Neural Network Classifiers–A Review. *IEEE Transactions on Neural Networks and Learning Systems*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. of the international conference on computer vision*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Conference on computer vision and pattern recognition (CVPR)*, 770–778.

Huang, J.; and Yau, H.-T. 2020. Dynamics of deep neural networks and neural tangent hierarchy. In *International conference on machine learning*, 4542–4551. PMLR.

Jacot, A.; Gabriel, F.; and Hongler, C. 2018. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Jakubovitz, D.; Giryes, R.; and Rodrigues, M. R. 2019. Generalization error in deep learning. In *Compressed Sensing and Its Applications*, 153–193.

Johnson, W. B.; and Lindenstrauss, J. 1984. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26.

Jurafsky, D.; and Martin, J. H. 2021. Speech and Language Processing. *Draft of 3rd edition*.

Kornblith, S.; Chen, T.; Lee, H.; and Norouzi, M. 2021. Why do better loss functions lead to less transferable features? *Advances in Neural Information Processing Systems*, 34: 28648–28662.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report.

Kunin, D.; Bloom, J.; Goeva, A.; and Seed, C. 2019. Loss landscapes of regularized linear autoencoders. In *International Conference on Machine Learning*, 3560–3569.

Larsen, K. G.; and Nelson, J. 2017. Optimality of the Johnson-Lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, 633–638. IEEE.

Liu, Y.; Qin, Z.; Anwar, S.; Caldwell, S.; and Gedeon, T. 2020. Are deep neural architectures losing information? invertibility is indispensable. In *International Conference on Neural Information Processing*, 172–184. Springer.

Lorenzen, S. S.; Igel, C.; and Nielsen, M. 2021. Information Bottleneck: Exact Analysis of (Quantized) Neural Networks. *arXiv preprint arXiv:2106.12912*.

Maier, A. K.; Syben, C.; Stimpel, B.; Würfl, T.; Hoffmann, M.; Schebesch, F.; Fu, W.; Mill, L.; Kling, L.; and Christiansen, S. 2019. Learning with known operators reduces maximum error bounds. *Nature machine intelligence*, 1(8): 373–380.

Meske, C.; Bunde, E.; Schneider, J.; and Gersch, M. 2022. Explainable artificial intelligence: objectives, stakeholders, and future research opportunities. *Information Systems Management*, 39(1): 53–63.

Nayman, N.; Golbert, A.; Noy, A.; Ping, T.; and Zelnik-Manor, L. 2022. Diverse Imagenet Models Transfer Better. *arXiv preprint arXiv:2204.09134*.

Poggio, T.; Banburski, A.; and Liao, Q. 2020. Theoretical issues in deep networks. *Proceedings of the National Academy of Sciences*, 117(48): 30039–30045.

Saxe, A. M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B. D.; and Cox, D. D. 2019. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12): 124020.

Schneider, J. 2022. Correlated Initialization for Correlated Data. *Neural Processing Letters*, 1–18.

Schneider, J.; and Vlachos, M. 2021. Explaining neural networks by decoding layer activations. In *International Symposium on Intelligent Data Analysis*, 63–75.

Schneider, J.; and Vlachos, M. 2022. Explaining classifiers by constructing familiar concepts. *Machine Learning*, 1–34.

Shwartz-Ziv, R.; and Tishby, N. 2017. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *Int. Conference on Learning Representations (ICLR)*.

Tapia, N. I.; and Estévez, P. A. 2020. On the information plane of autoencoders. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Tishby, N.; Pereira, F. C.; and Bialek, W. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Tishby, N.; and Zaslavsky, N. 2015. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop (ITW)*, 1–5.

Wang, Z.; Huang, S.-L.; Kuruoglu, E. E.; Sun, J.; Chen, X.; and Zheng, Y. 2021. PAC-Bayes Information Bottleneck. *arXiv preprint arXiv:2109.14509*.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*.