# Lecture 01: Introduction to Neural Network Theory, Failures of Classical Theory

Geoff Pleiss

Neural networks are hierarchical models where each layer produces learned basis functions by applying a simple non-linear transformation to linear combinations of outputs from the previous layer. The simplest neural network, a **multi-layer perceptron (MLP)**, follows the functional form:

$$\begin{gathered} f_{\text{NN}}(\boldsymbol{x}) = \sum_{i=1}^{d} \beta_i \phi_i^{(L)}(\boldsymbol{x}) \\ \left\{ \phi_i^{(\ell)} = \sigma\left( \textstyle\sum_{j=1}^{d} w_{ij}^{(\ell)} \phi_j^{(\ell-1)}(\boldsymbol{x}) \right) \right\}_{\ell=1}^{L} \\ \phi_i^{(0)} = x_i. \end{gathered} \tag{1}$$

Here, there are $L$ layers, each of which has $d$ units. The $\beta_i$ and $w_{ij}^{(\ell)}$ are the *learned parameters* of the network. The function $\sigma : \mathbb{R} \to \mathbb{R}$ is the *non-linear activation function* that enables neural networks to learn complex non-linear functions. Most neural networks use the rectified linear unit (ReLU) function $\sigma(x) = \max(0, x)$.

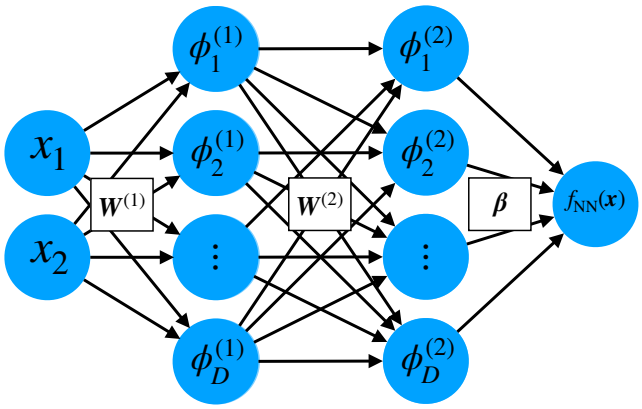



Figure 1: Graphical depiction of a MLP with two hidden layers.

---

## 1) Why Neural Networks are Popular

Neural networks were historically motivated as a crude approximation to human brains. More recently, researchers have instead pivoted to a less biological perspective to their success: neural networks are powerful function approximators that learn hierarchical representations of data. Intuitively, the first layer of a image neural network learns simple features that recognize edges and textures, the second layer composes these features to recognize more complex shapes, and so on, leading to a final layer that can recognize complex objects.

The simple answer as to why neural networks are popular is because of their empirical success. Over the past 13 years, they have become the de facto standard machine learning algorithm for text data, image data, graph data, and much more.

However, many statisticians and machine learning researchers resisted the use of neural networks, as

classical theory strongly suggested that they wouldn't work. We will discuss two key failures of classical learning theory to explain the success of neural networks.

---

## 2) Failures of Classical Theory: Generalization

**Generalization** is the ability of a learning algorithm $\mathcal{A}$ to produce a model $\hat{f} : \mathcal{X} \to \mathbb{R}$ that makes accurate predictions on unseen data. It is often measured by **risk**:[1]

$$\mathcal{R}(\hat{f}) = \mathbb{E}\left[\ell(\hat{f}(\boldsymbol{x}), y)\right]$$

where $\ell(\hat{f}(\boldsymbol{x}), y)$ is a loss function measuring the discrepancy between the prediction $\hat{f}(x)$ and the true value $y$. (Note that expectation is taken with respect to all random quantities: the test input $\boldsymbol{x}$, its associated label $y$, and the training data/randomness used to produce $\hat{f}$ from $\mathcal{A}$.) We often assume that the training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ are drawn i.i.d. from the same distribution as the test data.

For the past half century, the theoretical grounding of machine learning was centered on the the framework of **empirical risk minimization (ERM)**[2]—or, equivalently, the **probably approximately correct (PAC)** framework. Essentially, the ERM principle justifies the use of the model $f$ within some function space/hypothesis class $\mathcal{H}$ that minimizes the **empirical risk** $\mathcal{R}_{\text{emp}}$, i.e. the training loss:

$$\mathcal{A} \to \hat{f} = \arg\min_{f \in \mathcal{H}} \mathcal{R}_{\text{emp}}(f) = \frac{1}{N} \sum_{i=1}^{n} \ell(f(\boldsymbol{x}_i), y_i) \tag{2}$$

Empirical risk minimization relies on a function space/hypothesis class $\mathcal{H}$ that admits a **uniform law of large numbers**. Unlike the standard LLN, a uniform LLM implies that any empirical average involving a function $f \in \mathcal{H}$ converges to the true average *and the rate of convergence is bounded for all* $f \in \mathcal{H}$. In other words, the empirical risk $\mathcal{R}_{\text{emp}}(\hat{f})$ measured on the training data will eventually converge to the true test error of $\hat{f}$, *even though* $\hat{f}$ *is chosen using the training data!* And moreover, since $\hat{f}$ minimizes the loss over the training data, as $n \to \infty$ (and thus the training data distribution converges to the true distribution), we have that $\hat{f} \to \arg\min_{f \in \mathcal{H}} \mathcal{R}(f)$.

Under a finite amount of training data, we can still get a provable ERM guarantee for any $\mathcal{H}$ that admits a uniform LLN:

$$\mathcal{R}_{\text{emp}}(\hat{f}) - \min_{f \in \mathcal{H}} \mathcal{R}(f) \leq \tilde{O}\left(\sqrt{\frac{\text{cap}(\mathcal{H})}{n}}\right), \tag{3}$$

where $\text{cap}(\mathcal{H})$ is a measure of the **capacity** of the function space $\mathcal{H}$. If $\mathcal{H}$ is a set with a finite number of elements, then $\text{cap}(\mathcal{H}) = |\mathcal{H}|$.[3] If $\mathcal{H}$ has infinitely many elements, then the VC-dimension is a common measure of complexity. It is worth noting that this bound is tight for most notions of capacity; one can always construct some dataset and some $\mathcal{H}$ where the gap between empirical and true risk scales exactly as $\sqrt{\text{cap}(\mathcal{H})/n}$. A high-capacity $\mathcal{H}$ thus implies a larger gap between the empirical and true risk, which is indicative of overfitting.

[1] Denoting risk by $\mathcal{R}(\hat{f})$ is a slight abuse of notation. Technically, risk is a function of the learning algorithm $\mathcal{A}$ that produces $\hat{f}$, since $\hat{f}$ is a random variable. In a further abuse of notation, we will also use $\mathcal{R}(f)$ to denote the risk of a fixed function $f \in \mathcal{H}$.

[2] Technically, structural risk minimization (SRM)—a geneneralization of ERM—has been the basis. Essentially, SRM is ERM with a regularization term.

[3] The $\tilde{O}$ notation is the standard big-$O$ notation where logarithmic factors are suppressed.

Neural networks have a finite number of parameters and thus admit a uniform LLN. However, the extremely large number of parameters also implies a large capacity. Almost every notion of capacity—including VC-dimension, spectral bounds, and margin bounds—result in *vaccuous* generalization guarantees [Jiang et al., 2020], where the best bound between $\mathcal{R}_{\text{emp}}(\hat{f})$ and $\min_{f \in \mathcal{H}} \mathcal{R}(\hat{f})$ is worse than the risk associated with random guessing.

Yet neural networks, which are trained to minimize the empirical risk, generalize well in practice. These results suggest that either (1) we do not have a reliable measure of capacity for large neural networks, or (2) we have to look beyond the ERM principle to understand why neural networks generalize well.

---

## 3) Failures of Classical Theory: Optimization

The ERM analysis also assumes that, in practice, we are able to find the model that globally minimizes of the empirical risk. For most classical machine learning models, the empirical risk minimization problem is convex, and so global minimization can be performed in polynomial time. Though some "classical" machine learning models feature non-convex optimization problems, they are usually structured in a way where greedy optimization algorithms can find minima that are provably close to global minima.

Neural networks on the surface feature no special structure that would make optimization easy. Even if one were to remove the non-linear activation functions, the presence of multiple layers makes neural network training non-convex. Yet in practice modern neural networks (with SGD optimization, ReLU non-linearities, and normalization layers) are generally able to achieve essentially zero training error. Moreover, though two neural networks trained from a different initializations will converge to different sets of parameters, they will generally achieve almost the exact same training and test error!

---

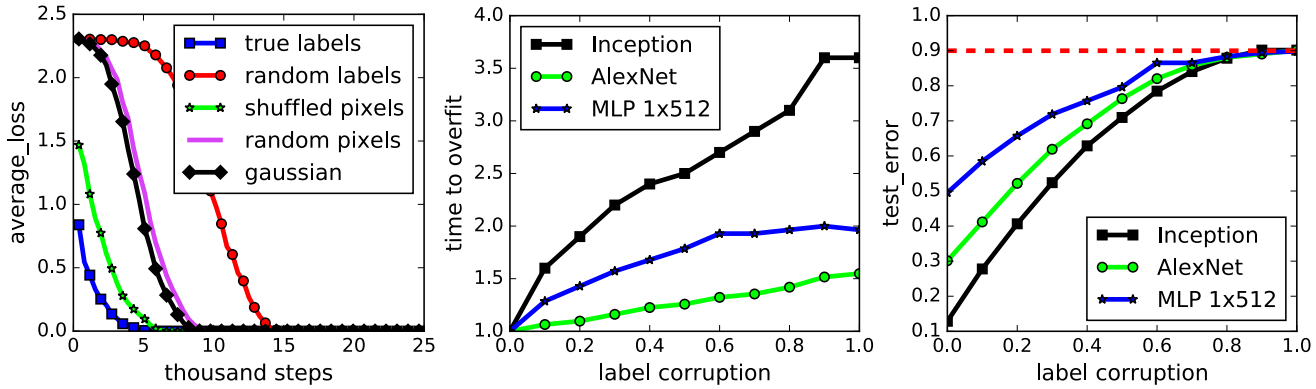
## 4) Understanding Deep Learning Requires Rethinking Generalization

In a landmark paper entitled "Understanding Deep Learning Requires Rethinking Generalization," Zhang et al. [2017] highlighted just how large the gap between classical theory and practice is. The authors performed a very simple experiment: they trained several common neural network architectures on a standard image classification benchmark (CIFAR-10). However, they randomly permuted the labels for some portion the training data—i.e. some portions of "dog" images were assigned the label "cat" and vice versa. The authors found that, even if 100% of labels were corrupted so that the labels were completely uninformative, the neural networks could still be trained to achieve 0% training error. This implies that the neural networks had the capacity to *memorize* the training data—as there was absolutely no "signal" in the images that influence the assigned training labels.

Unsurprisingly, the test error (on uncorrupted labels) was quite large when the training labels were corrupted, and the test error grew proportionally with the fraction of corrupted labels. What is surprising is that the test error did not grow as quickly as one might expect. Despite the fact that the neural networks were memorizing incorrect labels, the resulting models were still able to make predictions better than random guessing on (uncorrupted) unseen data!

This paper shocked the machine learning community, which was already convinced that we did not understand neural networks. It demonstrated that modern neural networks

- have sufficient capacity to memeorize large training sets;
- achieve zero training error, even on corrupted data, despite a non-convex training objective; and




(a) learning curves (b) convergence slowdown (c) generalization error growth

Figure 2: Training error and test error on CIFAR-10 as a function of the fraction of corrupted labels. Reproduced from [Zhang et al., 2017].

- generalize to unseen data, even when the noisy training data are *memorized*.

---

## 5) Outline of This Course

Throughout this course, we will investigate why neural networks generalize and optimize well despite overparameterization and non-convexity. As we will see, the answers flip classical intuitions on their head: neural networks generalize and optimize well *because of, not in spite of,* their overparameterized nature.

We will begin by investigating the generalization properties of high-dimensional linear regression, which will serve as a foundation when we move to neural networks. Despite the ubiquity of linear regression, many of the generalization results that we will discuss are from the past 5 years, as researchers had previously not thought to investigate models that are overparameterized and interpolate/memorize the training data.

We will then connect the results from linear regression to deep learning through the study of infinite width neural networks. Recent landmark results have shown that neural networks become increasingly similar to kernelized linear regressors as width approaches infinity. Though these results do not fully explain the success of (finite width) models, they being to unlock why "bigger is better" for neural networks.

With time permitting, we will discuss more recent developments in the theory of neural networks, including analyses of feature learning and scaling laws.

---

## References

M. Belkin. Fit without fear: Remarkable mathematical phenomena of deep learning through the prism of interpolation. *Acta Numerica*, 30:203–248, 2021.

Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.