Chapter 13

Generalization performance of machine learning models

This chapter gives a preview of generalization performance of deep networks. We will take a more abstract view of learning algorithms here and focus only on binary classification. We will first introduce a "learning model", i.e., a formal description of what learning means. The topics we will discuss stem from the work of two people: Leslie Valiant who developed the most popular learning model called Probably Approximately Correct Learning (PAC-learning) and Vladimir Vapnik who is a Russian statistician who developed a theory (called the VC-theory) that provided a definitive answer on the class of hypotheses that were learnable under the PAC model.

13.1 The PAC-Learning model

Our goal in machine learning is to use the training data in order to construct a model that generalizes well, i.e., has good performance outside of the training data. Formally, we search over a hypothesis space $\mathcal{F}$, e.g., a specific neural net architecture, using the available data to find a good hypothesis $f \in \mathcal{F}$. As we motivated in Chapter 2, without further assumptions, we cannot guarantee that this hypothesis works well on test data. We therefore assume two things in this chapter:

1. Nature provides independent and identically distributed samples $x \in \mathcal{X}$ from some (unknown to the learner) distribution P.

2. Nature labels these samples with $c(x)$ which is again unknown to the learner.

Both training and test data are samples from Nature's distribution P. We will also assume that even if the true labeler $c(x)$ is unknown to us, we know that it belongs to a chosen hypothesis class $c(x) \in \mathcal{C}$ and is deterministic, i.e., Bayes error is zero. Changing this assumption does not change the crux of this theory.

Consider a learning algorithm, denoted by L. Given a dataset $D = \left\{(x^i, c(x^i)\right\}_{i=1}^n$ and a hypothesis class $\mathcal{C}$, the population risk (for classification) of the hypoth-

esis output by this learning algorithm is

$$R(h) = \mathop{\mathbb{E}}_{x \sim P} \left[\mathbf{1}_{\{f(x) \neq c(x)\}} \right]$$

Let us assume that the learning algorithm is deterministic for now, i.e., given a training dataset D it returns a unique answer f. Let us assume that the hypothesis class that the learner searches over, named $\mathcal{F}$ is the same as the hypothesis class $\mathcal{C}$. What do we want from this algorithm?

We expect that it works well for all hypotheses Nature could use to label data $c \in \mathcal{C}$ and all datasets D drawn from P. The PAC-Learning model postulates the following desiderata upon the learning algorithm.

1. We are okay with an answer f with error

$$R(f) \in [0, 1/2)$$

because we only have access to finitely many training data. This is the "approximate correct" part of the PAC-Learning. However the error should decrease as n increases.

2. The dataset D is a random variable. This implies that the hypothesis outputted by the learning algorithm $f(D)$ is also a random variable. The above statement therefore should hold with some large probability over draws of the dataset D. In other words, there can be a small probability that a non-representative dataset D is drawn and we do not expect the learner to output a good hypothesis with $R(f) < 1/2$. However the probability of such failure, let us call it $\delta \in [0, 1/2)$, should also become smaller if more data is provided. This is the "probably" part of PAC-Learning.

We now have a definition of what it means to be a good learning algorithm.

Definition 13.1 (PAC-learnable hypothesis class). A hypothesis class $\mathcal{C}$ is PAC-learnable if there exists an algorithm L such that for every $c \in \mathcal{C}$, for every $\epsilon, \delta \in [0, 1/2)$, if L is given access to $n(\epsilon, \delta)$ i.i.d. training data from P and their corresponding labels c then it outputs a hypothesis $h_D \in \mathcal{C}$ such that

$$\mathbb{P}_D \left(R(f) < \epsilon \right) \geq 1 - \delta.$$

We want the learner to be statistically efficient, i.e., as our desiderata ϵ, δ get smaller, we should expect $n(\epsilon, \delta)$ to not grow too quickly. For instance, we would like $n(\epsilon, \delta)$ to be a polynomial function of $1/\epsilon$ and $1/\delta$. The minimum number of samples $n(\epsilon, \delta)$ required to learn a hypothesis class $\mathcal{C}$ is called the sample complexity of $\mathcal{C}$. One is also typically interested in the computational complexity of finding f, e.g., to avoid a brute-force algorithm L that searches over the entire hypothesis class $\mathcal{F} = \mathcal{C}$; we will not worry about it here.

It is important to notice that PAC-learning assumes nothing about *how* L is going to use the data, e.g., whether it runs SGD or what surrogate loss it uses, or even whether it performs Empirical Risk Minimization. In this sense, the above learning model is very abstract and we should expect only qualitative answers from this theory.

Example 13.2 (Learning Monotone Boolean Formulae). Let $x = [x_1, \ldots, x_d]$ be the datum and $c(x)$ be a conjunction, e.g.,

$$c(x) = x_1 \wedge x_3 \wedge x_4.$$

To take a few examples, $c(10011) = 0$ and $c(11110) = 1$. Such formulae are called monotone because no literals show up as negated in the formula.

We can have the hypothesis class $\mathcal{F}$ to be the set of all possible conjunctions of d variables $x_1, \ldots, x_d$. Each literal x_i can be in the conjunction or not, so the total number of hypotheses in $\mathcal{F}$ is 2^d. [1] Observe that since this is exponential in d, an algorithm L that brute-force searches over $\mathcal{F}$ will have a large computational complexity. Also observe that since the true hypothesis $c \in \mathcal{F}$, there exists an answer f that the algorithm L can output that achieves zero training error, i.e.,

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{f(x^i) \neq c(x^i)\}} = 0.$$

But for a fixed amount of data n, there is some probability that the minimizing hypothesis f has zero training error but large population risk. As the number of data n is large, we expect this event to be less and less probable.

Consider an algorithm L that does the following. It starts with the hypothesis

$$f^0(x) = x_1 \wedge x_2 \wedge \cdots \wedge x_d$$

with all literals and for every datum with a label 1, it deletes all literals x_i that are not in this datum to update the hypothesis f; this makes sense because if the deleted literals were zero in some input, f and c would predict different outputs. Remember that since $c(x) \in \mathcal{F}$, we cannot have a datum with input $1111\ldots1$ and output 0.

What kind of errors does this algorithm make? If some literal x_i was deleted, it is because it had the value $x_i = 0$ on a positively labeled sample. So, it should be deleted, otherwise the hypothesis will output 0. So, we only output a wrong hypothesis if more literals present than those in $c(x)$. So, the output $f(x)$ can only make an error on data labeled 1 by $c(x)$, never on the ones labeled zero. Our algorithm therefore only has false negatives.

We now see why requesting more samples diminishes the probability of this event happening. Let $p_i = \mathbb{P}_{x \sim P}\left[c(x) = 1, x_i = 0 \text{ in } x\right]$. Therefore

$$R(h) \leq \sum_{x_i \in f} p_i$$

If some p_i is small, then it does not contribute much to the error. If some p_i is large then we make sure to see enough samples so that we remove that x_i from f. After all, it only takes one appearance of this event to delete this x_i, and the event has probability p_i which is large. Rigorously, if all x_i in f have $p_i < \epsilon/d$ then $R(h) < \epsilon$. On the other hand, if some x_i has $p_i > \epsilon/d$ then the probability of having this x_i in f is the probability that the event of p_i never happens in the draw of n samples. But this new probability is smaller than $1 - \epsilon/d$. And the event will never happen in n i.i.d. draws with probability at most $(1 - \epsilon/d)^n \leq e^{-n\epsilon/d}$. Using the union bound, since there are at most d literals in f, the probability that there is at least one such "bad event" is at most $de^{-n\epsilon/d}$.

[1] Actually the total number of conjunctions is $2^d + 1$ because for the null-conjuction (without any literals) we can have the constant $c(x) = 0$ or $c(x) = 1$ for all x. We should therefore explicitly make sure $c(111\ldots11) = 0$ is not in the true labeling function. But we ignore this corner case, and silently assume that only the hypothesis $c(x) = 1 \forall x$ is in our class $\mathcal{C}$.

If this bad event never happens the population risk is less than ϵ. Of course, such a bad event happening would be devastating. For some distributions it could lead the error up to 1. However, in our PAC-learning setting we can accept this as long as it happens rarely with probability at most δ. Since

$$de^{-n\epsilon/d} < \delta \iff n \geq \frac{d}{\epsilon}\log\frac{d}{\delta}$$

we are guaranteed to meet the PAC criteria: of error less than ϵ with probability at least $1-\delta$.

Note that both the sample complexity and computational complexity are polynomial. We have thus shown that the class of Monotone Boolean Formulae is (ϵ, δ)-PAC learnable.

13.2 Concentration of Measure

Two very important results from probability theory that we will use are the Union Bound and the Chernoff Bound.

13.2.1 Union Bound (or Boole's Inequality)

For any countable set of events, $\{A_1, \cdots, A_n, \cdots\}$,

$$\mathbb{P}\left(\bigcup_i A_i\right) \leq \sum_i \mathbb{P}\left[A_i\right].$$

⚠ If we want a better approximation of the probability of the union of multiple events and we know more about the problem at hand we can use what are called Bonferroni inequalities.

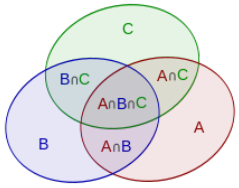
This is a rather loose, but useful, upper bound and is (mostly) embedded in the assumptions of what we call a "probability measure" in probability theory (σ-subadditivity). This essentially means that it can be used without any extra assumptions in practice.

By the inclusion-exclusion principle for finite set of events $\{A_1, \cdots, A_n\}$,

$$\mathbb{P}\left(\bigcup_{i=1}^n A_i\right) = \sum_{i=1}^n \mathbb{P}\left(A_i\right) - \sum_{1\leq i<j\leq n} \mathbb{P}\left(A_i, A_j\right) + \ldots + (-1)^{n-1}\mathbb{P}\left(A_1, A_2, \cdots, A_n\right)$$

We can get better approximations of the union, if we use the first $k \leq n$ terms above. If we stop at odd k, we get an upper bound. If we stop at even k we get a lower bound. The error of the approximation is decreasing with k.

❓ Where did we use the union bound in the proof for the PAC-learnability of the class of monotone Boolean functions?



13.2.2 Chernoff Bound

❓ Try to prove that $\mathbb{P}\left(\bigcap_{i=1}^n A_i\right) \geq 1 - \sum_{i=1}^n \mathbb{P}\left(A_i^c\right)$

Let $A_1, \cdots, A_n$ be a sequence of i.i.d. random variables. We focus on the case of Bernoulli random variables where $\mathbb{P}\left(A_i = 1\right) = p$. We would like

to estimate p from samples. One way to do this is to compute the empirical average

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n} A_i$$

and estimate how close it is to the true p. We know that as $n \to \infty$

Weak Law For all $\epsilon > 0$ we have

$$\mathbb{P}\left(|\hat{p} - p| > \epsilon\right) \to 0.$$

This is also known as convergence in probability.

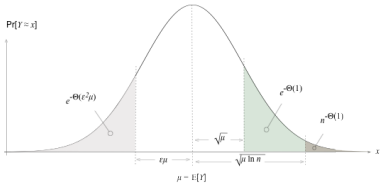
Strong Law We have almost sure convergence, i.e.,

$$\mathbb{P}\left(\lim_{n\to\infty} \hat{p} = p\right) = 1.$$

Central Limit Theorem As $n \to \infty$, the quantity $\sqrt{n}(\hat{p} - p)$ is distributed as a Normal distribution with mean zero and variance $p(1-p)$. Notice that as opposed to the law of large numbers, the central limit theorem also gives us a rate of convergence, i.e., how many samples n are necessary if want the difference to be close to a Normal distribution. If we set $\sigma^2 = p(1-p)$ we can rewrite the Central Limit Theorem as

$$\mathbb{P}\left(|\hat{p} - p| > \epsilon\right) \leq 2e^{-n\epsilon^2/(2\sigma^2)}.$$

⚠ This picture makes it easy to remember concentration inequalities for an n-dimensional Gaussian random variable Y.



Chernoff Bound Since $\sigma^2 = p(1-p) < 1/4$ we have from CLT that

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_i A_i - p\right| > \epsilon\right) \leq 2e^{-2n\epsilon^2}.$$

An easy way to remember the Chernoff bound is that if we want the average of n random variables to be ϵ-close to their expected value with probability at least $1-\delta$, then we need

$$n = \Omega\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$$

samples.

❓ Do you see any patterns in the Chernoff bound with sample complexity in PAC-learning?

Concentration of measure is an exciting area of probability theory and similar results can be obtained for other distributions, other functions than averaging of random variables $A_1, \ldots, A_n$ etc. Popular inequalities are Markov's Inequality, Chebyshev's Inequality and Chernoff Bounds (and Hoeffding's Inequality as an important special case). They are written in terms of increasing tightness, but also of increasing assumptions of what we need to know in order compute them. You can read a very good introduction to this topic at https://terrytao.wordpress.com/2010/01/03/254a-notes-1-concentration-of-measure/.

In general, Markov's inequality only needs the expectation, Chebyshev's Inequality needs the variance too, while Chernoff bounds usually need the

whole moment generating function. They are all applications of Markov's Inequality on higher order statistics. The value of Chernoff bounds is increasingly more important when we talk about distributions of few sufficient statistics, like the Bernoulli distribution (or any exponential distribution).

13.3 Uniform convergence

We now lift the assumption that Nature's labeling function $c \in \mathcal{C}$. After all, even if there exists such a true deterministic c we can never be sure that it is inside $\mathcal{F}$, say the class of neural networks of a specific architecture that we are using. This model is called the Agnostic PAC-Learning model.

We will stay within the confinements of Empirical Risk Minimization where we are provided with some samples where we output the hypothesis with the smallest training error

$$\hat{R}(f) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{f(x^i)\neq y^i\}} \quad \text{minimizing this gives} \quad f_{\text{ERM}} \in \mathcal{F}.$$

The population risk is

$$R(f) = \mathop{\mathbb{E}}_{(x,y)\sim P}\left[\mathbf{1}_{\{f_{\text{ERM}}(x)\neq y\}}\right] \quad \text{minimizing this gives} \quad f^* \in \mathcal{F}.$$

Observe that f^* is *not* the Bayes optimal predictor that we saw in the bias-variance tradeoff. This is because the former is restricted to the hypothesis class $\mathcal{F}$ while the latter has no such restriction to lie in $\mathcal{F}$, it is simply the optimal hypothesis that minimizes the population risk.

> Our goal while computing a *generalization bound* is to ask the following question: if we obtain an ERM hypothesis f_{ERM} with a good training error, then does this also mean that the population risk of the best hypothesis in the class f^* is small?

The above question is central, answering it in the affirmative ensures that we are using a correct hypothesis class (say neural architecture) and that the error on the training dataset is a good indicator of the performance on the entire distribution. This involves the following two steps.

1. First, we need to make sure that the difference

$$\left|\hat{R}(f_{\text{ERM}}) - R(f_{\text{ERM}})\right| \to 0, \quad n \to \infty.$$

   This is easy, it is akin to the concentration of measure we saw in the previous section.

2. Second, we need to ensure that

$$\hat{R}(f_{\text{ERM}}) \approx \quad R(f^*)$$

   with high probability for every training dataset of n samples using which f_{ERM} is computed. If this is true, it tells us something about the ERM procedure itself, i.e., it tells us whether minimizing the empirical risk

$\hat{R}(f)$ is a good thing if we want to build a classifier that works well on the population.

This is difficult to do, after all f_{ERM} and f^* are totally different hypothesis. Vapnik set up a powerful construction to do this. He showed that a *sufficient* condition to achieve the above is that for all hypotheses in $\mathcal{F}$, the empirical risk and population risk are similar. This is known as uniform convergence.

Let us now develop the two points above. Since data are drawn iid, we can use the Chernoff bound to get that

$$\forall f \in \mathcal{F}, \mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| > \epsilon\right) \leq 2e^{-2n\epsilon^2}.$$

If the hypothesis class is finite $\mathcal{F}$, we can use the union bound to show that for *any* hypothesis, the training error and population risk are close.

$$\begin{aligned}
&\mathbb{P}\left(\exists f \in \mathcal{F} : \left|\hat{R}(f) - R(f)\right| > \epsilon\right) \\
&\leq \sum_{f \in \mathcal{F}} \mathbb{P}\left(\left|\hat{R}(f) - R(f)\right| > \epsilon\right) \\
&\leq |\mathcal{F}|\, 2e^{-2n\epsilon^2}.
\end{aligned}$$

If we want this above probability of a bad event to be less than δ we therefore need

$$n \geq \frac{1}{2\epsilon^2} \log \frac{2\,|\mathcal{F}|}{\delta} \tag{13.1}$$

training data. Notice how this bound changed from the Monotone Boolean function example: we need $\mathcal{O}(1/\epsilon)$ times more samples to get the uniform convergence result.

Suppose we had a classifier f with 2% gap ($\epsilon = 0.02$) between the training error $\hat{R}(f)$ and the validation error (which is a proxy for the population risk $R(f)$), if we want to reduce this gap by half to 1% ($\epsilon = 0.01$) , we need 4 times as many training data. We could also reduce this gap by fitting a classifier with small $|\mathcal{F}|$ but in this case, both the training and validation error will increase even if their gap decreases.

Next, we need a relation between the population risk of f_{ERM} and the best possible predictor $f^*\mathcal{F}$ in our hypothesis class. Observe that

$$\begin{aligned}
R(f_{\text{ERM}}) &\leq \hat{R}(f_{\text{ERM}}) + \epsilon && (\text{Chernoff bound on } f_{\text{ERM}}) \\
&\leq \hat{R}(f^*) + \epsilon && (f_{\text{ERM}} \text{ has the smallest training error}) \\
&\leq R(f^*) + 2\epsilon && (\text{Chernoff bound on } f^*).
\end{aligned}$$

The two Chernoff bound inequalities hold with probability at least $1 - \delta$ so the final inequality

$$R(f_{\text{ERM}}) \leq R(f^*) + 2\epsilon$$

holds with probability at least $1 - 2\delta$. Substitute this in (13.1) to get

$$R(f_{\text{ERM}}) \leq R(f^*) + 2\sqrt{\frac{1}{2n} \log \frac{|\mathcal{F}|}{\delta}} \tag{13.2}$$

with probability $1 - \delta$. A result of this kind is called a Vapnik-Chernovenkis (VC) bound or a PAC bound.

Let us consider our monotone Boolean formulae example again. Since $|\mathcal{F}| = 2^d$, if the input dimension is $d = 1000$ and we set $\delta = 10^{-3}$, the VC-bound predicts the following. In this case, we should imagine running ERM to pick the best hypothesis f_{ERM}, not the elimination algorithm we discussed in the section on monotone Boolean formulae.

1. With $n = 1000$ data, we have $R(f_{\text{ERM}}) \leq R(f^*) + 1.41$. This is vacuous/non-informative since the population risk is an expectation of indicator variables and should therefore be less than 1.
2. With $n = 10^5$, we have $R(f_{\text{ERM}}) \leq R(f^*) + 0.44$. This is informative, it means that the population risk of the classifier obtained by ERM is within 44% of the population risk of the best classifier f^* in that class. Of course it is only meaningful if f^* generalizes well, i.e., if $R(f^*)$ is small. This will happen if the hypothesis class $\mathcal{F}$ is large enough.
3. With $n = 10^6$, we have $R(f_{\text{ERM}}) \leq R(f^*) + 0.04$.

13.4 Vapnik-Chernovenkis (VC) dimension

In the above section, the concept/hypothesis class was assumed to be finite $|\mathcal{C}| < \infty$. The union bound of course breaks if this is not the case. Notice that once we pick a neural architecture (hypothesis class), the number of possible models (hypotheses), each with different weight vectors, is infinite. Observe that in the monotone Boolean formulae example, the algorithm L was using the training data to eliminate hypothesis from $\mathcal{C}$, this is not going to work $\mathcal{C}$ is not finite. It is therefore a natural question whether we can still learn a hypothesis class with a finite number of training data.

Vladimir Vapnik and Alexey Chernovenkis (**?**) developed the so-called VC-theory to answer the above question. Technically, VC-theory transcends PAC-Learning but we will discuss only one aspect of it within the confinements of the PAC framework. VC-theory assigns a "complexity" to each hypothesis $f \in \mathcal{C}$.

Shattering a set of inputs We say that the set of inputs $D = \{x^1, \cdots, x^n\}$ is *shattered* by the concept class $\mathcal{C}$, if we can achieve every possible labeling out of the 2^n labellings using some concept $c \in \mathcal{C}$. The size of the largest set D that can be shattered by $\mathcal{C}$ is called the VC-dimension of the class $\mathcal{C}$. It is a measure of the complexity/expressiveness of the class; it counts how many different classifiers the class can express.

If we find a configuration of n inputs such that when we assign *any* labels to these data, we can still find a hypothesis in $\mathcal{C}$ that can realize this labeling, then

$$\text{VC}(\mathcal{C}) \geq n.$$

On the other hand, if for every possible configuration of $n + 1$ inputs, we can always find a labeling such that no hypothesis in $\mathcal{C}$ can realize this labeling, then

$$n \leq \text{VC}(\mathcal{C}).$$

If we find some n for which both of the above statements are true, then

$$\text{VC}(\mathcal{C}) = n.$$

Some examples.

- d-dim Linear Threshold Functions: VC-dim $= d + 1$.

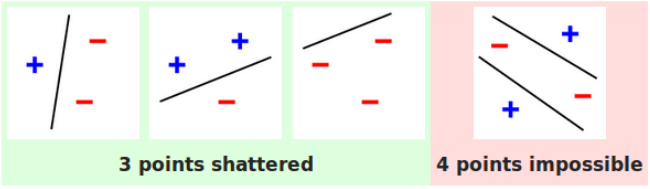


Figure 13.1: d=2: See that for the lower bound, we found some configuration of the 3 points, such that a linear threshold function always separates the points consistently with the labels; for any possible labeling. 3 such labellings are shown, convince yourselves that it can be done for all 8 cases. Observe that we cannot do the same for 4 points. In the figure above one such unrealizable configuration is given (With the "XOR" labeling). To prove the upper bound we need to talk about ANY configuration though. See that the only other case for 4 points, is that one point is inside the convex hull generated from the other 3. Find the labeling that cannot be obtained with linear classifiers in this case.

- 2 dimensional axis aligned rectangles: VC-dim = 4 (exercise)
- Monotone Boolean Formulae: VC-dim $= n$ (exercise).
- If the hypothesis class is finite, then

$$\text{VC}(\mathcal{F}) \leq \log |\mathcal{F}| .$$

- If $x \in \mathbb{R}$ and our concept class includes classifiers of the form

$$\text{sign}(\sin(wx))$$

where w is a learned parameter, then

$$\text{VC} = \infty.$$

- For a neural network with p weights and sign activation function

$$\text{VC} = \mathcal{O}(p \log p).$$

It is a deep result that if the VC-dimension of concept class is finite $V = \text{VC}(\mathcal{F}) < \infty$, then this class has the uniform convergence property (for any $f \in \mathcal{F}$, the empirical and population error are close). Therefore, we can learn this concept class agnostically (without worrying about whether Nature's labeling function c is in our hypothesis class $\mathcal{F}$ or not) in the PAC framework with

$$n = \Omega\left(\frac{V}{\epsilon^2}\log\frac{V}{\epsilon} + \frac{1}{\epsilon^2}\log\frac{1}{\delta}\right)$$

training data. If a hypothesis class as infinite VC-dimension, then it is not PAC-learnable and it also does not have the uniform convergence property.

The above result written in another form looks as follows. For a (finite or infinite) hypothesis class $\mathcal{F}$ with finite VC-dimension $V = \text{VC}(\mathcal{F})$

$$R(f_{\text{ERM}}) \leq R(f^*) + 2\sqrt{\frac{1}{n}\left(2V - \log \delta\right)} \tag{13.3}$$

with probability at least $1 - \delta$. This is an important expression to remember: the number of samples n required to learn a concept class scales linearly with the VC-dimension V. A more refined version of this bound looks like

$$R(f_{\text{ERM}}) \leq R(f^*) + 2\sqrt{\frac{1}{n}\left(V\left(\log \frac{2n}{V} + 1\right) + \log \frac{4}{\delta}\right)}. \tag{13.4}$$

Bounds on the VC-dimension of deep neural networks For general classifiers, it is typically difficult to compute the VC dimension. One instead finds upper and lower bounds for the VC dimension to be used in inequalities of the form (13.4). Bounds on the VC-dimension of deep network architectures are available (**?**). With p weights and L layers, an essentially tight VC-dimension looks like

$$\Omega\left(p\, L \log \frac{p}{L}\right) = \text{VC}(\mathcal{F}) = \mathcal{O}(p\, L \log p)$$

for deep networks with ReLU nonlinearities.

This bound is not entirely useful in the VC-theory however. For instance, the ALL-CNN network you used in your homework with $p \approx 10^6$ and $L = 10$ has VC $\approx 10^8$. If we use the coarse VC-bound in (13.3) with $n = 50,000$ samples, we have

$$R(f_{\text{ERM}}) \leq R(f^*) + 40$$

which is a vacuous generalization bound. However, remember that this is simply an *upper bound* on the generalization error of ERM. It is clear from empirical results in the literature (including your homework problems) that deep networks indeed generalize well to new data outside the training set and that means $R(f_{\text{ERM}})$ is small.

The gap in applying VC-theory to deep networks therefore likely stems from the need for uniform convergence: we may not need that the empirical and population risk are close for *all* hypotheses in the class. If we only have uniform convergence within a small subset $F \subset \mathcal{F}$ and if $\text{VC}(F) \ll \text{VC}(\mathcal{F})$ and if the training algorithms like SGD always find ERM minimizers $f_{\text{ERM}} \in F$, then VC-theory/PAC-Learning do predict that deep networks will generalize well. Understanding this is the subject of a large body of ongoing research.