# Optimizing for Generalization Guarantees

Yuchen Xin

Autumn 2025

## 1 Introduction

Throughout machine learning theory we see a recurrent theme of *bounding true errors using empirical errors with high probability*. However, as deep learning models (with far more parameters than the amount of training data) get more ubiquitous, data-independent bounds from standard learning theory become vacuous (see Section 1.1). To get around this, we utilize the data we are given, to construct data-dependent bounds that hold nonvacuously with high probability.

In particular, this paper focuses on optimizing a bound that relates empirical error with true error. The deep learning models in question will be standard multilayer perceptrons, trained for a binary classification variant of the MNIST dataset from [2]. But we take these models and introduce an additional "stochastic" neural network structure (made rigorous in Section 2). Effectively, instead of working with just the hypothesis class $\mathcal{H}$, we work with parametrized *distributions* over $\mathcal{H}$, such that the *stochastic* models sample from $\mathcal{H}$ before prediction. Applying PAC-Bayes theory as in [3] gives us a tight bound on the generalization error with respect to the empirical error, and by post-training our model to optimize the parameters of the distribution over $\mathcal{H}$, we arrive at a data-dependent generalization error bound that holds *nonvacuously with high probability* – what we set out to get!

An astute reader and attentive student of CSE 493S/599S may realize that this is an advanced form *conformal prediction*. However, the "conformity" here is not over output labels, but rather the model weights themselves, and what hypothesis in $\mathcal{H}$ the model corresponds to. There is also a sense of *calibration* like in [1]: we take a model that achieves good train/test loss but has only heuristic uncertainty, and by optimizing parameters to tighten a theoretical PAC-Bayes generalization bound, we get a certificate of rigorous uncertainty:

$$\text{Heuristic uncertainty} \longrightarrow \boxed{\text{"Calibrating" with the PAC-Bayes Bound}} \longrightarrow \text{Rigorous uncertainty}$$

In this paper, we first make rigorous why standard bounds (familiar to us from lecture) become vacuous. Then, we rigorously present key theoretical results regarding PAC-Bayes optimization on stochastic neural networks. I will build on the literature to derive a theorem for sample size required to achieve $\epsilon$ generalization difference. Next, I train models to reproduce selected results in [3], and perform additional experiments to try different ways of optimizing the PAC-Bayes bound. Finally, I end with a discussion on why randomness (like in our stochastic models) is sometimes good, if not necessary, for guarantees.

## 1.1 Motivation

As stated above, modern neural networks often have more parameters than training data; as a result, their complexity (VC, Rademacher, etc.) is high. For example, see Theorem 1 at the end of this section, which show that even two-layer neural nets with double the parameters as the sample size can exactly fit any given sample, effectively lower-bounding the size of shatterable sets (i.e. its VC complexity). High complexity implies the theoretical generalization bounds like those we prove in CSE 493S/599S become vacuous.

In particular, let $\mathcal{H}$ be a hypothesis class with $\text{VCdim}(\mathcal{H}) = d$, and let $S$ be any i.i.d. sample of size $|S| = m$ from a distribution $\mathcal{D}$. For some hypothesis $h \in \mathcal{H}$, let $L_S[h]$ be the empirical sample average 0-1 loss and $L_{\mathcal{D}}[h]$ the true generalization average 0-1 loss. With $d$ finite, we proved in class that this implies the uniform convergence property, and to do so we derived the following bound:

$$\mathbb{E}_S\left[\sup_{h\in\mathcal{H}} |L_{\mathcal{D}}[h] - L_{\mathcal{S}}[h]|\right] \leq \frac{4 + \sqrt{\log \tau_{\mathcal{H}}(2m)}}{\sqrt{2m}} \tag{1}$$

where, by Sauer's Lemma,

$$\text{if } m \geq d, \quad \tau_{\mathcal{H}}(m) \leq \left(\frac{em}{d}\right)^d, \qquad \text{and if } m \leq d, \quad \tau_{\mathcal{H}}(m) = 2^m$$

Since we work with average 0-1 loss, the LHS of Equation 1 is is bounded by 1. If the RHS of Equation 1 exceeds 1, we get a vacuous bound, i.e. one that does not tell us more than we already know. Clearly, when the complexity of the model $d$ far exceeds the data size $m$, the RHS becomes $\frac{4+\sqrt{2m}}{\sqrt{2m}} = 1 + \frac{4}{\sqrt{2m}} \geq 1$ and therefore gives us a vacuous bound!

Yet, empirically, we know modern neural networks with such high complexity work well in practice – they commonly achieve low test error, and everybody relies on them. This suggests the existence of generalization capabilities that can be found theoretically by other means.

---

**Theorem** (Theorem 1)**.** *There exists a two-layer neural net with ReLU activations of $2n + d$ parameters that can agree with any function $f : S \to \mathbb{R}$ where $S$ is the sample with $|S| = n$ and $S \subseteq \mathbb{R}^d$.*

*Proof. I adapted this proof from [5], fixing typos and logical gaps from the original paper.*

Let $b \in \mathbb{R}^n$ and $a \in \mathbb{R}^d$ be the parameters involved in the affine transformation prior to ReLU activation, and let $w \in \mathbb{R}^n$ be the weights for the final linear output layer. With these parameters, the two-layer neural net with ReLU activations is equivalent to the function

$$g(x) = \sum_{j=1}^{n} w_j \max\{a^\top x - b_j, 0\}$$

Let $S = \{(x_i, y_i)\}_{i=1}^n$ be any sample of $n$ distinct elements, and notate $y \in \mathbb{R}^n$ as the column vector of all the $y_i$. Define $A \in \mathbb{R}^{n\times n}$ as $A_{ij} = \max\{a^\top x_i - b_j, 0\}$. Then our network predicts $g(S) = Aw$, and we want this to equal $y$ by a suitable choice of $b, a, w$. In what follows we choose $b, a$ to make $A$ invertible; then, setting $w = A^{-1}y$ would complete the proof.

Since the $x_i$ are distinct by definition, there must exist some $a \in \mathbb{R}^d$ such that the numbers $c_i := a^\top x_i$ are all distinct (can be proven with linear algebra that I omit). Also, note that we can swap rows of $A$ without affecting its invertibility (since a row swap is an invertible operation that preserves the magnitude of the determinant). Thus, without loss of generality, take $c_1 < \cdots < c_n$. Then choose each entry $b_i$ so that we have the interleaving sequence $b_1 < c_1 < \cdots < b_n < c_n$, i.e. for each $1 \leq i \leq j \leq n$ we have $b_i < c_j$ or $c_j - b_i > 0$, and conversely if $i > j$ then $c_j - b_i < 0$.

This means $A_{ij} = \max\{c_i - b_j, 0\}$ is zero above the main diagonal but nonzero (and in fact positive) everywhere else, which characterizes $A$ as a lower triangular matrix with positive elements along the main diagonal. Then, since for triangular matrices we have $\det(A) = \prod_{i=1}^n A_{ii} \neq 0$, we know $A$ is invertible.

□

## 2 Stochastic Model Bounding

As previously introduced, to apply PAC-Bayes theory we work with stochastic neural networks. We now make this rigorous.

Consider the hypothesis class $\mathcal{H}$ indexed by weights $w \in \mathbb{R}^d$, with each weight inducing a (binary) predictor $h_w : \mathbb{R}^k \to \{-1, 1\}$. We now speak of a randomized classifier as a distribution $Q$ on the $\mathbb{R}^d$ weights. A stochastic model is one that samples weights from $Q$ during inference, and then propagates the input through the sampled weights to get a (random) output.

To measure the error of these stochastic learning algorithms, we utilize the 0-1 loss $\ell : \mathbb{R} \times \{-1, 1\} \to \{0, 1\}$ given by $\ell(\hat{y}, y) = \mathbb{I}(\text{sign}(\hat{y}) \neq y)$. Then, for a sample $S_m = \{(x_i, y_i)\}_{i=1}^m \overset{\text{i.i.d.}}{\sim} \mathcal{D}$ and hypothesis $h \in \mathcal{H}$, we define

$$\hat{e}(h, S_m) := \frac{1}{m} \sum_{i=1}^{m} \ell(h(x_i), y_i) \tag{2}$$

as the empirical error of $h$, and then the empirical error and true error of $Q$ (simply called "empirical error" and "true error" for the remainder of this paper) as

$$\hat{e}(Q, S_m) := \mathbb{E}_{w \sim Q}[\hat{e}(h_w, S_m)]$$

$$e(Q) := \mathbb{E}_{w \sim Q}\left[\mathbb{E}_{S_m \overset{\text{i.i.d.}}{\sim} \mathcal{D}}[\hat{e}(h_w, S_m)]\right] = \mathbb{E}_{w \sim Q}\left[\Pr_{(x,y) \sim \mathcal{D}}[\text{sign}(h_w(x)) \neq y]\right] \tag{3}$$

respectively, where the last equality follows from $S_m$ being sampled i.i.d. from $\mathcal{D}$, as well as the definition of $\ell$. I re-emphasize that the models we train are not trying to learn some fixed $w$, but rather the distribution $Q$ over possible $w$ – hence these errors marginalize over $w$.

Now, how can we bound $e(Q)$? Our best guess for the true error is always the empirical error (for a given data sample $S_m$), so we would like to relate $\hat{e}(Q, S_m)$ and $e(Q)$. To do so, we introduce the Kullback-Liebler (KL) divergence (from [4]) of two distributions $Q, P$ on $\mathbb{R}^d$ as

$$\text{KL}(Q \| P) = \int \log \frac{q(x)}{p(x)} q(x) \mathrm{d}x \tag{4}$$

In practice, we interpret the KL divergence in the context of Bayesian probability. We take $P$ to be the prior (before seeing any data), and $Q$ to be the posterior (after learning from the data). Then $\text{KL}(Q \| P)$ measures how much information is lost if we approximate $Q$ by $P$, or roughly how much $Q$ deviates from $P$.

From Equations 2 and 3 above, we can interpret the errors as one minus the probability of being correct, or just $1 -$ accuracy. Then the errors describe means of Bernoulli distributions of whether or not the model's binary prediction is correct. If $q$ and $p$ are means of Bernoulli distributions of the posterior $Q$ and prior $P$, we simply write $\text{KL}(q \| p)$ to denote the KL divergence of the Bernoulli distributions. Then, by 4, we are interested in the KL divergence of errors $\text{KL}(\hat{e}(Q, S_m) \| e(Q))$. Intuitively, this term is the generalization complexity, quantifying how far empirical error deviates from true error, just like the RHS of Equation 1.

In the following section we will get a bound on $\text{KL}(\hat{e}(Q, S_m) \| e(Q))$ and use that to bound the true error $e(Q)$.

### 2.1 The PAC-Bayes Bound

By a change of measure, one can prove Theorem 2.3 in [3], the **PAC-Bayes bound**: with probability $1 - \delta$ over samples $S_m \overset{\text{i.i.d.}}{\sim} D$, for all prior distributions $P$ over $\mathcal{H}$ independent of the data $S_m$ and all posterior distributions $Q$ over $\mathcal{H}$ possibly dependent on $S_m$,

$$\text{KL}(\hat{e}(Q, S_m) || e(Q)) \leq \frac{\text{KL}(Q || P) + \log \frac{m}{\delta}}{m - 1} \tag{5}$$

Thus the generalization complexity is related to how much the distribution over $\mathcal{H}$ changes as a result from learning from the data. But we have the freedom of choosing the prior $P$ and how to learn the posterior $Q$!

Now, let's use this to bound $e(Q)$. In general for Bernoulli means $p, q$ and a bound on $\mathrm{KL}(q\|p) \leq c$, the KL inverse with respect to $q, c$ is the maximum $p$ consistent with the bound:

$$\mathrm{KL}^{-1}(q \mid c) := \sup\{p \in [0,1] : \mathrm{KL}(q\|p) \leq c\} \tag{6}$$

In effect, by taking $q = \hat{e}(Q, S_m)$ and $c$ the PAC-Bayes bound from 5, then 6 gives us an upper bound for the true error, i.e.

$$e(Q) \leq \mathrm{KL}^{-1}\left(\hat{e}(Q, S_m) \middle| \frac{\mathrm{KL}(Q||P) + \log\frac{m}{\delta}}{m-1}\right)$$

Now, the formula in 6 is hard to use or compute, so instead we use a tight upper bound of the KL inverse.

**Lemma** (Lemma 1)**.** *We have* $\mathrm{KL}^{-1}(q \mid c) \leq q + \sqrt{\frac{1}{2}c}$.

*Proof.* From [3] we use the tight bound $2(q-p)^2 \leq \mathrm{KL}(q\|p)$. To find the KL inverse we suppose $\mathrm{KL}(q\|p) \leq c$, which means any consistent $p$ must satisfy $2(q-p)^2 \leq c$, or $|q-p| \leq \sqrt{\frac{1}{2}c}$.

If $p \geq q$ then $|q-p| = p-q$, meaning $p \leq q + \sqrt{\frac{1}{2}c}$. If $p < q$ then trivially the bound applies, since $c \geq 0$ (due to the KL divergence being non-negative). Thus $\sup\{p \in [0,1] : \mathrm{KL}(q\|p) \leq c\} \leq q + \sqrt{\frac{1}{2}c}$. □

By Lemma 1, we finally have the useful bound

$$e(Q) \leq \hat{e}(Q, S_m) + \frac{\mathrm{KL}(Q||P) + \log\frac{m}{\delta}}{m-1} \tag{7}$$

which holds with probability $1-\delta$ over possible samples $S_m$. This should feel familiar: the true error is related to the empirical error plus some complexity term dependent on the space we are working with, which was VC dimension in agnostic-PAC learning, and KL divergence here under PAC-Bayes. This structure also helps us understand how to optimize for minimal $e(Q)$ – one must simultaneously perform Empirical Risk Minimization (ERM) to minimize $\hat{e}(Q, S_m)$, while accounting for "regularization" by the KL term to control generalization behavior.

In practice, we train a non-stochastic model for ERM (as we normally do, e.g. via SGD) and extract its weights $w_0$. Then we add stochasticity $Q$ by, for example, letting each weight be sampled from a normal distribution with means at $w_0$. Heuristically, we should already be starting with low empirical error $\hat{e}(Q, S_m)$. Then, we fix a data-independent prior $P$, like a hyperparameter, as well as a desired confidence $1-\delta$. We also suppose $Q$ comes from a parametric family of distributions, like Gaussians, to get a closed form expression for 7. This completes the setup, and allows us to finally post-train/calibrate the parameters of $Q$ (i.e. our stochastic model) to optimize 7. To do so, we can use SGD.[1]

When we're done, we evaluate our final stochastic model by computing the KL inverse more precisely than what Lemma 1 gives us. Following the authors of [3], one can use Newton's Method to iteratively refine an approximation of the KL divergence to obtain the desired quantity. We end up with a theoretical high probability guarantee of an upper bound for $e(Q)$, which (as seen in many experiments) turns out to be highly nonvacuous. This makes sense, because much of the derivation above relied on tight bounds, so we are not "losing" much information along the way.

*TL;DR: Optimizing the KL inverse (or "optimizing the PAC-Bayes bound") as in Equation 7 through SGD with respect to a parametrized form for $Q$ is the bridge to getting a guaranteed true error upper bound from the empirical error. In later sections, we will actually experiment with optimizing the PAC-Bayes bound.*

[1] Note that $\hat{e}(Q, S_m)$ is neither differentiable (with respect to the parameters of $Q$) for the purposes of SGD, nor even computable as an expectation over 0-1 loss. Practically, we switch the loss $\ell$ used in $\hat{e}(Q, S_m)$ to a differentiable surrogate loss that upper bounds $\ell$, and then use unbiased estimates of the expectation over $\ell$ by sampling from $w \sim Q$.

## 2.2 What About the "AC" in PAC-Bayes?

Ever since reading [3] about optimizing the PAC-Bayes bound, I've always felt there was a gap. We touched on Bayes with the priors and posteriors, and we know the PAC-Bayes bound holds with high probability, but where is the explicit "approximately correct" part that we are familiar with in CSE 493S/599S? I proceed to prove a theorem about the PAC-Bayes bound that should feel familiar from (agnostic) PAC learnability, with the only assumption being that $\mathrm{KL}(Q\|P)$ is finite (much like assuming the VC dimension is finite).

**Theorem** (Theorem 2)**.** *Let $\epsilon > 0$ be given, and let $P$ be a fixed prior. Let $Q$ be a distribution such that $\mathrm{KL}(Q\|P)$ is at most some number $K$. Then, for any $m \geq \frac{K+\log\frac{2}{\epsilon\delta}+\epsilon-1}{\epsilon/2}$, w.p. $1-\delta$ over samples $S_m$ of size $m$,*

$$e(Q) - \hat{\epsilon}(Q, S_m) \leq \epsilon$$

*Proof.* Rearranging Equation 7 and using the assumption, we get

$$e(Q) - \hat{e}(Q, S_m) \leq \frac{\mathrm{KL}(Q\|P) + \log\frac{m}{\delta}}{m-1} \leq \frac{K + \log\frac{m}{\delta}}{m-1}$$

We seek the minimal $m$ that forces the RHS above to be at most $\epsilon$. First, notice that since $\log(m)$ is concave in $m$, it's upper bounded by its tangent approximation. Such an approximation is found as follows: for any $t > 0$, the derivative of log is $\frac{1}{t}$, giving the linear approximation $y = \log t + \frac{1}{t}(m-t)$. Thus

$$\log m \leq \log t + \frac{m-t}{t}$$

Noting that $\log\frac{m}{\delta} = \log m - \log\delta$, we have

$$\frac{K + \log\frac{m}{\delta}}{m-1} \leq \frac{K + (\log t + \frac{m-t}{t}) - \log\delta}{m-1} = \frac{K + \log\frac{t}{\delta} - 1 + \frac{m}{t}}{m-1}$$

If we enforce that the RHS above is bounded above by $\epsilon$ then the original condition will be too. Doing so gives us $K + \log\frac{t}{\delta} - 1 + \frac{m}{t} \leq (m-1)\epsilon$, which is equivalent to

$$K + \log\frac{t}{\delta} + \epsilon - 1 \leq \left(\epsilon - \frac{1}{t}\right) m \qquad \text{or} \qquad m \geq \frac{K + \log\frac{t}{\delta} + \epsilon - 1}{\epsilon - \frac{1}{t}}$$

This already works as a lower bound on $m$, but is too general for practical use as it contains $t$. Choosing $t = \frac{2}{\epsilon} > 0$, we can simplify the last expression to

$$m \geq \frac{K + \log\frac{2}{\epsilon\delta} + \epsilon - 1}{\epsilon/2}$$

□

There are a couple implications of Theorem 2.

1. As described above, we train to find $Q$ that optimizes the PAC-Bayes bound, but all we have is an upper bound on the true error. By Theorem 2, we will know how many data points are needed to compute the actual true error $e(Q)$ to within $\epsilon$ precision, using the empirical error $\hat{e}(Q, S_m)$. Similarly, by varying $\delta$ we can estimate different confidence bounds for $e(Q)$.
2. As the amount of training data $m$ grows, we expect $\hat{e}(Q, S_m)$ (which can be interpreted as part of the train loss) to go down. But Theorem 2 also tells us that the epsilon gap between $\hat{e}(Q, S_m)$ and $e(Q)$ can be decreased more and more, so we end up simultaneously minimizing true error, as desired.

# 3 PAC-Bayes Experiments

With the theory out of the way, we can finally perform some experiments on PAC-Bayes optimization, to turn a normal model into a stochastic one with theoretical generalization guarantees.

We focus our attention to the problem of binary classification in MNIST – whether or not a $32 \times 32$ pixel handwritten digit is of a digit $\leq 4$ or $> 4$. We use a one-hidden-layer multilayer perceptron model with ReLU activations, and binary cross entropy loss, trained using the Adam optimizer on the binary-ified MNIST data.

We adopt the natural choice of picking a Gaussian prior $P = \mathcal{N}(w_0, \lambda I)$, where $w_0$ are the weights of the pretrained non-stochastic model, and $\lambda$ is a hyperparameter.[2] We also pick Gaussians for the posterior, with diagonal covariances for simplicity, and parametrize with means $w \in \mathbb{R}^d$ and diagonal covariances $\text{diag}(s)$ with $s \in \mathbb{R}^d_+$. These choices have closed form solutions for the KL divergence:

$$\text{KL}(Q\|P) = \text{KL}(\mathcal{N}(w, \text{diag}(s))\|\mathcal{N}(w_0, \lambda I)) = \frac{1}{2}\left(\frac{1}{\lambda}\|s\|_1 - d + \frac{1}{\lambda}\|w - w_0\|_2^2 + d\log\lambda - 1_d \cdot \log s\right) \tag{8}$$

As mentioned in Footnote 1, we also switch the 0-1 loss function to a differentiable, convex surrogate loss $\check{\ell}(\hat{y}, y) = \frac{1}{\log(2)}\log(1 + e^{-\hat{y}y}) \in \mathbb{R}_+$. Applying these choices to Equation 7 with some extra modifications due to Footnote 2, we get a differentiable PAC-Bayes bound objective that can be optimized via SGD.

The accompanying PyTorch code for this project (see Appendix) does just that, computing Equation 7 with these choices to get loss, and backpropagating to perform gradient descent. We use RMSProp with 0 momentum as done in [3], but adapt and rewrite the code entirely to run in modern PyTorch. There were, however, some notable differences in hyperparameters in my experiments:

1. The authors of [3] used SGD (batch size 1) to optimize the PAC-Bayes bound, and explicitly did not try mini-batch. Since SGD was taking too long, I use a mini-batch size of 16.
2. The authors of [3] trained over 4 epochs, using a learning rate of 0.001 for the first 3 epochs and dropping to 0.0001 for the last 1. I tried this and got mostly vacuous bound results, so I increased to training over 10 epochs, using a learning rate of 0.001 for the first 7 epochs and dropping to 0.0001 for the last 3.

The below figure compares my results[3] with results from [3] across multiple single-hidden-layer multilayer perceptron networks, with different hidden dimension $h$.

| Experiment | $h = 600$ ([3]) | $h = 600$ (me) | $h = 1200$ ([3]) | $h = 1200$ (me) |
|---|---|---|---|---|
| NN Test Error | 0.018 | 0.014 | 0.018 | 0.014 |
| SNN Train Error | 0.028 | 0.0254 | 0.027 | 0.0205 |
| SNN Test Error | 0.034 | 0.0334 | 0.035 | 0.0278 |
| PAC-Bayes Bound | 0.161 | 0.1471 | 0.179 | 0.3468 |
| $\text{KL}(Q\|P)$ | 5144 | 5108 | 5977 | 20318 |

Table 1: PAC-Bayes optimization results comparison. We keep $\delta = 0.05$ for all experiments. NN refers to the "pre-trained" model with fixed weights, while SNN refers to the stochastic "post-trained" model. $\text{KL}(Q\|P)$ in this case is the KL divergence between the Gaussian posterior and Gaussian prior.

Based on Table 1, I make a couple observations:

- The SNN test error falls below the PAC-Bayes bound, confirming the fact that Equation 7 holds. The PAC-Bayes bound is also $< 1$ and therefore nonvacuous, giving us a meaningful bound of average misclassification error (or equivalently, $1 -$ accuracy).
- The SNN train and test errors are above the NN test error, demonstrating that there is no free lunch – adding randomness to our models trades away accuracy, but gives us nonvacuous guarantees.

[2] In practice, as done in [3], we incorporate $\lambda$ into the optimization objective and use a union bound over $\lambda$ to maintain correctness of the PAC-Bayes bound.

[3] Note that all of my results come from training models locally on a Macbook Pro M4 CPU.

In Table 1 we also see that my results closely align with that of [3] for $h = 600$. However, for the higher dimensional model ($h = 1200$) with more parameters, the KL divergence and PAC-Bayes bound are far worse in my models, suggesting that mini-batch gradient descent (as compared to SGD) is unable to get all the weights to converge in the same number of epochs.

Besides the results of the model's performance, we can also uncover what happens during training of the PAC-Bayes bound. For the $h = 600$ model, we have the following curves:

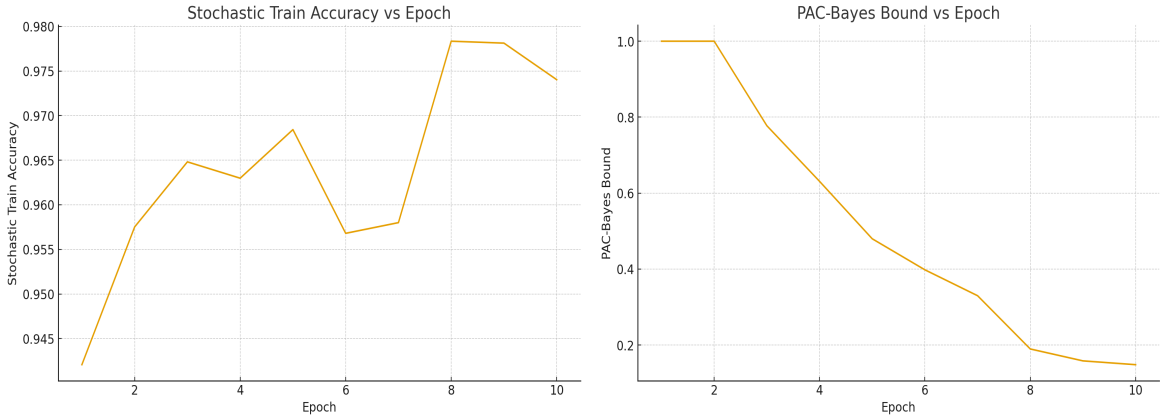


Figure 1: Stochastic train accuracy and PAC-Bayes bound estimate vs. epochs for the $h = 600$ stochastic model.

It is obvious from Equation 7 and Figure 1 that the PAC-Bayes bound should decrease as we optimize it with gradient descent. Interestingly, even though the PAC-Bayes bound incorporates both an empirical loss term (related directly to $1 -$ accuracy) and a KL term, optimizing the PAC-Bayes bound does seem to generally decrease the empirical loss term (as seen by an increase in stochastic train accuracy in 1).

### 3.1 Hyperparameter Experiment

To further explore why my results differ in Table 1, I trained models on different hyperparameters of the batch size $b$, and I even try to use the Adam optimizer instead of RMSProp:

| Experiment | $b = 8$ | $b = 16$ | $b = 128$ | Adam |
|---|---|---|---|---|
| SNN Train Error | 0.0302 | 0.0254 | 0.0212 | 0.0146 |
| SNN Test Error | 0.0376 | 0.0334 | 0.0281 | 0.0226 |
| PAC-Bayes Bound | 0.0738 | 0.1471 | 1 | 0.5330 |
| KL($Q\|P$) | 1031 | 5108 | 171318 | 40968 |

Table 2: PAC-Bayes optimization performance with different hyperparameters on the $h = 600$ base model, with a fixed number of 10 epochs.

Remarkably, we see that as the batch size decreases, the PAC-Bayes bound decreases significantly to the same order of magnitude as the SNN test error, which are much tighter than the experimental results obtained in Figure 1 and even in [3]! This also confirms my intuition that the difference in batch size contributed to worse performance for the $h = 1200$ model in 1.

In general, these results suggest the "loss landscape" of the PAC-Bayes objective in Equation 7 is very noisy, and more randomness from gradient descent allows escaping local minima to lower minima. Again, we notice there is no free lunch: while we significantly improve our theoretical bound, both the SNN train and test errors actually increase (slightly) as $b$ decreases.

We also note that using the Adam optimizer produced worse results than sticking with RMSProp.

### 3.2 Can We Recover Determinism?

From these experiments we repeatedly notice that in adding randomness to our models (going from NN to SNN), we are trading away test accuracy for tighter PAC-Bayes guarantees. This is the manifestation of the underlying concept that **randomness is necessary for PAC-Bayes to work**.

Theoretically, recall the errors defined in 3 are themselves expectations over $w \sim Q$, and therefore have good concentration properties. This is what allows the PAC-Bayes bound in 7 to be derived in the first place. Additionally, greater randomness in the posterior $Q$ means lower $\mathrm{KL}(Q\|P)$ (a deterministic delta distribution deviates far more from any prior distribution $P$ than a more spread out distribution), meaning we can more tightly associate the empirical and true errors.

In the case of Gaussians in 8, a deterministic posterior that minimizes $\mathrm{KL}(Q\|P)$ would correspond to low variances $|s|$. However, while there is a $\|s\|_1$ term, there is also a $-1_d \cdot \log s$ term, and the closer $s$ is to zero, the greater the loss contribution from $-1_d \cdot \log s$. In fact, we see the following behavior in 2:

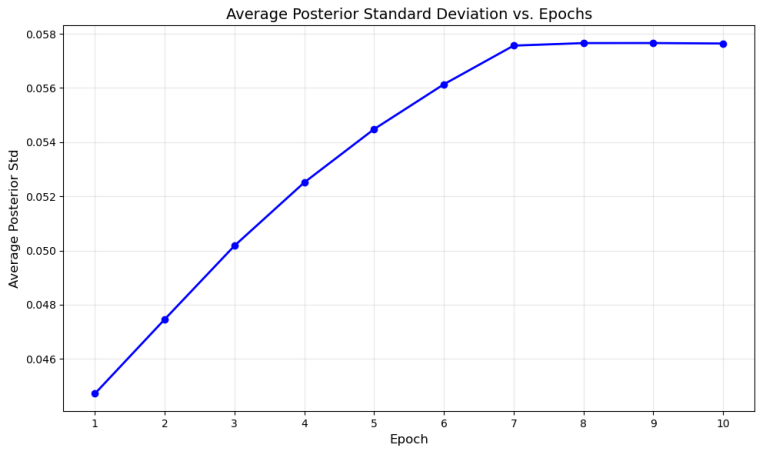


Figure 2: Average standard deviation of the posterior $Q$ vs. epochs. When optimizing the PAC-Bayes objective, instead of moving towards determinism, we actually move towards greater randomness in the posterior.

As a final experiment, I added a small offset of 0.0001 to mitigate the contribution from this term (i.e., I used $-1_d \cdot \log(s + 0.0001 \cdot 1_d)$), and then optimized the PAC-Bayes bound again for the $b = 16$, $h = 600$ base model. The resulting stochastic model achieved an SNN test score of 0.0319 and a PAC-Bayes bound of 0.1864, slightly worse than the model trained without offsets back in Table 1.

## 4 Conclusion

At the slight cost of test accuracy, we can train the randomness of our models and successfully achieve nonvacuous, tight theoretical guarantees of the true error given the empirical error. This extra model calibration step may prove useful in mission-critical machine learning applications where confidence on the true error is a requirement.

Of course, there could be other ways to better optimize the PAC-Bayes bound. In future work, one can experiment with different parametric families of $Q$ and $P$ (for example, using Gaussians with nonzero off-diagonal covariances), or one can derive alternative bounds of the KL inverse that have better convergence properties when optimized under SGD.

# References


[1] Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2022. URL: https://arxiv.org/abs/2107.07511, arXiv:2107.07511.

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

[3] Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data, 2017. URL: https://arxiv.org/abs/1703.11008, arXiv:1703.11008.

[4] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[5] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017. URL: https://arxiv.org/abs/1611.03530, arXiv:1611.03530.


# Appendix

The code used for this project can be found at https://github.com/TheYuch/cse599s-final/.