The Implicit and Explicit Regularization Effects of Dropout

Colin Wei∗ Sham Kakade† Tengyu Ma‡

March 3, 2020

Abstract

Dropout is a widely-used regularization technique, often required to obtain state-of-the-art for a
number of architectures. This work demonstrates that dropout introduces two distinct but entangled
regularization effects: an explicit effect (also studied in prior work) which occurs since dropout modifies
the expected training objective, and, perhaps surprisingly, an additional implicit effect from the stochasticity
in the dropout training update. This implicit regularization effect is analogous to the effect of stochasticity
in small mini-batch stochastic gradient descent. We disentangle these two effects through controlled
experiments. We then derive analytic simplifications which characterize each effect in terms of the
derivatives of the model and the loss, for deep neural networks. We demonstrate these simplified, analytic
regularizers accurately capture the important aspects of dropout, showing they faithfully replace dropout
in practice.

1 Introduction

Dropout is a commonly used regularization technique for neural nets (Hinton et al., 2012; Srivastava et al.,
2014). In NLP, dropout is the norm on both small and large models, as it is much more effective than
methods such as ℓ2 regularization (Merity et al., 2017a). In vision, dropout is often used to train extremely
large models such as EfficientNet-B7 (Tan & Le, 2019).

At training time, dropout sets a random subset of activations to zero, perturbing the network output
with a remarkable amount of noise. Testing is performed on the full model, and it is somewhat mysterious
that dropout works so well despite this difference between train and test. The esoteric nature of dropout has
inspired a large body of work studying its regularization effects: Wager et al. (2013); Helmbold & Long (2015);
Cavazza et al. (2017); Mianjy et al. (2018); Mianjy & Arora (2019) study dropout for linear models, matrix
factorization, and linearized networks; Arora et al. (2020) study deep networks with dropout only at the last
layer. These works primarily study simpler settings than those used in practice, and, as we demonstrate,
there is an implicit regularization effect of dropout that is not addressed by prior work.

A large body of recent work has studied implicit, or algorithmic regularization in deep learning, defined to
be a regularization effect imposed by the training algorithm, not by the objective (see for example (Gunasekar
et al., 2017; Li et al., 2017; Gunasekar et al., 2018b; Arora et al., 2019) and references therein). One notable
element of this is in comparing the generalization performance of SGD vs GD: the implicit regularization effect
of stochasticity in SGD has been empirically studied in the context of small v.s. large batch training Keskar
et al. (2016), where it is observed that noisier small-batch SGD converges to “flatter” local minima which
generalize better, whereas large-batch SGD converges “sharper” local minima which generalize more poorly.
The starting point of this work is observing that in practice, dropout also introduces an implicit source of
regularization because it adds noise to the gradient updates (somewhat analogous to the small v.s. large batch

∗Stanford University, email: colinwei@stanford.edu
†Microsoft Research & University of Washington, email: sham@cs.washington.edu
‡Stanford University, email: tengyuma@stanford.edu
training). Prior studies of dropout only analyze its explicit regularization effect, focusing on how it modifies the expected loss\(^1\). Understanding dropout in practical settings requires studying both regularization effects. This paper focuses on a sharp characterization of the regularization effects in dropout, where we:

- disentangle and analytically characterize the explicit and implicit regularization effects of dropout.
- derive simplified, analytical, and interpretable regularizers which completely replace dropout for language modeling tasks.

More concretely, this work makes the following contributions:

1. This work empirically shows that dropout provides both explicit and implicit regularization effects. Dropout modifies the expected training objective, and it is natural to define the explicit regularizer as the difference between the expected training objective and the standard objective, as follows:

\[
R(F) = \mathbb{E}_x \left[ \mathbb{E}_{\text{drop}} [\ell(F_{\text{drop}}(x))] \right] - \mathbb{E}_x [\ell(F(x))]
\]

Here \(F_{\text{drop}}\) denotes the dropout model and \(\text{drop}\) denotes the randomness from dropout. Moreover, the optimization uses a stochastic approximation of the expected training loss by sampling the dropout noise, which gives rise to an implicit regularization effect.

In practice, the two regularization effects are entangled and easy to conflate. Section 3 provides results of experiments which disentangle these effects.

2. We then distill these two regularization effects, providing simpler and more interpretable regularizers that depend on the derivatives of the model and loss (Section 4). Intuitively, dropout regularizes the stability of the model and loss output evaluated on each training datapoint. Theoretically (in Section 4.3), we provide a generalization bound which helps justify the dependencies of these regularizers on the loss derivatives.

3. Empirically, detailed experiments are provided in Section 5 showing that these simplified, analytical regularizers can faithfully match and replace dropout for both LSTM and Transformer architectures, on the Penn Treebank, Wikitext-2, and Wikitext-103 datasets. To our knowledge, these are the most accurate empirical demonstrations of theory matching practice with regards to the analysis of dropout.\(^2\)

4. Finally, the form of the derived explicit regularizer provides detailed intuition on how to regularize the stability of a deep model. When the number of output classes (i.e. vocabulary in language modeling) is large, dropout regularizes most heavily the stability of predictions corresponding to classes to which the model assigns a prediction probability that is not too certain (i.e., not close to either 0 or 1). Our ablation experiments in Section 5.2 reveal this is critical for the effectiveness of dropout, and our theory in Section 4.3 offers additional justification for this perspective.

More generally, we hope that the precise methodological derivations that we provide can inform the future study and derivation of data-dependent regularizers in deep learning.

## 2 Preliminaries

**Notation.** For a function \(f(a) : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}\) on a vector \(a\), we use \(D^k\left[a\right]f(b) \in \mathbb{R}^{d_2 \times d_1}\) to denote the \(k\)-th derivative of \(f\) with respect to the variable \(a\) (with a bold subscript to distinguish the variable from its value) evaluated at \(b\). We drop the subscript \(a\) when the emphasis is unnecessary. Let \(f \circ g\) denote the composition of \(f\) with \(g\). For vector \(v\), let diag\((v)\) denote the matrix with entries of \(v\) on its diagonal and 0 elsewhere. Let tr\((M)\) denote the trace of \(M\). For matrices \(M_1, M_2\), let \(\langle M_1, M_2 \rangle\) denote the inner product of their vectorizations. For a vector \(v\), we use \((v)_i\) to denote the \(i\)-th coordinate of \(v\), dropping the parenthesis when it is clear from context. For vectors \(v_1, v_2\), \(v_1 \odot v_2\) refers to their entrywise product. We let \(v_1^{(2)} \doteq v_1 \odot v_1\). Let \(\mathbb{1}\) denote the all 1’s vector. Throughout the paper, we consider a neural network \(F\) with weights \(W\) and a loss function \(\ell : \mathbb{R}^c \times [c] \to \mathbb{R}\), where \(c\) is the number of classes. We omit the dependence

---

\(^1\) Prior work [Mianjy et al., 2018] refers to this as the “implicit bias” of dropout. We refer to this as explicit regularization and reserve the term “implicit” to mean algorithmic regularization effect which does not change the objective.

\(^2\) Our code is available at [https://github.com/cwein3/dropout-analytical](https://github.com/cwein3/dropout-analytical).
Figure 1: **Averaging dropout noise degrades performance.** Perplexity vs. epoch of LSTMs trained with mini-batch dropout, \( \text{DROPOUT}_k \) for various \( k \) (see Algorithm 1). Training perplexity is evaluated without dropout. Increasing the number of samples of dropout noise, \( k \), reduces the amount of update noise arising from the stochasticity of dropout. Though the training objective does not change with \( k \), as \( k \) increases, the validation performance degrades. This suggests that the update noise from dropout provides an implicit regularization effect. \( \text{Left: Penn Treebank. Right: WikiText-2.} \)

### Algorithm 1 \( \text{DROPOUT}_k \), mini-batch dropout update using \( k \) samples of noise.

**Input:** Training examples \( \{x_i\}_{i=1}^m \).

Sample noise \( \eta_j \) for \( i \in [m], j \in [k] \).

Compute \( g = \nabla_W \left( \frac{1}{m} \sum_{i=1}^m \ell_{\text{drop}, k}(F, x_i, \{\eta_j\}_{j=1}^k) \right) \). \( \Rightarrow \) Use \( g \) for optimization algorithm.

on the weights \( W \) when it is clear from context. We use \( x \) and \( y \) to denote inputs and labels. The loss is computed via \( \ell(F(x), y) \), where we hide \( y \) when it is clear from context.

**Dropout.** The most common variant of dropout, node dropout [Hinton et al., 2012; Srivastava et al., 2014], randomly sets hidden activations in a given layer to 0. Formally, for some probability \( q \) and layer \( h \in \mathbb{R}^d \) (i.e. \( h \) is the vector of outputs of some layer), we sample a scaling vector \( \eta \in \mathbb{R}^d \) with independent random coordinates:

\[
(\eta)_k = \begin{cases} 
-1 & \text{with probability } q \\
\frac{q}{1-q} & \text{with probability } 1-q 
\end{cases}
\]

Here \( k \) indexes a coordinate of \( h \). Note that \( \eta \) is a zero mean random variable. We then apply dropout by computing

\[
h_{\text{drop}} = (1 + \eta) \odot h
\]

and using \( h_{\text{drop}} \) instead of \( h \). With slight abuse of notation, we let \( \eta \) denote the collection of such vectors over all layers. \( F(x, \eta) \) denotes the output of model \( F \) on input \( x \) using dropout noise \( \eta \).

# 3 Disentangling Explicit and Implicit Regularization in Dropout

We now present an experimental study designed to disentangle the two regularization effects, which confirms the existence of implicit regularization in dropout. Furthermore, this approach allows us to study each effect in isolation.

Let \( L(F) = \mathbb{E}_x[\ell(F(x))] \) denote the population loss without dropout. This is the test criterion regardless of whether dropout is used during training. However, dropout modifies the expected training objective even conditioned on a fixed example \( x \). The training loss of an example \( x \) averaged over the dropout noise \( \eta \) (defined in Section 2) is

\[
\ell_{\text{drop}}(F, x) = \mathbb{E}_\eta[\ell(F(x, \eta))] \neq \ell(F(x)) \tag{3.1}
\]

Consequently, the expected training objective also differs from \( L(F) \):

\[
L_{\text{drop}}(F) = \mathbb{E}_x[\ell_{\text{drop}}(F, x)] \neq L(F)
\]

3
Figure 2: **Confirming implicit regularization effect.** Validation perplexity vs. epoch of LSTMs trained with Dropout 1, Dropout 4, and Dropout 4 with noise added via the procedure in Section 3.1. By adding noise to Dropout 4, we recover the performance of Dropout 1. Thus, the noise we add has an implicit regularization effect. **Left:** Penn Treebank. **Right:** WikiText-2.

It is natural to define the explicit regularizer as the difference between the expected training objective (averaged over both $x$ and $\eta$) and the standard objective, i.e.

$$R_{\text{drop}}(F) = L_{\text{drop}}(F) - L(F).$$

Due to the fact that in practice, we only have access to a finite training sample (and not the population), it is helpful to define explicit regularizer on a single example as follows:

$$R_{\text{drop}}(F, x) = \ell_{\text{drop}}(F, x) - \ell(F(x)).$$

Previous work studies the analytical forms or properties of these regularizers for various models. However, in practice, $\ell_{\text{drop}}(F, x)$ (and its gradient $\nabla W \ell_{\text{drop}}(F, x)$) are only stochastically estimated by sampling a single $\eta$ and computing $\ell(F(x, \eta))$ (and $\nabla W \ell(F(x, \eta))$ respectively). For example, SGD (with mini-batch size 1), performs the update:

$$W \leftarrow W - \gamma \nabla W \ell(F(x, \eta)),$$

where $\gamma$ is the stepsize, $x$ is a randomly sampled datapoint, and $\eta$ is a randomly sampled dropout noise variable.

We demonstrate that the stochasticity from sampling $\eta$ provides an implicit regularization effect which contributes to the test-time effectiveness of dropout\(^3\). Our strategy for disentangling the regularization effects is simple: we remove noise from the gradient estimate by optimizing a more accurate estimate of $\ell_{\text{drop}}(F, x)$ than $\ell(F(x, \eta))$. Formally, we can perform “mini-batch” dropout by averaging the loss over $k$ samples of the noise $\{\eta_i\}_{i=1}^{k}$:

$$\hat{\ell}_{\text{drop}, k}(F, x, \{\eta_i\}_{i=1}^{k}) = \frac{1}{k} \sum_{i=1}^{k} \ell(F(x, \eta_i)) \tag{3.2}$$

For training, we now use the stochastic gradient $\nabla W \hat{\ell}_{\text{drop}, k}$, reducing the gradient covariance by a factor of $k$. We refer to the mini-batched dropout update by Dropout\(_k\) as shorthand and formally describe it in Algorithm I.

If there were no implicit regularization from the stochasticity of dropout, then we would expect Dropout\(_k\) to have similar test performance to Dropout\(_1\), which is equivalent to standard dropout. In Figure 1, we plot the validation accuracy vs. training steps for models trained using Dropout\(_k\) for various values of $k$. Figure 1 shows that, perhaps surprisingly, performance degrades quite sharply for larger choices of $k$. However, the explicit regularizer is still helpful, as Dropout\(_{32}\) does not overfit as severely as the model.

---

\(^3\)There is also an implicit regularization effect from sampling the SGD minibatch. As the minibatch size is fixed in our experiments, this is distinct from the implicit regularization effect of dropout demonstrated in Figure 1 and studying it is orthogonal to our work.
trained without dropout (for Penn Treebank, the best perplexity without dropout is around 120, which is outside the bounds of the graph). $\text{DROPOUT}_k$ and $\text{DROPOUT}_1$ optimize the same expected objective, so the change in algorithm must be the cause of these performance discrepancies.

### 3.1 Injecting Dropout Noise Fixes $\text{DROPOUT}_k$

Our proposed explanation for Figure 1 is that the gradient noise induced by dropout provides an implicit regularization effect. We verify this constructively by adding noise to the $\text{DROPOUT}_k$ updates in order to recover the performance of standard dropout. Let $\xi_{\text{drop}}$ denote the fluctuation of the stochastic dropout gradient around its mean:

$$\xi_{\text{drop}}(F, x; \eta) = \nabla_W \ell(F(x, \eta)) - \nabla_W \ell_{\text{drop}}(F, x)$$

(3.3)

Note that $\xi_{\text{drop}}$ is exactly the gradient noise in standard dropout. Furthermore, we have $\text{Cov}(\nabla_W \ell_{\text{drop}}, k) = \frac{1}{k} \text{Cov}(\xi_{\text{drop}})$. To correct the covariance of the $\text{DROPOUT}_k$ gradient, we will add mean-zero noise with covariance $(1 - \frac{1}{k}) \text{Cov}(\xi_{\text{drop}})$. Let $\eta_1$ and $\eta_2$ be two independent draws of the dropout noise. Define $\tilde{\xi}_{\text{drop}}$ as:

$$\tilde{\xi}_{\text{drop}}(F, x; \eta_1, \eta_2) = \nabla_W \ell(F(x, \eta_1)) - \nabla_W \ell(F(x, \eta_2))$$

Note that $\text{Cov}(\tilde{\xi}_{\text{drop}}) = 2\text{Cov}(\xi_{\text{drop}})$. Thus, by adding the term $\sqrt{\frac{1}{2}(1 - \frac{1}{k})}\tilde{\xi}_{\text{drop}}$ to the $\text{DROPOUT}_k$ gradient, we obtain a gradient estimate with the same covariance as $\xi_{\text{drop}}$.

In Figure 2, we verify that this correction procedure recovers the test performance of $\text{DROPOUT}_1$. Thus, we have constructed a (complicated) implicit regularizer which explains the discrepancy between $\text{DROPOUT}_k$ and $\text{DROPOUT}_1$. In Section 4.2, we will explore its simplifications.

### 4 Characterizing the Dropout Regularizers

Having disentangled the explicit and implicit regularization effects of dropout, we will now study them separately. In this section, we adapt the analysis tools of (Wager et al., 2013) to study both regularization effects for neural networks. We derive analytic simplifications for both regularizers in terms of the model and loss derivatives. At a high level, our derivations show that dropout regularizes the data-dependent stability of the model and loss on the training examples. This demonstrates a key difference between dropout and $\ell_2$ regularization, which is agnostic to the data.

In Section 4.1, we present and derive our explicit regularizer. In Section 4.2, we derive an update noise distribution which captures the implicit regularization effect in dropout. In Section 4.3, we prove a generalization bound for the cross-entropy loss which further justifies our stability-based regularizers. In Section 5, we empirically demonstrate that our derivations accurately capture the regularization effects in dropout – we can match the performance of dropout for language modeling tasks by using only our regularizers.

For simplicity, we focus on node dropout (Hinton et al., 2012; Srivastava et al., 2014), though our analysis applies to variants such as DropConnect as well (Wan et al., 2013).

#### 4.1 Characterizing the Explicit Regularizer

**Single-layer Dropout.** For simplicity, we start by considering node dropout applied to a single layer $i$ of the network. For the rest of the paper, we use $h_i$ to denote the $i$-th hidden layer of the network and let $F_i$ denote the composition of the layers after $h_i$, that is, the function that takes in $h_i$ as input, and outputs the model prediction. (Thus, $F_i(h_i) = F(x)$).
We apply Taylor expansion around $\delta = 0$:

$$
\ell(F(x, \eta)) - \ell(F(x)) \approx D_{h_i}(\ell \circ F_i)[h_i]\delta + \frac{\delta^T (D_{h_i}^2(\ell \circ F_i)[h_i])\delta}{2}
$$

This provides an approximate version of the dropout explicit regularizer $R_{\text{drop}}$:

$$
R_{\text{drop}}(F, x) = E_{\eta}[\ell(F(x, \eta))] - \ell(F(x)) \\
\approx \frac{1}{2} E_{\delta}[\delta^T (D_{h_i}^2(\ell \circ F_i)[h_i])\delta]
$$

Here the expectation over the linear term in (4.1) vanished because $\delta = \eta_i \odot h_i(x)$ is a mean-zero vector.

Next we take expectation over $\delta$:

$$
E_\delta[\delta^T D_{h_i}^2(\ell \circ F_i)[h_i]\delta] \\
= \langle D_{h_i}^2(\ell \circ F_i)[h_i], \frac{E[\delta\delta^T]}{2} \rangle \\
= \frac{q}{2(q-1)} \langle D_{h_i}^2(\ell \circ F_i)[h_i], \text{diag}(h_i(x)^{\odot 2}) \rangle
$$

(4.2)

where $q$ is the dropout probability and we used the fact that $E[\delta\delta^T] = \frac{q}{q-1}\text{diag}(h_i(x)^{\odot 2})$ because $\delta = \eta_i \odot h_i(x)$ and the coordinates of $\eta_i$ are independent.

We obtain an analytical approximation for the explicit regularizer $R_{\text{drop}}(F, x)$ by combining the equations above. Next we will rewrite the RHS of (4.2) in a more interpretable form by further dropping some terms.

For notational simplicity, let $J_{F,i}(x)$ be the Jacobians of the network output with respect to the hidden layers, and $H_{\text{out}}(x)$ be the Hessian of the loss with respect to the network outputs:

$$
J_{F,i}(x) \triangleq D_{h_i} F_i[h_i(x)] \text{ and } H_{\text{out}}(x) \triangleq D_F^2 \ell(F(x))
$$

We claim that $R_{\text{drop}}(F, x)$ (or the RHS of (4.2)) can be replaced by the following analytical form

$$
R_{\text{approx}}^i(F, x) \triangleq \langle J_{F,i}(x)^T H_{\text{out}}(x) J_{F,i}(x), \text{diag}(h_i(x)^{\odot 2}) \rangle
$$

(4.3)

Readers may find this reminiscent of the decomposition of the Hessian of neural nets loss $\text{LeCun et al. (2012)}$; $\text{Sagun et al. (2017)}$. Indeed, we decompose $D_{h_i}^2(\ell \circ F_i)[h_i]$ into two terms, and drop the non-PSD term that depends on the Hessian of the model (which is less suitable as a regularizer and has been argued to be less important empirically $\text{Sagun et al. (2017)}$). A full derivation and justification is given in Section A.1.

**Multi-layer Dropout.** To deal with dropout on all layers, we simply take Taylor expansion with all the $\delta$’s at every layer. Cross terms cancel because the masks of different layers are independent, and the resulting regularizer is a sum of equation (4.3) over $i$, giving our analytical explicit regularizer:

$$
R_{\text{approx}}(F, x) \triangleq \sum_i \langle J_{F,i}(x)^T H_{\text{out}}(x) J_{F,i}(x), \text{diag}(h_i(x)^{\odot 2}) \rangle
$$

(4.4)

**Interpretation.** Our regularizer ensures that the Jacobians and hidden layers of the model output are small when measured in the norm of $H_{\text{out}}$. We note that for cross entropy loss, $H_{\text{out}}(x) = \text{diag}(p) - pp^T \succeq 0$, where

---

4Taylor expansion typically requires a small level of perturbation, which may not hold if the dropout probability is large. In Section A.2, we argue that performing Taylor expansion around the next layer could remedy this issue. As it does not change the final result, we omit this analysis here.
$p$ is the probability vector predicted by the model encoding the distribution over output class labels. As the diagonal entries take the form $p_k(1-p_k)$, this Hessian places stronger emphasis on output classes which the model believes are plausible but not certain. Our experiments in Section 5.2 demonstrate that this particular weighting is an important factor for the success of dropout – alternative ways to weight the stability of each output class in the regularizer do not perform as well.

Keskar et al. (2016); Yao et al. (2018); Jastrzebski et al. (2018) study the relationship between SGD batch size and notions of “flatness” of local minima via metrics related to the magnitudes of the eigenvalues of the second derivative of the loss with respect to the model parameters. They observe that flatter local minima tend to correlate with better generalization. Our regularizer encourages a notion of flatness that depends on the second derivative of the loss with respect to the hidden layers.

4.2 Characterizing the Implicit Regularization Effect

In this section, we derive a gradient noise distribution which can replace the mean-zero gradient noise in dropout, $\xi_{\text{drop}}$.

**Single Layer Dropout.** As before, we start by considering the single-layer case. Instead of directly approximating $\xi_{\text{drop}}$, which involves the intractable term $\nabla_W \ell_{\text{drop}}(F, x)$, we aim to approximate the noise $\tilde{\xi}_{\text{drop}}$ defined in Section 3.1 which we showed to be able to replace $\xi_{\text{drop}}$.

We apply Taylor expansion to approximate $\tilde{\xi}_{\text{drop}}$, only keeping the mean-zero linear terms. Letting $\delta_1 = \eta_i^{(1)} \odot h_i(x)$ and $\delta_2 = \eta_i^{(2)} \odot h_i(x)$ denote two different perturbations to the $i$-th layer, we have:

$$
\tilde{\xi}_{\text{drop}}(F, x, \eta_i^{(1)}, \eta_i^{(2)}) = \nabla_W \left( \ell(F_i(h_i + \delta_1)) - \ell(F_i(h_i + \delta_2)) \right) \\
\approx \nabla_W \left( J_{\text{loss},i}(x)(\eta_i^{(1)} - \eta_i^{(2)}) \odot h_i(x) \right)
$$

where $J_{\text{loss},i}(x)$ denotes the Jacobian of the loss with respect to the hidden layers: $J_{\text{loss},i}(x) \doteq \nabla h_i(\ell \circ F_i)[h_i(x)]$. Now we can replace the difference $\eta_i^{(1)} - \eta_i^{(2)}$ by $\eta_i \sqrt{2}$, as the covariance is unchanged. After adjusting the scaling to match the covariance of $\xi_{\text{drop}}$, we obtain the following analytic form for update noise:

$$
\xi_{\text{approx}}(F, x, \eta_i) \doteq \nabla_W \left( J_{\text{loss},i}(x)(\eta \odot h_i(x)) \right) \label{eq:4.5}
$$

**Multi-layer Dropout.** To handle multi-layer dropout, we Taylor expand over all the layers, obtaining a sum of (4.5) over the layers:

$$
\xi_{\text{approx}}(F, x, \{\eta_i\}) \doteq \nabla_W \left( \sum_i J_{\text{loss},i}(x)(\eta_i \odot h_i(x)) \right) \label{eq:4.6}
$$

---

5 As the subscript has been used to index the layer, we use the superscript to index the different dropout noise samples.

6 To make tuning slightly simpler, we compute the noise by sampling the coordinates of $\eta_i$ uniformly from $\{-1, +1\}$ and scaling by $\sqrt{\frac{2}{k-1}}$, as this preserves the covariance.
Additionally, we define the norm bound of the derivatives (the loss Hessians) of the model Jacobian with the loss Hessian. More precise interpretations are left for future work.

In Section 5, we demonstrate that a combination of our explicit and implicit regularizer can successfully replace dropout. The general update rule which applies these regularizers in lieu of dropout is described in Algorithm 2.

4.3 Theoretical Support for Stability-based Regularization

Recent works [Arora et al., 2018, Nagarajan & Kolter, 2019, Wei & Ma, 2019] support our stability-based regularization by bounding generalization of the model in terms of its Jacobian norms on the training data. These bounds align with the Jacobian terms in the regularization (4.4). However, they miss a crucial aspect of the regularizers derived in Section 4.1 as they only consider derivatives of the model output, ignoring the loss derivatives (the $H_{\text{out}}(x)$ term in equation (4.4)). Though this is a subtle distinction, in Section 5.2 we demonstrate that the loss derivatives are necessary on language modeling tasks.

In this section, we prove a new generalization bound for cross entropy loss on linear models. Our bound helps further justify the forms of our regularizers in (4.4) and (4.6), as every term in our bound is scaled by a derivative of the loss.

Let $\ell_{ce}$ denote the standard cross-entropy loss on $c$ classes with true label $y$. For linear models parameterized by weight matrix $W$, we compute the loss by

$$\ell_{ce}(Wx) = -\log \frac{\exp((Wx)_y)}{\sum_y \exp((Wx)_y)}$$

Let $\overline{\ell_{ce}} = \min\{B, \ell_{ce}\}$ denote the truncation of the cross-entropy loss to some fixed bound $B > 0$. For matrix $M$, define the following $\| \cdot \|_{2,1}$-norm of $M$: $\|M\|_{2,1} = \sum_i \sqrt{\sum_j (M_{ij}^2)}$. Let $P$ denote the population data distribution and $P_n$ the distribution over training samples. We have the following generalization bound:

**Theorem 4.1.** With probability $1 - \delta$ over the training examples, for all weight matrices $W$ satisfying the norm bound $\|W\|_{2,1} \leq A$, the following holds:

$$\mathbb{E}_P[\overline{\ell_{ce}}(Wx)] - 1.01\mathbb{E}_{P_n}[\overline{\ell_{ce}}(Wx)] \leq \left(\frac{A\mu(W)^\frac{3}{2}(\theta B)^\frac{1}{2}}{n^{\frac{3}{2}}} + \frac{BA^2\theta}{\sqrt{n}} + \frac{B A^2\theta}{n(\log^2 (\frac{B A^2\theta}{n}) + 1)} + \zeta\right).$$

Here $\mu(W), \nu(W)$ measure the Jacobians and Hessians of the loss and are defined by

$$\mu(W) = \mathbb{E}_{P_n}[\|D\ell_{ce}(Wx)\|_2]$$
$$\nu(W) = \mathbb{E}_{P_n}[\text{tr}(D^2\ell_{ce}(Wx))]$$

(4.7)

Additionally, we define $\theta = \log^3(nc) \max_i \|x_i\|^2$ and $\zeta = B(\log(1/\delta) + \log \log n)$ is a low order term.

We provide the proof in Section 6. Theorem 4.1 helps justify regularizing the loss derivatives, showing that even if the weights are large, one can guarantee good generalization by ensuring that the loss Hessians...
Algorithm 2 The general form of update for combinations of our explicit and implicit regularizers.

**Input:** minibatch \( \{x_i\}_{i=1}^m \), explicit regularizer \( R \), gradient noise \( \xi \), regularization strengths \( \lambda_1, \lambda_2 \).

- Compute \( g = \frac{1}{m} \sum_{i=1}^{m} \nabla_W (\ell(F(x_i)) + \lambda_1 R(F, x_i)). \)
- Update \( g = g + \frac{1}{m} \sum_{i=1}^{m} \lambda_2 \xi(F, x_i). \)

\( \Rightarrow \) Use \( g \) for optimization algorithm.

Figure 3: **Our explicit regularizer v.s. dropout.** Validation perplexity vs. epoch of LSTMs trained with DROPOUT\(_1\), DROPOUT\(_{32}\) (see Algorithm 1), and our explicit regularizer only. Our explicit regularizer \( (4.4) \) outperforms DROPOUT\(_{32}\) but does not match DROPOUT\(_1\) since it is missing the implicit regularization benefit of dropout. **Left:** Penn Treebank. **Right:** WikiText-2.

and Jacobians are small. Note that the third term in the bound can be independent of the weight matrix norms if the loss Hessian is sufficiently small: when the data and weights are well-aligned, the third term can be as small as \( O\left( \frac{B \log^2(nc)}{n} \right) \) (see Section B.4). In contrast, prior bounds for this setting contain a term scaling with some power of \( \|W\|/\sqrt{n} \) (Kakade et al., 2009; Srebro et al., 2010). This scaling suggests using \( \ell_2 \) regularization, which does not work for language modeling (see Table 3 and 4 in Section D).

5 Experiments

In this section, we empirically confirm that our derivations in Section 4 provide accurate characterizations of dropout. Our focus is on language modeling tasks using the LSTM and Transformer architectures.

5.1 Our Derived Regularizers can Replace Dropout

In this section, we show that the regularizers derived in Section 4 can replace dropout for LSTMs on language modeling tasks. We work with Penn Treebank (Marcus et al., 1994), a corpus of 887,521 tokens and Wikitext-2 (Merity et al., 2016), a corpus of 2,088,628 tokens. In Section 5.3, we study whether our findings can also scale to larger datasets and architectures such as Transformer-XL (Dai et al., 2019).

For the experiments in this section, we base our model and code on Merity et al. (2017a, 2018). For the dropout-trained models, we use node dropout on the output, hidden, and embedding layers as well as DropConnect (Wan et al., 2013) on the weight matrices. We fix the dropout probability to \( q = 0.4 \) for these experiments. To compute the update gradients for our regularizers, we follow the general rule described in Algorithm 2. We specify additional hyperparameters in Section D.

We study three settings described in detail below: our explicit regularizer \( R_{\text{approx}} \) only, adding our noise \( \xi_{\text{approx}} \) to DROPOUT\(_k\) updates, and combining our explicit and implicit regularizers. Tables 3 and Tables 4 in Section D summarize the experimental results on our regularizers for the Penn Treebank and Wikitext-2 datasets. We obtain our results without tuning, as we use the regularization coefficient suggested in Section 4 to match the dropout strength. The Jacobian optimization required for the analytical regularizers results in around 3x runtime slowdown compared to dropout, though we note that the analytical regularizers appear to optimize in fewer iterations (see Figure 5).

9
Figure 4: Our implicit regularizer v.s. dropout. Validation perplexity vs. epoch of LSTMs trained using mini-batch dropout, Dropout\(k\), with injection of noise \(\xi_{\text{approx}}\) (Algorithm 4). For reference, we also plot Dropout\(_1\) and Dropout\(_8\) with no noise (Algorithm 1). Dropout\(_k\) with noise injection matches Dropout\(_1\) for \(k = 2, 4, 8\), affirming that our noise distribution \(\xi_{\text{approx}}\) captures the implicit regularization effect of dropout noise. Left: Penn Treebank. Right: WikiText-2.

Figure 5: Our combined regularizer v.s. dropout. Validation perplexity vs. epoch of LSTMs trained with our regularizers vs standard dropout. Our regularizers can match dropout and appear to improve the validation perplexity faster. Left: Penn Treebank. Right: WikiText-2.

Replacing Dropout Explicit Regularization. In Figure 3, we compare our explicit regularizer (4.4) to mini-batch dropout, Dropout\(_k\), with \(k = 1, 32\). For \(k = 32\), the implicit regularization effect of dropout is heavily reduced, bringing the training procedure closer to training on \(\ell_{\text{drop}}\) exactly. Our explicit regularizer outperforms Dropout\(_{32}\), confirming that it matches the explicit regularization effect of dropout. It does not match the performance of Dropout\(_1\) because it is missing the implicit regularization effect.

Replacing Dropout Implicit Regularization. We demonstrate that the update noise derived in (4.6) can effectively replicate the implicit regularization effect of dropout. We inject appropriately scaled \(\xi_{\text{approx}}\) noise into the Dropout\(_k\) training procedure. As the covariance of \(\nabla W \hat{\ell}_{\text{drop},k}\) scales with \(\frac{1}{k}\), we scale \(\xi_{\text{approx}}\) by a factor \(\sqrt{1 - \frac{1}{k}}\). Thus, if \(\xi_{\text{approx}}\) and \(\xi_{\text{drop}}\) had the same covariance, the covariance of the updates would remain constant across \(k\). Algorithm 4 in Section C formally describes this procedure. In Figure 4, we demonstrate that this procedure can closely track the performance of Dropout\(_1\) for various values of \(k\), affirming that \(\xi_{\text{approx}}\) captures essential properties of \(\xi_{\text{drop}}\).

Completely Replacing Dropout. We demonstrate that the combination of our regularizers can completely replace dropout. We apply algorithm 2, setting \(R = R_{\text{approx}}\) and \(\xi = \xi_{\text{approx}}\). In Figure 5, we plot the validation perplexity vs. time of a model trained with our regularization vs. Dropout\(_1\). Figure 5 demonstrates that our regularization is enough to replace dropout, confirming the validity of our derivations. We note that our regularizer appears to require fewer iterations to decrease the validation perplexity. This raises the exciting possibility of designing more efficient regularizers than dropout, which we leave for future work.
Table 1: Regularization effect of $\tilde{R}_{\text{approx}}$ (see (5.1)) on Penn Treebank with and without implicit regularization. $\tilde{R}_{\text{approx}}$ can significantly outperform $\ell_2$ regularization but does not match dropout even with implicit regularization, whereas $R_{\text{approx}}$ can.

<table>
<thead>
<tr>
<th>Training Method</th>
<th>Best Val. Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell_2$ reg (tuned)</td>
<td>112.04</td>
</tr>
<tr>
<td>No implicit</td>
<td></td>
</tr>
<tr>
<td>$\tilde{R}_{\text{approx}}$ (tuned)</td>
<td>84.06</td>
</tr>
<tr>
<td>$R_{\text{approx}}$ (4.4)</td>
<td>84.52</td>
</tr>
<tr>
<td>With implicit</td>
<td></td>
</tr>
<tr>
<td>$\tilde{R}<em>{\text{approx}}$ (tuned) and $\xi</em>{\text{approx}}$ (4.5)</td>
<td>79.54</td>
</tr>
<tr>
<td>$R_{\text{approx}}$ and $\xi_{\text{approx}}$ (4.5)</td>
<td>72.99</td>
</tr>
</tbody>
</table>

5.2 Regularizing the Loss Hessian is Necessary

We argue that simply regularizing the stability of the model outputs is not sufficient. As argued in Section 4.1, our derivations show that dropout enforces stronger stability for output coordinates where the model assigns non-trivial probability mass but is not extremely confident. To demonstrate this is helpful, we experiment with replacing $H_{\text{out}}$ in our explicit regularizer (see (4.4)) with two alternative quantities.

**Identity Cannot Replace Loss Hessian.** For the first variant, we use an identity matrix instead of the loss Hessian (so the regularizer weights each output coordinate equally). We provide implementation details in Section C. On Penn Treebank, this was ineffective: after thoroughly tuning the regularization strength, the best validation accuracy we obtained was 108.76, which is comparable to the performance of $\ell_2$ regularization and much worse than dropout.

**Using the Loss Jacobian Instead of Hessian.** For cross entropy loss, in the case where the model predicts the true label very confidently, the loss Hessian $H_{\text{out}}(x)$ approaches the outer product of the loss Jacobian with itself: $H_{\text{out}}(x) \approx D_F \ell'(F(x))^\top D_F \ell'(F(x))$ (see Section C.1). Substituting this approximation into our explicit regularizer gives the following alternative regularizer:

$$\tilde{R}_{\text{approx}}(F, x) \triangleq \sum_i J_{\text{loss}, i} \text{diag}(h_i(x) \odot 2) J_{\text{loss}, i}^\top$$ (5.1)

On Penn Treebank, we find that this regularizer is much more effective than $\ell_2$ regularization but cannot match dropout. We test whether $\tilde{R}_{\text{approx}}$ performs well as $R_{\text{approx}}$ with or without implicit regularization. In both cases, we tune the explicit regularization strength. Table 1 summarizes the results compared to the Hessian-based regularizer. $\tilde{R}_{\text{approx}}$ on its own significantly outperforms $\ell_2$ regularization and can match $R_{\text{approx}}$ after tuning. However, with update noise it does not match dropout or $R_{\text{approx}}$ (even after tuning).

5.3 Additional Settings

We test how well our findings translate to larger datasets and different architectures. We use the Wikitext-103 dataset, which contains 103,227,021 tokens, and the Transformer-XL [Dai et al., 2019] and QRNN [Bradbury et al., 2016] architectures. First, we explore whether the implicit regularization effect of dropout is as important on larger datasets. We train the Transformer-XL and QRNN architectures on the Wikitext-103 corpus using DROPOUT$_k$ for $k = 1, 2, 4$. Table 2 shows that for Transformer-XL trained on the full dataset, the implicit regularization effect disappears. We observe the same for QRNN (see Section D).

In Table 2, we also demonstrate that there is an implicit regularization effect when we downsample Wikitext-103 by a factor of 5, though it is not as crucial. Thus, the importance of the implicit regularization depends on the dataset size.

Finally, we confirm that our explicit regularizer is effective on a larger dataset. For Wikitext-103 and Transformer-XL, Table 2 shows that our explicit regularizer achieves validation perplexity of 24.12, within 0.7 of dropout.
Table 2: Experimental results on WikiText-103 dataset for Transformer architecture. The implicit regularization effect of dropout noise appears to decrease with dataset size.

<table>
<thead>
<tr>
<th>Dataset Size</th>
<th>Training Method</th>
<th>Best Val. Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Dataset</td>
<td>No regularization</td>
<td>29.45</td>
</tr>
<tr>
<td></td>
<td>DROPOT^1</td>
<td>23.39</td>
</tr>
<tr>
<td></td>
<td>DROPOT^2</td>
<td>23.13</td>
</tr>
<tr>
<td></td>
<td>DROPOT^4</td>
<td>23.14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rapprox</th>
<th>DROPOT^1</th>
<th>24.12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2× Dataset</td>
<td>DROPOT^2</td>
<td>46.05</td>
</tr>
<tr>
<td></td>
<td>DROPOT^4</td>
<td>47.40</td>
</tr>
</tbody>
</table>

6 Related Work

Dropout has been the focus of several theoretical works studying its properties for both optimization and generalization (Wager et al., 2013; Baldi & Sadowski, 2013; Helmbold & Long, 2015; Gal & Ghahramani, 2016a; Helmbold & Long, 2017; Cavazza et al., 2017; Mianjy et al., 2018; Mianjy & Arora, 2019; Arora et al., 2020; Wang & Manning, 2013; Maeda, 2014; Gal & Ghahramani, 2016a; Ma et al., 2016) study dropout from a Bayesian perspective. Gao et al. (2019) empirically study the effect of applying dropout masks in one only direction of the network (either the forward or backward pass).

Wager et al. (2013); Helmbold & Long (2015) use a Taylor expansion to analyze dropout in linear models, and our work extends their analysis to neural networks. Cavazza et al. (2017); Mianjy et al. (2018); Mianjy & Arora (2019) study the expected dropout objective for matrix factorization and linearized neural nets, respectively. Recent work (Arora et al., 2020) studies dropout applied to only the last layer of a deep neural net, computing an exact expression for the explicit regularizer which depends on the magnitudes of coordinates in the last hidden layer. Our analysis of the explicit regularization results in a more general expression which contains similar quantities, but considers dropout at all layers of the network. These prior works focus on explicit regularization and do not study the implicit regularization effects of dropout.

There has been a large body of prior work studying the relationship between gradient noise and generalization (Keskar et al., 2016; Keskar & Socher, 2017; Smith & Le, 2017; Jastrzebski et al., 2018; Xing et al., 2018; Li et al., 2019; Chaudhari & Soatto, 2018) study how the noise distribution in SGD affects generalization. Wen et al. (2019) study the expected dropout objective for matrix factorization and linearized neural nets, respectively. Recent work (Arora et al., 2020) studies dropout applied to only the last layer of a deep neural net, computing an exact expression for the explicit regularizer which depends on the magnitudes of coordinates in the last hidden layer. Our analysis of the explicit regularization results in a more general expression which contains similar quantities, but considers dropout at all layers of the network. These prior works focus on explicit regularization and do not study the implicit regularization effects of dropout.

Prior works have also studied data-dependent regularizers of the model and loss stability. Sokolić et al. (2017); Hoffman et al. (2019) apply Jacobian-based regularization to train robust classifiers. Krueger & Memisevic (2015) propose a data-dependent regularizer for the stability of RNN activations. Novak et al. (2018); Arora et al. (2018); Nagarajan & Kolter (2019); Wei & Ma (2019ab) study the relationship between model stability and generalization.

Finally, regularization for deep models is an important issue in NLP. Zaremba et al. (2014) demonstrated that dropout can be very helpful for NLP tasks. Semeniuta et al. (2016); Gal & Ghahramani (2016b) propose variants of dropout designed for recurrent neural networks. Krueger & Memisevic (2015); Merity et al. (2017b) study temporal activation stability regularization. Merity et al. (2017b); Meis et al. (2017); Merity et al. (2018) demonstrate that the proper tuning of regularizers can greatly impact performance.

On the broader topic of generalization theory of neural networks, Zhang et al. (2016); Neyshabur et al. (2018) observe that deep learning defies a lot of conventional statistical wisdom. Several works have studied generalization bounds for deep networks (see (Bartlett et al., 2017; Neyshabur et al., 2017; Golowich et al., 2017; Dziugaite & Roy, 2017; Arora et al., 2018; Wei & Ma, 2019)) and references therein). Another line of work studies implicit regularization in deep learning (see (Gunasekar et al., 2017, 2018a,b; Soudry et al., 2018; Woodworth et al., 2019; Arora et al., 2019) and references therein).
7 Conclusion

In this work, we show that dropout actually introduces two entangled sources of regularization: an explicit one which modifies the expected objective, and an implicit one due to stochasticity in the updates. We empirically disentangle these regularizers and derive analytic simplifications which faithfully distill each regularization effect. We demonstrate that our simplified regularizers can replace dropout in practice. Our derivations show that dropout regularizes the stability of the model and loss around the training data.

More broadly, our analytic characterizations of dropout can provide intuition on what works and what doesn’t for stability-based regularizers in deep learning. We hope that these intuitions can help inform and motivate the design of more principled regularizers for deep networks.

Acknowledgements

We would like to thank Michael Xie for suggesting the experiment which disentangled the implicit and explicit regularization effects. Sham Kakade would also like to thank Xinyi Chen, Cyril Zhang, and Yi Zhang for numerous helpful discussions and help with earlier experimentation. Colin Wei acknowledges support from an NSF Graduate Research Fellowship. The work is also partially supported by SDSI and SAIL at Stanford. Sham Kakade acknowledges funding from the Washington Research Foundation for Innovation in Data-intensive Discovery, and the NSF Awards CCF-1703574, and CCF-1740551.

References


A Full Derivations in Section 4

A.1 Full Derivation of Equation (4.3)

We decompose the second derivative of the loss with respect to $h_i$, $D^2_{hi} (\ell \circ F_i)[hi(x)]$, into a positive-semidefinite component and non-PSD component:

$$D^2_{hi} (\ell \circ F_i)[hi(x)] = \frac{1}{q} \langle J_{Fi}(x)^\top H_{out} J_{Fi}(x), \text{diag}(h_i(x)^{\otimes 2}) \rangle + \frac{1}{2(q-1)} \langle M, \text{diag}(h_i(x)^{\otimes 2}) \rangle$$

(A.1)

where

$$M \doteq \sum_{j \in [c]} (D_F(\ell(F(x))))_j D^2_{hi}(F_i)_j[h_i(x)]$$

is some matrix capturing second derivatives of the model output. We used $(_j)$ to index the $j$-th coordinate in the output, to avoid confusion with indexing the layers. The first term in (A.1) is positive-semidefinite when $\ell$ is convex, but $M$ is unlikely to be positive-semidefinite since it involves the Hessian of a non-convex model. Plugging everything back into (4.2), we obtain

$$R_{\text{drop}}(F, x) \doteq \frac{q}{2(q-1)} \langle J_{Fi}(x)^\top H_{out} J_{Fi}(x), \text{diag}(h_i(x)^{\otimes 2}) \rangle + \frac{q}{2(q-1)} \langle M, \text{diag}(h_i(x)^{\otimes 2}) \rangle$$

(A.2)

(A.3)

To ensure that our regularizer is nonnegative, we ignore the second term containing the non-PSD matrix $M$. Ignoring the non-PSD term in this kind of decomposition was also suggested in Sagun et al. (2017). We also omit the factor of $\frac{q}{2(q-1)}$ in (4.3) for simplicity.

A.2 Justification of Taylor Expansion

As dropout introduces a change that has magnitude which is multiplicative in the size of the coordinates of the hidden layers, the perturbation due to dropout might not be small. Since Taylor expansions typically require a small level of perturbation, in this section we argue that when the application of dropout is followed by a linear transformation layer, the perturbation to the linear layer could be small. Furthermore, we demonstrate that performing Taylor expansion with respect to this layer will ultimately give the same regularizer.

We work in the same setting of the derivation in Section 4.1. We add the additional assumption that $h_i$ is followed by a linear transformation parameterized by weight matrix $Z$. Thus, we can express $F(x) = F_i(h_i(x)) = F'_{i+1}(Z h_i(x))$ where $F'_{i+1}$ denotes all the computation after the matrix multiplication $Z$. ($F'_{i+1}$ differs from $F_{i+1}$ just by an additional activation layer that follows the matrix multiplication by $Z$.) Now we can compute the loss after applying dropout on $h_i$ by

$$\ell(F(x, \eta)) = \ell(F'_{i+1}(Z h_i(x) + Z \delta))$$

(A.4)

Our key observation, as detailed below, is that although the perturbation $\delta$ to $h_i(x)$ could be large relative to the magnitudes of the coordinates of $h_i(x)$, the perturbation $Z \delta$ may be much smaller relative to the magnitudes of the coordinates of $Z h_i(x)$. Thus, the effect of the dropout noise can be mitigated as it passes through linear layers of the network, making the Taylor expansion more realistic.

Concretely, consider the standard deviation of the $j$-th coordinate of $Z \delta$:

$$\text{std}[(Z \delta)_j] = \sqrt{\frac{\sum_k Z^2_{jk}(h_i(x))_k}{k}}$$

(A.5)

On the other hand, we have

$$(Z h_i(x))_j = \sum_k Z_{jk} \cdot (h_i(x))_k$$

(A.6)

17
In the case where $\{Z_{jk}\}_k$ and $h_i$ share the same sign on each coordinate, the signal $\langle Z h_i(x) \rangle_j \text{ (A.6)}$ can be larger than the size of the perturbation, that is, std[$(Z \delta)_j$] \text{ (A.5)}, by a factor of $\sqrt{d}$. For example, consider the case where all entries of $Z$ and $h_i$ are 1. In other words, even though the vector $\delta$ seems to be comparable to $h_i$ in the norm, after passing through the linear transformation, due to the cancellation arising from the randomness in $\delta$, $Z\delta$ can be much smaller compared to $Z h_i$.

Thus, when the weight matrix and hidden layer are well-aligned, the level of perturbation caused by dropout to the subsequent linear layer might not be too large. This supports our use of Taylor expansion. Now we can also check that Taylor expanding around $Z \delta = 0$ gives the same regularizers. (This is unsurprising because the form of Taylor expansion is invariant to linear transformation.) Using $h_{i+1}'$ to refer to $Z h_i$, we have

$$
\ell(F(x,\eta)) \approx \ell(F(x)) + Dh_{i+1}'(\ell \circ F_{i+1})[Z h_i(x)]Z \delta + \frac{\delta^T Z^T D h_{i+1}'^2 (\ell \circ F_{i+1})[Z h_i(x)]Z \delta}{2}
$$

Now we observe that

$$
D h_{i+1}'(\ell \circ F_{i+1})[Z h_i(x)]Z = Dh_i(\ell \circ F_1)[h_i(x)]
$$

and

$$
Z^T D h_{i+1}'^2 (\ell \circ F_{i+1})[Z h_i(x)]Z = D h_i(\ell \circ F_1)[h_i(x)]
$$

Substituting these back into (A.7) brings us back to (4.1), which served as the starting point for the derivations of our explicit and implicit regularizers. Thus, we obtain the same analytic expressions by performing Taylor expansion around $Z \delta$.

To provide further justification, we evaluate the quality of the Taylor expansion for an LSTM trained with hidden layer dropout with probability 0.5 on Penn Treebank, and found that the quadratic term accounted for > 80% of the difference between the losses with and without dropout. In more details, we found that $E_x[R_{\text{drop}}(x) - R_{\text{approx}}(x)] = 0.009$, whereas $E_x[R_{\text{drop}}(x)] = 0.053$ for this particular model, where the expectation was taken over the training set. Thus, the Taylor approximation is sufficiently tight for (4.4) to capture the important explicit regularization effects of dropout, as is supported in Section 5.

B Proof of Theorem 4.1

In this section, we will analyze a general loss function $\ell : \mathbb{R}^c \times [c] \to \mathbb{R}$ (note that we will frequently hide the dependence on the label $y$, as it is not important for our proofs). As before, for some fixed bound $B > 0$, define $\ell = \min\{B, \ell\}$ to be the truncation of the loss. We carry over the remainder of the notation from the setting in Section 4.3.

Our proof of Theorem 4.1 will rely on the following slightly more general statement for loss functions with an exponential tail.

**Theorem B.1.** Suppose $\ell(\cdot,y)$ is convex and satisfies

$$
\|D(\text{tr} \circ D^2 \ell(\cdot,y))[h]\|_2 \leq \tau \text{tr}(D^2 \ell(\cdot,y)[h])
$$

for all $h$, $y$, and some $\tau > 0$. With probability $1 - \delta$ over the draw of the training examples, for all $W \in \mathbb{R}^{c \times d}$ satisfying the norm bound $\|W^T\|_{2,1} \leq A$ the following holds:

$$
E_\mathcal{D}[\bar{\ell}(Wx)] - 1.01E_{h\sim p}[\bar{\ell}(Wx,y)] \leq \frac{(A \mu(W))^{2/3}(\theta B)^{1/3}}{n^{1/3}} + \frac{A \sqrt{\eta}(W)B}{\sqrt{n}} + \frac{BA^2 \theta \tau^2}{n \left(\log^2 \frac{BA^2 \theta \tau^2}{\eta \mu(W)} + 1\right)} + \frac{B(\log 1/\delta + \log \log n)}{n}
$$

(B.3)
where $\theta = \max_i \|x_i\|^2 \log^3(nc)$ and $\mu, \nu$ measure the Jacobians and Hessians of the loss and are defined by

$$\mu(W) \triangleq \frac{\sum_{i=1}^n \|D\ell(\cdot, y)[Wx_i]\|_2}{n} \quad \text{(B.4)}$$

$$\nu(W) \triangleq \frac{\sum_{i=1}^n \text{tr}(D^2\ell(\cdot, y)[Wx_i])}{n} \quad \text{(B.5)}$$

Given Theorem $\text{[B.1]}$, we can complete the proof of Theorem 4.1 by observing that $\ell_{ce}$ satisfies (B.1), as formally stated in the following lemma.

**Lemma B.1.** Let $\ell_{ce}$ denote the cross entropy loss. Then for any $h, y, \ell_{ce}$ satisfies (B.1) with $\tau = \sqrt{2}$. We provide the full proof of Lemma B.1 in Section B.3.

**Proof of Theorem 4.1.** Using Lemma B.1 to observe that $\ell_{ce}$ satisfies (B.1) with $\tau = \sqrt{2}$, we can plug this value of $\tau$ into Theorem B.1 to get the desired result. Thus, it suffices to prove Theorem B.1. To do so, we will rely on the following lemmas.

**Lemma B.2.** In the setting of Theorem B.1, define $\kappa_{\max} \triangleq \max_i \|x_i\|_2$. With probability $1 - \delta$ over the draw of the sample $\{(x_i, y_i)\}_{i=1}^n$, for all $W \in \mathbb{R}^{c \times d}, \|W\|_2 \leq A$ and $\alpha > 0$, the following holds:

$$E_P[\ell(W, x, y)] \leq (1 + 1/\alpha)E_P[\ell(W, x, y)] + \rho' \left( \min_{\beta > 0} G_{W, \alpha}(\beta) + (1 + \alpha) \frac{B(\log 1/\delta + \log \log n)}{n} \right) \quad \text{(B.6)}$$

where $G_{W, \alpha}$ is the data-dependent function defined by

$$G_{W, \alpha}(\beta) \triangleq \beta(1 + 1/\alpha)\mu(W)\kappa + (1 + 1/\alpha)\nu(W)\exp(\tau\beta\kappa) - \beta\kappa - 1 \frac{\tau^2}{\beta^2 n} + (1 + \alpha)B \frac{A^2 \log^3(nc)}{\beta^2 n} \quad \text{(B.7)}$$

where $\mu(W), \nu(W)$ are defined as in (B.4) and (B.5) which (implicitly) depend on the training data, and $\rho' > 0$ is some universal constant.

We prove this lemma in Section B.1.

**Lemma B.3.** In the setting of Lemma B.2, let $G_{W, \alpha}(\beta)$ be defined as in (B.7). Define $\theta = \log^3(nc)\kappa^2$. Then

$$\min_{\beta > 0} G_{W, \alpha}(\beta) \leq (1 + 1/\alpha)^{2/3} (1 + \alpha)^{1/3} (\theta B)^{1/3} (A\mu(W))^{2/3} + \frac{A \sqrt{\nu(W)(1 + \alpha)(1 + 1/\alpha)\theta B}}{\sqrt{n}} + \frac{BA^2 \theta^2}{n \left( \log^2 \left( \frac{(1 + \alpha)BA^2 \theta^4}{n(1 + 1/\alpha)\nu(W)} \right) + 1 \right)} \quad \text{(B.8)}$$

where $\mu, \nu$ are defined in (B.4) and (B.5).

We prove this Lemma in Section B.2. We now can prove Theorem B.1 by combining the lemmas above.

**Proof of Theorem B.1.** Combining Lemmas B.2 and Lemma B.3 and choosing $\alpha = 100$, we get the desired result. \qed
B.1 Proof of Lemma B.2

In this section, we derive the proof of Lemma B.2. Our proof bounds the generalization of a perturbed loss function. By trading off between perturbation level and generalization error, we obtain Lemma B.2. Define the following perturbed version of the loss \( \tilde{\ell}_\sigma \):

\[
\tilde{\ell}_\sigma(W, x, y) = \max_{\delta \in \mathbb{R}^d} s_\sigma(\|\delta\|_2) \hat{\ell}(W x + \delta x, y)
\]  

where

\[
s_\sigma(t) = \begin{cases} 
(1 - t/\sigma)^2 & \text{for } t < \sigma \\
0 & \text{for } t \geq \sigma 
\end{cases}
\]

This is reminiscent of the all-layer margin technique of [Wei & Ma, 2019b], except we analyze a continuous upper bound for the 0-1 loss. We provide the following generalization bound for \( \tilde{\ell}_\sigma \).

**Lemma B.4.** With probability \(1 - \delta\) over the draw of the training sample, for all \( W \in \mathbb{R}^{c \times d} \) with \( \|W^T\|_{2,1} \leq A, \) and all \( \alpha > 0 \), we have

\[
\mathbb{E}P[\tilde{\ell}_\sigma(W, x, y)] \leq (1 + 1/\alpha)\mathbb{E}P_n[\tilde{\ell}_\sigma(W, x, y)] + \rho(1 + \alpha)B \left( \frac{A^2 \log^3(nc)}{\sigma^2 n} + \frac{\log(1/\delta + \log \log n)}{n} \right)
\]  

for some universal constant \( \rho > 0 \).

We prove the above lemma in Section B.1.1. Our proof technique for obtaining Lemma B.2 will be as follows: we first note that \( \tilde{\ell}_\sigma \) is an upperbound on \( \hat{\ell} \). Second, we will upper bound \( \tilde{\ell}_\sigma \) in terms of \( \ell \) and the derivatives of \( \ell \) on the training data. Combining these two bounds and optimizing over \( \sigma \) will roughly give Lemma B.2. We have the following lemma bounding \( \tilde{\ell}_\sigma \).

**Lemma B.5.** In the setting of Lemma B.2, suppose \( \ell(\cdot, y) \) satisfies (B.1) for all \( h, y \). Then we have the upper bound

\[
\tilde{\ell}_\sigma(W, x, y) \leq \ell(W x, y) + \|D\ell(\cdot, y)[W x]\|_2 |x|_2 \sigma + \text{tr}(D^2\ell(\cdot, y)[W x]) \exp(\sigma \|x\|_2) - \sigma |x|_2 - 1
\]

We prove this lemma in Section B.1.2.

**Proof of Lemma B.2.** Our starting point is Lemma B.4. Our strategy will be to bound \( \tilde{\ell}_\sigma \) in terms of the original loss \( \ell \), and pick the best possible choice of \( \sigma \) for this bound. Define \( \beta_j = \sqrt{\exp(j)} \) and let \( S = \{\beta_j : j = 0, \ldots, J\} \) where we set \( J = \lfloor \log(\sqrt{n}) \rfloor \). Our strategy will be to apply Lemma B.4 for all choices of \( \sigma \) in \( S \) and show that there is some choice of \( \sigma \in S \) giving (B.6).

We apply Lemma B.4 for \( \sigma = \beta_0, \ldots, \beta_J \) using probability \( \delta/|S| \) and union bound over the failure probability. This allows us to conclude that with probability \( 1 - \delta \), for all \( \|W^T\|_{2,1} \leq A \) and \( \beta \in S \),

\[
\mathbb{E}P[\tilde{\ell}_\beta(W, x, y)] \leq (1 + 1/\alpha)\mathbb{E}P_n[\tilde{\ell}_\beta(W, x, y)] + \rho(1 + \alpha)B \left( \frac{A^2 \log^3(nc)}{\beta^2 n} + \frac{\log |S|/\delta + \log \log n}{n} \right)
\]

\[
\leq (1 + 1/\alpha)\mathbb{E}P_n[\tilde{\ell}_\beta(W, \cdot)] + \rho(1 + \alpha)B \frac{A^2 \log^3(nc)}{\beta^2 n} + \rho'(1 + \alpha)B \frac{\log 1/\delta + \log \log n}{n}
\]

In the last line, \( \rho' \) is a universal constant. We used the fact that \( |S| \leq \log n \). Now using the fact that \( \tilde{\ell}_\sigma \) upper bounds \( \ell \) and applying Lemma B.2 we obtain from (B.15) for all \( \|W^T\|_{2,1} \leq A, \) \( j = 0, \ldots, J \):

\[
\mathbb{E}P[\ell(W x, y)] \leq (1 + 1/\alpha) \frac{1}{n} \sum_{i=1}^n \ell(W x, y) + \rho_1 G_{W, \alpha}(\beta_j) + \rho_2(1 + \alpha)B \frac{\log 1/\delta + \log \log n}{n}
\]
where \(\rho_1, \rho_2\) are universal constants and \(G_{W,\alpha}(\beta)\) is defined in \(\text{(B.7)}\). (Note that \(G_{W,\alpha}(\beta)\) depends on \(W\) and the training data.) For a fixed choice of \(W\), let \(\beta^* \doteq \arg \min_{\beta \geq 0} G_{W,\alpha}(\beta)\). First, if \(\beta^* \in [\beta_0, \beta_J]\), then by construction \(\exists \beta \in S\) such that \(\beta \in [\beta^*/\epsilon, \beta^*]\) where \(\epsilon\) denotes the mathematical constant. For this choice of \(j\), we have \(G_{W,\alpha}(\beta) \leq \epsilon^2 G_{W,\alpha}(\beta^*)\) (as the third term in \(G_{W,\alpha}(\beta)\) is the only decreasing term in \(\beta\), and it differs by a factor of at most \(\epsilon^2\) from \(\beta^*\)).

Now consider the case when \(\beta^* < \beta_0 = \frac{\sqrt{\pi}A^{3/2}(nc)}{\sqrt{n}}\). In this case, \(G_{W,\alpha}(\beta^*) > B\), and so we trivially have \(\text{(B.6)}\) since \(\ell\) is upper bounded by \(B\).

Finally, in the case when \(\beta^* > \beta_J\), we note that \(G_{W,\alpha}(\beta_J) - G_{W,\alpha}(\beta^*) \leq \frac{B}{n}\). This is again because only the third term in \(G_{W,\alpha}(\beta)\) is decreasing in \(\beta\), and for \(\beta > \beta_J\), this term is at most \((1 + \alpha)B/n\).

Thus, for all choices of \(\beta^*\), we can conclude that

\[
\mathbb{E}_P[\ell(W, y)] \leq (1 + 1/\alpha)\mathbb{E}_{\rho_1}[\ell(W, y)] + \rho_3 G_{W,\alpha}(\beta^*) + \rho_4 (1 + \alpha)B \frac{\log 1/\delta + \log \log n}{n} \tag{B.17}
\]

for universal constants \(\rho_3, \rho_4 > 0\). This gives the desired result.

It suffices to prove Lemmas \(\text{B.4}\) and \(\text{B.5}\).

### B.1.1 Proof of Lemma \(\text{B.4}\)

To prove Lemma \(\text{B.4}\), we must define the empirical Rademacher complexity of a class of functions. For \(F\) a class of functions taking values in \(\mathbb{R}\), the empirical Rademacher complexity is defined by

\[
\hat{\mathcal{R}}(F) = \mathbb{E}_z[\sup_{f \in F} z_i f(x_i, y_i)] \tag{B.18}
\]

where \((x_i, y_i)\) are datapoints in the training sample and \(z_i\) are drawn i.i.d. and uniformly from \([-1, +1]\). Furthermore, for some set \(S\) (i.e., some function class), let \(\mathcal{N}_S(\epsilon, \mathcal{S})\) be the covering number of \(S\) in the metric induced by the norm \(\| \cdot \|\) with error \(\epsilon\). We will use the notation \(L_2(P_n)\) to denote the following norm defined via the training sample: \(\|f\|_{L_2(P_n)} = (\mathbb{E}_{x \sim P_n}[f(x)^2])^{1/2}\). \(\mathcal{N}_2(L_2(P_n))\) will then denote the covering number in norm \(\| \cdot \|_{L_2(P_n)}\). We will use the proof technique of \(\text{[Spebro et al., 2010]}\).

We will require the following bound on how \(\ell_\sigma\) changes when the weight matrix \(W\) changes.

**Claim B.1.** For any \(W, W'\) we have \((\tilde{\ell}_\sigma(W, x, y) - \tilde{\ell}_\sigma(W', x, y))^2 \leq (\tilde{\ell}_\sigma(W, x, y) + \tilde{\ell}_\sigma(W', x, y))^4 B\|W - W'\|_2^2 / (\epsilon^2 \sigma^4)\).

Define \(\mathbf{\hat{L}}_\sigma(r) = \{\tilde{\ell}_\sigma(W, \cdot) : W \in \mathbb{R}^{c \times d}, \|W\|_{2,1} \leq A, \mathbb{E}_{P_n}[\tilde{\ell}_\sigma(W, x, y)] \leq r\}\) to be the data-dependent\(^7\) class of loss functions with average empirical loss at most \(r\). Next, we will require a certain covering number bound for \(\{W x_i\}_{i=1}^n\):

**Claim B.2.** In the above setting, define the set \(\mathcal{W}(r) \doteq \{W \in \mathbb{R}^{c \times d} : \|W\|_{2,1} \leq A, \mathbb{E}_{P_n}[\tilde{\ell}_\sigma(W, x, y)] \leq r\}\). For any choice of \(\epsilon > 0\), there exists a set of matrices \(\hat{\mathcal{W}} \subset \mathcal{W}(r)\) with cardinality bounded by

\[
\log |\hat{\mathcal{W}}| \leq 1152 \frac{4A^2}{\epsilon^2} \log_2(2|16A/\epsilon + 2n| + 1) + \log c
\]

satisfying the following: for all \(W \in \mathcal{W}(r)\), there exists \(\hat{W} \in \hat{\mathcal{W}}\) such that

\[
\frac{\|W x_i - \hat{W} x_i\|_2}{\|x_i\|_2} \leq \epsilon \forall i = 1, \ldots, n \tag{B.19}
\]

Applying Claims \(\text{B.1}\) and \(\text{B.2}\) lets us bound the covering number of \(\hat{\mathcal{L}}_\sigma(r)\).

\(^7\)Note that \(\hat{\mathcal{L}}_\sigma(r)\) is data-dependent because the loss on the training data is required to be bounded by \(r\).
Claim B.3. In the above setting, we have the covering number bound
\[
\log N_{L_2(\mathcal{P}_n)} \left( \frac{\sqrt{8Br}}{\sigma}, \tilde{\ell}_\sigma(r) \right) \leq 1152 \left[ \frac{4A^2}{\epsilon^2} \right] (\log_2 (216A/\epsilon + 2)n + 1) + \log c \tag{B.20}
\]

Proof of Claim B.3 Let \( \tilde{W}_\epsilon \) be the set of matrices inducing the \( \epsilon \) cover of \( \{W x_i/\|x_i\|_2\}_{i=1}^n \) whose cardinality is bounded in Claim B.2. For any \( \tilde{W} \in \tilde{W}_\epsilon \), we can compute
\[
\|	ilde{\ell}_\sigma(W, \cdot, \cdot) - \tilde{\ell}_\sigma(\tilde{W}, \cdot, \cdot)\|_{L_2(\mathcal{P}_n)} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\tilde{\ell}_\sigma(W, x_i, y_i) - \tilde{\ell}_\sigma(\tilde{W}, x_i, y_i))^2}
\leq \sqrt{\frac{1}{n} \sum_{i=1}^n (\tilde{\ell}_\sigma(W, x_i, y_i) + \tilde{\ell}_\sigma(\tilde{W}, x_i, y_i)) \frac{4B\|W x_i - W' x_i\|_2^2}{\|x_i\|_2^2}} \tag{by Claim B.1}
\leq \frac{2}{\sigma} \sqrt{\frac{B}{n} \sum_{i=1}^n (\tilde{\ell}_\sigma(W, x_i, y_i) + \tilde{\ell}_\sigma(\tilde{W}, x_i, y_i)) \max_i \frac{\|W x_i - W' x_i\|_2}{\|x_i\|_2}} \tag{B.22}
\leq \frac{\sqrt{8Br}}{\sigma} \epsilon \tag{B.23}
\]

Thus, using \( \{\tilde{\ell}_\sigma(\tilde{W}, \cdot, \cdot) : \tilde{W} \in \tilde{W}_\epsilon\} \) lets us conclude (B.20).

This translates into the following Rademacher complexity bound for \( \hat{\ell}_\sigma(r) \):

Claim B.4. In the setting of Claim B.3, we have
\[
\hat{\mathcal{R}}(\hat{\ell}_\sigma(r)) \leq \frac{3500A\sqrt{Br \log^4(35nc)}}{\sigma \sqrt{n}} \tag{B.24}
\]

Proof of Claim B.4 We apply Dudley’s entropy integral using the covering number bound in Claim B.3. This mirrors the calculation used to prove Lemma 2.2 in (Srebro et al., 2010). From Lemma A.1 of (Srebro et al., 2010), we have
\[
\hat{\mathcal{R}}(\hat{\ell}_\sigma(r)) \leq \inf_{\alpha > 0} \left( 4\alpha + 10 \int_{\alpha}^{\sqrt{Br}} \sqrt{\frac{\log N_{L_2(\mathcal{P}_n)}(\epsilon, \hat{\ell}_\sigma(r))}{n}} d\epsilon \right) \tag{B.25}
\]

Now we perform a change of variables \( \epsilon = \frac{\sqrt{8Br}}{\sigma} \epsilon' \), after which (B.25) becomes
\[
\hat{\mathcal{R}}(\hat{\ell}_\sigma(r)) \leq \inf_{\alpha > 0} \left( 4\alpha + \frac{10}{\sqrt{8Br}} \int_{\frac{\sqrt{8Br}}{\sigma} \alpha}^{\sqrt{8Br}/\sigma} \sqrt{\frac{\log N_{L_2(\mathcal{P}_n)}(\frac{\sqrt{8Br}}{\sigma} \epsilon', \hat{\ell}_\sigma(r))}{n}} d\epsilon' \right) \tag{B.26}
\leq \inf_{\alpha > 0} \left( 4\alpha + 680 \frac{A}{\sigma} \frac{\sqrt{8Br}}{\sigma} \int_{\frac{\sqrt{8Br}}{\sigma} \alpha}^{\min\{2A, \sigma/\sqrt{8}\}} \frac{\log_2(216A/\epsilon' + 2)n + 1}{{\epsilon'}^{\sqrt{n}}} d\epsilon' \right) \tag{B.27}
\]

To change the upper limit of the integral from \( \sigma/\sqrt{8} \) to \( \min\{2A, \sigma/\sqrt{8}\} \), we used the fact that we only need to integrate \( \epsilon' \) to \( 2A \), because for \( \epsilon' > 2A \) the log covering number is 0 by Claim B.3. Now we plug in \( \alpha = \frac{A\sqrt{8Br}}{\sigma \sqrt{n}} \).
We construct a perturbation \( r \) which is essentially local Rademacher complexity (Bousquet, 2002). Thus, rearranging and using the fact that
\[
\tilde{\ell}(\hat{\sigma}(r)) \leq \frac{A\sqrt{128B\sigma}}{\sqrt{n}} + 680 \frac{A\sqrt{8B\sigma}}{\sigma} \int_{\frac{A}{4\sigma}}^{2A} \frac{\sqrt{3\log 35nc}}{\epsilon'\sqrt{n}} d\epsilon'
\] (B.28)
\[
\leq \frac{A\sqrt{128B\sigma}}{\sqrt{n}} + 680 \frac{A\sqrt{24Br\log(35nc)}}{\sigma\sqrt{n}} \log(2\sqrt{n})
\] (B.29)
\[
\leq \frac{3500A\sqrt{Br\log^3(35nc)}}{\sigma\sqrt{n}}
\] (B.30)

Using Claim \[\text{B.4}\] we can complete the proof of Lemma \[\text{B.4}\] using the technique of \[\text{Srebro et al., 2010}\], which is essentially local Rademacher complexity \[\text{Bousquet, 2002}\].

**Proof of Lemma \[\text{B.4}\]** Define \( \psi(r) \doteq \frac{3500A\sqrt{Br\log^3(35nc)}}{\sigma\sqrt{n}} \). By Claim \[\text{B.4}\] we have \( \hat{\sigma}(r) \leq \psi(r) \).

Thus, we can apply the steps from the proof of Theorem 1 in \[\text{Srebro et al., 2010}\] (which invokes Theorem 6.1 of \[\text{Bousquet, 2002}\]), we have with probability \( 1 - \delta \), for all \( W \in \mathbb{R}^{c \times d}, \|W\|_{2,1} \leq A \)
\[
\mathbb{E}_p[\tilde{\ell}(W, x, y)] \leq \mathbb{E}_p[\tilde{\ell}(W, x, y)] + 106r^* + \frac{48B}{n} (\log 1/\delta + \log \log n)
\]
\[
+ \mathbb{E}_p[\tilde{\ell}(W, x, y)](8r^* + \frac{4B}{n}(\log 1/\delta + \log \log n))
\]

By the AM-GM inequality, for all \( \alpha > 0 \), we have
\[
\mathbb{E}_p[\tilde{\ell}(W, x, y)] \leq (1 + 1/2\alpha)\mathbb{E}_p[\tilde{\ell}(W, x, y)] + r^*(106 + 4\alpha) + \frac{(48 + 2\alpha)B}{n}(\log 1/\delta + \log \log n)
\] (B.31)

where \( r^* \) satisfies \( \psi(r^*) = r^* \). Now using the fact that \( r^* \leq \frac{BA^2\log^3(nc)}{\sigma n} \), we obtain \( \text{(B.12)} \) for some \( \rho > 0 \). \( \square \)

**Proof of Claim \[\text{B.7}\]** Let \( \delta^* \) be the optimal perturbation for \( W \) and \( x, y \), i.e.
\[
\delta^* \doteq \arg \max_{\delta \in \mathbb{R}} s_\sigma(\|\delta\|_2)\tilde{\ell}(Wx + \delta\|x\|_2, y)
\]

We construct a perturbation \( \delta' \) for the objective of \( \tilde{\ell}(W', x, y) \) as follows: define \( \delta' \doteq \delta^* + \frac{Wx - W'x}{\|x\|_2} \). It follows that
\[
\tilde{\ell}(W', x, y) \geq s_\sigma(\|\delta'\|_2)\tilde{\ell}(W'x + \delta'\|x\|_2, y)
\]
\[
= s_\sigma(\|\delta'\|_2)\tilde{\ell}(Wx + \delta\|x\|_2, y)
\]
\[
\geq s_\sigma\left(\|\delta\|_2 + \frac{\|Wx - W'x\|_2}{\|x\|_2}\right)\tilde{\ell}(Wx + \delta\|x\|_2, y)
\]
\[
\geq s_\sigma(\|\delta\|_2) - \frac{2\|Wx - W'x\|_2}{\|x\|_2}\sqrt{s_\sigma(\|\delta\|_2)}\tilde{\ell}(Wx + \delta\|x\|_2, y)
\] (by construction of \( \delta' \))
\[
\geq \left(s_\sigma(\|\delta\|_2) - \sqrt{s_\sigma(\|\delta\|_2)}\right)\tilde{\ell}(Wx + \delta\|x\|_2, y)
\]

Thus, rearranging and using the fact that \( \tilde{\ell}(W, x, y) = s_\sigma(\|\delta\|_2)\tilde{\ell}(Wx + \delta\|x\|_2, y) \), we obtain
\[
\tilde{\ell}(W', x, y) - \tilde{\ell}(W, x, y) \geq -\frac{2\|Wx - W'x\|_2}{\|x\|_2}\sqrt{s_\sigma(\|\delta\|_2)}\tilde{\ell}(Wx + \delta\|x\|_2, y)
\]
\[
\geq -\sqrt{B}\frac{2\|Wx - W'x\|_2}{\|x\|_2}\sqrt{\tilde{\ell}(W, x, y)}
\]
(\text{using the upper bound } (\tilde{\ell}(Wx + \delta\|x\|_2, y))^{1/2} \leq \sqrt{B})

23
Using the same reasoning, we can also obtain
\[
\tilde{\ell}_\sigma(W, x, y) - \tilde{\ell}_\sigma(W', x, y) \geq -\sqrt{B^2 \|W x - W' x\|_2} \sqrt{\tilde{\ell}_\sigma(W', x, y)} \tag{B.34}
\]

It thus follows that
\[
|\tilde{\ell}_\sigma(W', x, y) - \tilde{\ell}_\sigma(W, x, y)| \leq \sqrt{B^2 \|W x - W' x\|_2} \max\{\sqrt{\tilde{\ell}_\sigma(W, x, y)}, \sqrt{\tilde{\ell}_\sigma(W', x, y)}\} \tag{B.35}
\]

Squaring both sides gives the desired result.

**Proof of Claim B.2.** We first construct a set of matrices \(\tilde{W}_c \subset \mathbb{R}^{c \times d}\) satisfying for all \(W \in \mathbb{R}^{c \times d}\) with \(\|W^\top\|_{2,1} \leq A\), there exists \(\tilde{W} \in \tilde{W}_c\) with
\[
\frac{\|W x_i - \tilde{W} x_i\|_2}{\|x_i\|_2} \leq \epsilon \quad i = 1, \ldots, n \tag{B.36}
\]

We first note that when \(\epsilon \geq A\), this set only needs cardinality 1, as we simply take \(\tilde{W}\) to only have the all 0's matrix. First, consider a \(\epsilon/2\)-cover in \(\|\cdot\|_2\)-norm of the set \(\{v \in \mathbb{R}^c : \|v\|_1 \leq A\}\), which we denote by \(\tilde{V}\). By classical results, such a cover exists with \(\log\) cardinality \(4A^2/\epsilon^2\log(\epsilon + 1)\).

Next, by Theorem 4 of Zhang (2002), for all choices of \(\epsilon', A > 0\), there exists a set \(\tilde{U}(\epsilon', a) \subset \{u \in \mathbb{R}^d : \|u\|_2 \leq a\}\) such that for any \(u \in \mathbb{R}^d\), \(\|u\|_2 \leq a\), there exists \(\tilde{u} \in \tilde{U}(\epsilon', a)\) such that \(\frac{\|u - \tilde{u}\|_2}{\|x_i\|_2} \leq \epsilon' \quad \forall i = 1, \ldots, n\). Furthermore, the cardinality of this set satisfies the bound
\[
\tilde{U}(\epsilon', a) \leq 144A^2/\epsilon^2 \log_2(2[A/\epsilon'] + 2n + 1) \tag{B.37}
\]

Now for any \(v \in \tilde{V}\), we add to our cover the set \(\tilde{W}_c(v) \doteq \{\tilde{W} \in \mathbb{R}^{c \times d} : \tilde{W} \in \tilde{U}(\epsilon\sqrt{\|v\|_1/4\|v\|_1}, |v_j|}\}. We have
\[
\log |\tilde{W}_c(v)| \leq \sum_{j=1}^c \log |\tilde{U}(\epsilon\sqrt{|v_j|/4\|v\|_1}, |v_j|)| \tag{B.38}
\]
\[
\leq \sum_{j=1}^c 576 \frac{\|v\|_1|v_j|}{\epsilon^2} \log_2(2[8\sqrt{\|v\|_1|v_j|}/\epsilon + 2]n + 1) \tag{B.39}
\]
\[
\leq 576 \frac{\|v\|_1^2}{\epsilon^2} \log_2(2[8\|v\|_1/\epsilon + 2]n + 1) \tag{B.40}
\]
\[
\leq \frac{576}{\epsilon^2} A^2 \log_2(2[8A/\epsilon + 2]n + 1) \tag{B.41}
\]

Furthermore, setting \(\tilde{W}_c = \cup_{v \in \tilde{V}} \tilde{W}_c(v)\), we thus have
\[
\log |\tilde{W}_c| \leq \log |\tilde{V}| + 576 \frac{A^2}{\epsilon^2} \log_2(2[8A/\epsilon + 2]n + 1) \tag{B.42}
\]
\[
\leq 1152 \frac{A^2}{\epsilon^2} \log_2(2[8A/\epsilon + 2]n + 1) + \log c \tag{B.43}
\]

To obtain the last line, we use the fact that \(\tilde{W}_c\) has cardinality 0 when \(\epsilon \geq A\). It remains to show that \(\tilde{W}\) satisfies the desired error properties. For any \(W\) satisfying \(\|W^\top\|_{2,1} \leq A\), there exists \(\bar{v} \in \tilde{V}\) satisfying
\[
\|\bar{v} - (\|\tilde{W}\|_2)_{j=1}^c\|_2^2 \leq \epsilon/2 \tag{B.44}
\]
Furthermore, by construction there exists $\hat{W} \in \hat{W}_x(\bar{v})$ satisfying
\[
\frac{\|I_j^T \bar{v}_j - \|I_j^T \hat{W}x_i\|_2}{\|x_i\|_2} \leq \frac{\epsilon}{2} \sqrt{\frac{|\bar{v}_j|}{\|x_i\|_2}} \forall j = 1, \ldots, c, \ i = 1, \ldots, n
\] (B.45)

It follows that for all $i = 1, \ldots, n$, we have
\[
\frac{\|Wx_i - \hat{W}x_i\|_2}{\|x_i\|_2} \leq \sqrt{\sum_j (I_j^T (\hat{W}x_i - Wx_i))^2} \leq \frac{\epsilon}{2} \sqrt{\sum_j (\hat{W}x_i)^2} + \sqrt{\sum_j (\hat{W}x_i - Wx_i)^2}
\] (by triangle inequality)
\[
\leq \frac{\epsilon}{2} + \sqrt{\sum_j (\bar{v}_j - \|I_j^T W\|_2)^2} \leq \frac{\epsilon}{2} \sqrt{\sum_j (\hat{W}x_i)^2} + \sqrt{\sum_j (\bar{v}_j - \|I_j^T W\|_2)^2}
\] (applying (B.45))
\[
\leq \epsilon/2 + \sqrt{\sum_j (\bar{v}_j - \|I_j^T W\|_2)^2} \leq \epsilon\] (by (B.44))

To conclude the statement of the lemma, we note that for each element $\hat{W} \in \hat{W}_{c/2}$, we can add to $\hat{W}_x$ a single $\hat{W}' \in W(r)$ satisfying
\[
\frac{\|\hat{W}x_i - \hat{W}'x_i\|_2}{\|x_i\|_2} \leq \epsilon/2 \forall i = 1, \ldots, n
\] (B.47)

Then $\hat{W}_x$ will be the desired cover with cardinality bounded by
\[
\log |\hat{W}_x| \leq \log |\hat{W}_{c/2}| \leq 1152 \left(\frac{4A^2}{\epsilon^2}\right) (\log_2 (2[16A/\epsilon + 2]n + 1) + \log \epsilon)
\] (B.48)

### B.1.2 Proof of Lemma [B.5]

We will rely on the following bound on the change of a function satisfying [B.1].

**Claim B.5.** Suppose the loss function $\ell : \mathbb{R}^c \rightarrow \mathbb{R}$ is convex and satisfies $\|D(\text{tr} \circ D^2\ell)(h)\|_2 \leq \tau \text{tr}(D^2\ell(h))$ for all $h$ and some $\tau > 0$. Then for all $h, h'$, we have
\[
\ell(h') \leq \ell(h) + \|D\ell(h)\|_2 \|h' - h\|_2 + \text{tr}(D^2\ell(h)) \exp(\tau \|h' - h\|_2) - \tau \|h' - h\|_2 - 1
\] (B.49)

**Proof.** The proof of this claim mirrors the proof of Proposition 1 in [Bach et al. 2010]. □

Now we complete the proof of Lemma B.5.
Proof of Lemma [B.3] We can calculate
\[
\tilde{\ell}(W, x, y) = \max_{\delta \in \mathbb{R}} s_\sigma(\|\delta\|_2) \tilde{\ell}(W x + \delta \|x\|_2, y)
\]
which gives the desired statement from the definitions of \(G\).

To see this, let \(G\). Applying this to \(G\), we observe that to minimize any function of the form \(a_1(\exp(\beta) - \beta - 1) + a_2/\beta^2\), we can set \(\beta = (a_2)^{-1}a_1^{1/3}\alpha^{1/3}\). Applying this to \(G\) gives
\[
\min_{\beta} G(\beta) \leq \min_{\beta}(G_1(\beta) + G_3(\beta)) + \min_{\beta}(G_2(\beta) + G_3(\beta))
\]
which gives the desired statement from the definitions of \(G\).

We first claim that
\[
\min_{\beta > 0} G(\beta) < \min_{\beta > 0}(G_1(\beta) + G_3(\beta)) + \min_{\beta}(G_2(\beta) + G_3(\beta))
\]
To see this, let \(\beta^*, \beta_1, \beta_2\) be the minimizers for \(G, G_1 + G_3, G_2 + G_3\), respectively, and assume without loss of generality that \(\beta_2 \geq \beta_1\). Then we have
\[
G(\beta^*) \leq G(\beta_1) = G_1(\beta_1) + G_2(\beta_1) + G_3(\beta_1)
\]
(\(G_2\) is an increasing function)
\[
< G_1(\beta_1) + G_2(\beta_2) + G_3(\beta_1) + G_3(\beta_2) + G_3(\beta_2)
\]
which gives the desired statement from the definitions of \(\beta_1, \beta_2\). Thus, it suffices to minimize \(G_1 + G_3, G_2 + G_3\) separately.

To bound the minimum of \(G_1 + G_3\), we observe that to minimize any function of the form \(a_1 \beta + a_2/\beta^2\), we can set \(\beta = (a_1)^{-1/3}a_2^{1/3}\). Applying this to \(G_1 + G_3\) gives
\[
\min_{\beta} G_1(\beta) + G_3(\beta) \leq \frac{(1 + 1/\alpha)^{2/3}(1 + \alpha)^{1/3} B^{1/3} \log(nc)}{\eta^{1/3}} (A_\mu \kappa)^{2/3}
\]
Next, we bound the minimum of $G_2 + G_3$. As the function $\exp(a) - a - 1$ is increasing in $a$, we have

\[ G_2(\beta) \leq (1 + 1/\alpha)\nu \frac{\exp(\tau \beta \kappa) - \tau \beta \kappa - 1}{\tau^2} \]  

(B.60)

where $\nu$ is defined in (B.5). Let $\tilde{G}_2(\beta)$ denote the right-hand side of the above equation. Now we can invoke Claim [B.6] on the variable $\tau \beta \kappa$ to conclude that

\[ \min_{\beta > 0} G_2(\beta) + G_3(\beta) < \min_{\beta > 0} \tilde{G}_2 + G_3(\beta) \leq \frac{A \kappa \sqrt{(1 + 1/\alpha)B \log^3(nc)}}{\sqrt{n}} \frac{BA^2 \log^3(nc) \kappa^2 \tau^2}{(\log^2 (\frac{(1+\alpha)BA^2 \log^3(nc) \kappa^2 \tau^2}{\alpha(1+1/\alpha)\nu}) + 1)} \]

(B.61)

Finally, invoking (B.56) and applying the definition of $\theta$ gives the desired result. □

Proof of Claim [B.6] First consider the case when $a_1 > a_2$. In this case, set $\beta' = \sqrt{a_2/a_1} < 1$. As $\exp(b) - b - 1 \leq b^2$ for $0 \leq b < 1$, we have in this case $G(\beta') \leq \sqrt{a_1 a_2}$.

Otherwise, consider the case when $a_2 > a_1$ but $\frac{a_2}{a_1} \leq \log^2 \frac{a_2}{a_1}, \log^2 \frac{a_2}{a_1} < 1$, or $\log^2 \frac{a_2}{a_1} \leq 2 \log \log \frac{a_2}{a_1} + 2$. If any of these three equations hold, then $a_2/a_1$ is upper bounded by some universal constant, in which case setting $\beta' = 1$ immediately gives $G(\beta') \leq \sqrt{a_1 a_2}$.

Otherwise, consider the case when $\frac{a_2}{a_1} > \log^2 \frac{a_2}{a_1}, \log^2 \frac{a_2}{a_1} > 1$, and $\log^2 \frac{a_2}{a_1} > 2 \log \log \frac{a_2}{a_1} + 2$ all hold. In this case, we set $\beta' = \log \left( \frac{a_2}{a_1} \right)$. Note that $\beta' > 0$ and $\beta' = \log \frac{a_2}{a_1} - \log \log^2 \frac{a_2}{a_1} > \frac{1}{2} \log \frac{a_2}{a_1} + 1$ by our conditions on $a_2/a_1$. Then we have

\[ G(\beta') \leq a_1 \exp(\beta) + a_2/\beta^2 \leq \frac{a_2}{\log^2 \frac{a_2}{a_1} + 1} \left( \log \frac{a_2}{a_1} + 1 \right)^2 \]

(B.63)

(B.64)

(B.65)

Combining the three cases gives the desired claim. □

B.3 Additional Proofs of Helper Lemmas

We first provide the proof of Lemma [B.1]. We use $p$ to denote the vector of probabilities predicted by the cross entropy loss, formally defined in Section [E]. We will rely on the derivatives of the cross entropy loss computed in Section [E].

Proof of Lemma [B.1] We first compute $\text{tr}(D^2 e_x^c[h]) = \sum_i p_i - p_i^2$, by Section [E]. Next, we have $D \text{tr}(D^2 e_y^c)[h] = \sum_i (1 - 2p_i)p_i(\bar{1}_i - p)$. Now by the convexity of $| \cdot |_2$, as $\sum_i p_i = 1$, we have

\[ \| \sum_i (1 - 2p_i)p_i(\bar{1}_i - p) \|_2 \leq \sum_i p_i |1 - 2p_i| \| \bar{1}_i - p \|_2 \leq \sum_i p_i \| \bar{1}_i - p \|_2 \]  

(B.66)

Now we have

\[ \| \bar{1}_i - p \|_2 = \sqrt{(1 - p_i)^2 + \sum_{j \neq i} p_j^2} \leq \sqrt{(1 - p_i)^2 + \left( \sum_{j \neq i} p_i \right)^2} \]

(B.67)

(B.68)

(B.69)
Plugging this back into the previous equation, we obtain

\[ \| \sum_i (1 - 2p_i) p_i (\hat{1}_i - p) \|_2 \leq \sum_i p_i \| \hat{1}_i - p \|_2 \leq \sqrt{2} \left( \sum_i p_i - p_i^2 \right) \]  

(B.70)

This gives the desired result.

Next, the following statement is useful for Claim B.1.

Claim B.7. In the setting of Claim B.1 for any \( a, b > 0 \), we have

\[ (s_\sigma(a + b) - s_\sigma(a))^2 \leq 4 \frac{1}{\sigma^2} s_\sigma(a) b^2 \]  

(B.71)

Proof. First, in the case where \( a + b \geq \sigma, a \geq \sigma \), we have \( s_\sigma(a + b) = s_\sigma(a) = 0 \) so the inequality trivially holds.

Second, in the case where \( a + b \geq \sigma, a < \sigma \), we have \( s_\sigma(a + b) = 0 \), so the LHS of (B.71) is simply \( s_\sigma(a)^2 \). Now we note that \( b \geq \sigma - a \), so \( b^2/\sigma^2 \geq (1 - a/\sigma)^2 = s_\sigma(a) \). Thus, we have \( s_\sigma(a)^2 \leq s_\sigma(a)b^2/\sigma^2 \), so (B.71) follows.

Third, in the case where \( a + b < \sigma, a < \sigma \), we have

\[ s_\sigma(a) - s_\sigma(a + b) = \left( 2 - \frac{2a + b}{\sigma} \right) \left( \frac{b}{\sigma} \right) \]  

(expanding the expression for \( s_\sigma \))

Squaring both sides, we obtain

\[ (s_\sigma(a) - s_\sigma(a + b))^2 = \left( 2 - \frac{2a + b}{\sigma} \right)^2 \frac{b^2}{\sigma^2} \]  

(B.72)

\[ \leq 4(1 - a/\sigma)^2 \frac{b^2}{\sigma^2} \]  

(B.73)

\[ \leq 4s_\sigma(a) \frac{b^2}{\sigma^2} \]  

(B.74)

\[ \square \]

### B.4 Discussion of Bound

Consider the case where the empirical distribution is only supported on \( c' \ll c \) tightly clustered classes and all the datapoints \( x_i \) have norm 1. Suppose that the norms of the rows of \( W \) are balanced and concentrated on the \( c' \) classes in the empirical sample. Let \( p \) denote the softmax probability vector defined in (B.3). Consider weights \( W \) which are well aligned with the data, so that \( 1 - p(W x_i) y_i \approx \exp(-\|\hat{1}_i y_i W\|_2) \approx \exp(-\|W\|_F/\sqrt{c'}) \), for every training example \((x_i, y_i)\) (this is possible because the softmax probability vector is exponential-tailed).

In this case, by the expression for the Hessian of cross entropy loss (see Section E), we would have \( \log(1/\nu(W)) \approx E_{p_i} [-\log(1 - p(W x)_y)] \geq \frac{|W|_F}{\sqrt{c'}} \). On the other hand, we also have \( \|W\|_{2,1} = \sqrt{c'} \|W\|_F \). Thus, the third term in the bound becomes \( O\left( \frac{(B/c')^2 \log(n/\delta)}{\alpha} \right) \).

### C Additional Implementation Details

We implement our code in PyTorch, basing our LSTM implementation on the following code: [https://github.com/salesforce/awd-lstm-lm](https://github.com/salesforce/awd-lstm-lm). We base our Transformer-XL implementation on the following code: [https://github.com/kimiyoung/transformer-xl](https://github.com/kimiyoung/transformer-xl) Code for downloading and pre-processing the datasets which we use are also contained in these repositories. We run our code on NVIDIA TitanXp GPUs. We provide detailed descriptions of the algorithms we implement below.
Algorithm 3 Unbiased estimate of $R_{\text{approx}}$ for cross-entropy loss.

Input: data $x$.
Sample $\hat{y} \sim \text{softmax}(F(x))$.
Initialize $r = 0$.
for layers $i$ do
  Compute $\hat{J}_i = D\ell_{\text{ce}} \circ F_i[h_i(x)]$.
  Update $r = r + \hat{J}_i \text{diag}(h_i(x)^{\odot 2}) \hat{J}_i^\top$.
end for
Return $r$.

C.1 Implementing Our Explicit Regularizer

Because of the large output dimensionality of language modeling tasks, we cannot compute $R_{\text{approx}}$ exactly and instead approximate it by sampling. In this section, we describe how to implement the sampling procedure for variants of the popular cross entropy loss defined in (E.1). Recall that we use the notation $\ell_{\text{ce}}(v, y) = -\log \text{softmax}(v)_y$, where softmax$(v)$ denotes the softmax distribution computed from $v$.

We leverage the following relationship between the first and second order derivatives of the loss:

$$H_{\text{out}}(x) = \mathbb{E}_{\hat{y} \sim \text{softmax}(F(x))}[D \ell_{\text{ce}}^y[F(x)]^\top D \ell_{\text{ce}}^y[F(x)]]$$  \hspace{1cm} (C.1)

This relationship formally proved in Claim [E.1]. Note in particular that our regularizer $R_{\text{approx}}$ does not depend on the true label $y$ since the loss Hessian $H_{\text{out}}$ is independent of $y$. Algorithm 3 leverages this relationship to compute an unbiased estimator for $R_{\text{approx}}$ based on (C.1) by sampling a label $\hat{y} \sim \text{softmax}(F(x))$ and computing the loss Jacobian for label $\hat{y}$ instead of the full Hessian matrix $H_{\text{out}}$. We formally prove the correctness of Algorithm 3 below.

Claim C.1. Algorithm 3 gives an unbiased estimate of $R_{\text{approx}}$ defined in (4.4).

Proof. Note that

$$\mathbb{E}_{\hat{y} \sim \text{softmax}(F(x))}[\hat{J}_i \text{diag}(h_i(x)^{\odot 2}) \hat{J}_i^\top] = \mathbb{E}_{\hat{y} \sim \text{softmax}(F(x))}[\langle \hat{J}_i^\top \hat{J}_i, \text{diag}(h_i(x)^{\odot 2}) \rangle]$$ \hspace{1cm} (C.2)

$$= \mathbb{E}_{\hat{y} \sim \text{softmax}(F(x))}[\langle J_{F,i}(x)^\top D \ell_{\text{ce}}^y[F(x)]^\top D \ell_{\text{ce}}^y[F(x)] J_{F,i}(x), \text{diag}(h_i(x)^{\odot 2}) \rangle]$$ \hspace{1cm} (C.3)

$$= \langle J_{F,i}(x)^\top H_{\text{out}}(x) J_{F,i}(x), \text{diag}(h_i(x)^{\odot 2}) \rangle \hspace{1cm} \text{(by C.1)}$$

Summing over $i$ gives the desired result.

Algorithm 3 admits a straightforward extension to the adaptive softmax loss [Grave et al., 2017] which computes the derivative for the loss with respect to a sampled cluster label and sampled word within the cluster.

C.2 Implementing Figure 4 Experiment

Algorithm 4 describes more formally how to implement DROPOUT$_k$ with injection of noise $\xi_{\text{approx}}$, which was plotted in Figure 4.

---

When computing the gradient update, we do not differentiate through the sampling probabilities softmax$(F(x))$. Thus, though our loss estimate is unbiased, our estimate of $\nabla_W R_{\text{approx}}(F, x)$ is biased. This does not appear to matter in practice.
C.3 Using Identity Instead of Loss Hessian

It is non-trivial to implement this experiment described in Section 5.2, as the dimensionality of the output is large and naively computing the regularizer \( \langle J^T_{F,i} J_{F,i}, \text{diag}(h_i^{(2)}) \rangle \) requires computing the output Jacobian \( J_{F,i} \) exactly. To circumvent this issue, we use sampling. Letting \( \eta_i \) be a random vector whose coordinates are independently and uniformly sampled from \([-1, 1] \) with coordinates independently and uniformly distributed in \([-1, 1] \), we have

\[
\mathbb{E}[(\eta_i \odot h_i)^T J^T_{F,i} J_{F,i} \eta_i \odot h_i)] = \langle J^T_{F,i} J_{F,i}, \text{diag}(h_i^{(2)}) \rangle \tag{C.4}
\]

Now to compute the value \( J_{F,i} \eta_i \odot h_i \) we use the method for computing Jacobian vector products described here: [https://j-towns.github.io/2017/06/12/A-new-trick.html](https://j-towns.github.io/2017/06/12/A-new-trick.html).

D Additional Experimental Results

D.1 Additional Results for Section 5.1

For all experiments in Section 5.1 we use SGD with learning rate of 30 with gradient clipping with a threshold of 0.35, and default \( \ell_2 \)-regularization of 1.2e-6 (unless we specify that we tuned this parameter). These parameters are the defaults from the awd-lstm-lm repository.\footnote{https://github.com/salesforce/awd-lstm-lm} For Penn Treebank, we use a batch size of 20 training for 150 epochs and for WikiText-2, we use a batch size of 40 training for 100 epochs. We use these same base settings for the experiments in Section 5.2. Combining our explicit and implicit regularizers adds 7-8 times runtime overhead per iteration compared to dropout.

We use a coefficient of \( \lambda_1 = 2/3 \) for our explicit regularizer. For our implicit regularizer, for the experiments corresponding to Figure 4 we provide implementation details in Algorithm 4. For the experiments which combine our explicit and implicit regularizers, we implement \( \xi_{\text{approx}} \) by sampling \( \eta \) with coordinates independently and uniformly distributed from \([-1, 1] \), and computing \( \xi_{\text{approx}}(F, x, \eta) \) with coefficient \( \lambda_2 = \sqrt{2/3} \) in Algorithm 2. This is meant to match the dropout probability of 0.4.

In Tables 3 and 4 we summarize our results across the experiments in Section 5.1.

D.2 Additional Results for Section 5.3

For our Transformer-XL experiments, we use the hyperparameters for the base model on WikiText-103 contained in the following repository: [https://github.com/kimiyoung/transformer-xl/](https://github.com/kimiyoung/transformer-xl/). We use an explicit regularization coefficient of \( \lambda = 0.11 \) to match the dropout probability \( q = 0.1 \). Our explicit regularizer takes 3 times longer per iteration than dropout.

We also examine the implicit regularization effect of dropout on the QRNN architecture for WikiText-103. We use a batch size of 15 with the Adam optimizer and an initial learning rate of 5e-4 for all our runs. We chose these parameters to fit the DROPOT updates in memory. The other hyperparameters are set to their defaults in the awd-lstm repository. We chose to use QRNN as they are faster to train than LSTMs [Bradbury et al., 2016; Merity et al., 2018]. Table 5 demonstrates that the implicit regularization effect does not appear...

---

**Algorithm 4** Procedure for injecting our update noise into Dropout \( k \) updates.

**Input:** minibatch \( \{x_i\}_{i=1}^m \), number of dropout noise samples \( k \), dropout probability \( q \).

1. Sample noise \( \eta_{ij} \) for \( i \in [m], j \in [k] \).
2. Compute \( g = \nabla W \left( \frac{1}{m} \sum_{i=1}^m \hat{\ell}_{\text{drop}, k}(F,F_i,\eta_{ij}) \right) \).
3. Sample noise \( \eta_{mk+1} \) with coordinates independently and uniformly distributed in \([-1, 1] \).
4. Update \( g \) by adding \( \sqrt{\frac{q}{q-1}} \sqrt{1 - \frac{k}{q}} \xi_{\text{approx}}(F, x, \eta_{mk+1}) \).
5. Use \( g \) for optimization algorithm.
Table 3: A summary of the validation perplexities of our regularizers and various baselines on Penn Treebank. Our regularizers match their dropout counterparts.

<table>
<thead>
<tr>
<th>Training Method</th>
<th>Best Val. Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
</tr>
<tr>
<td>DROPOUT&lt;sub&gt;1&lt;/sub&gt;</td>
<td>73.76</td>
</tr>
<tr>
<td>DROPOUT&lt;sub&gt;8&lt;/sub&gt;</td>
<td>83.82</td>
</tr>
<tr>
<td>DROPOUT&lt;sub&gt;32&lt;/sub&gt;</td>
<td>89.12</td>
</tr>
<tr>
<td>No regularization</td>
<td>122.16</td>
</tr>
<tr>
<td>Best $\ell_2$ reg.</td>
<td>112.04</td>
</tr>
<tr>
<td><strong>Our regularizers</strong></td>
<td></td>
</tr>
<tr>
<td>$R_{\text{approx}}$ [4.4]</td>
<td>84.52</td>
</tr>
<tr>
<td>$R_{\text{approx}}, 8$ samples</td>
<td>84.27</td>
</tr>
<tr>
<td>Algorithm 4, $k = 8$</td>
<td>74.49</td>
</tr>
<tr>
<td>$R_{\text{approx}}$ and $\xi_{\text{approx}}$ [4.6]</td>
<td>72.99</td>
</tr>
</tbody>
</table>

Table 4: A summary of the validation perplexities of our regularizers and various baselines on Wikitext-2. Our regularizers match their dropout counterparts.

<table>
<thead>
<tr>
<th>Training Method</th>
<th>Best Val. Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
</tr>
<tr>
<td>DROPOUT&lt;sub&gt;1&lt;/sub&gt;</td>
<td>90.97</td>
</tr>
<tr>
<td>DROPOUT&lt;sub&gt;8&lt;/sub&gt;</td>
<td>95.91</td>
</tr>
<tr>
<td>DROPOUT&lt;sub&gt;32&lt;/sub&gt;</td>
<td>98.10</td>
</tr>
<tr>
<td>No regularization</td>
<td>144.12</td>
</tr>
<tr>
<td>Best $\ell_2$ reg.</td>
<td>137.50</td>
</tr>
<tr>
<td><strong>Our regularizers</strong></td>
<td></td>
</tr>
<tr>
<td>$R_{\text{approx}}$ [4.4]</td>
<td>92.26</td>
</tr>
<tr>
<td>$R_{\text{approx}}$ (8 samples)</td>
<td>94.73</td>
</tr>
<tr>
<td>Algorithm 4, $k = 8$</td>
<td>90.87</td>
</tr>
<tr>
<td>$R_{\text{approx}}$ and $\xi_{\text{approx}}$ [4.6]</td>
<td>84.57</td>
</tr>
</tbody>
</table>
Table 5: Experimental results on the full WikiText-103 dataset for QRNN architecture.

<table>
<thead>
<tr>
<th>Training Method</th>
<th>Best Val. Ppl.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DROPOUT_1</td>
<td>34.24</td>
</tr>
<tr>
<td>DROPOUT_2</td>
<td>33.35</td>
</tr>
<tr>
<td>DROPOUT_4</td>
<td>32.74</td>
</tr>
<tr>
<td>DROPOUT_8</td>
<td>32.78</td>
</tr>
</tbody>
</table>

on the WikiText-103 dataset, which also matches our observations for the Transformer-XL architecture on this same dataset. This suggests that the dataset size, and not architecture, influences whether the implicit regularization effect appears.

E Useful Properties of Cross-Entropy Loss

Recall that we defined the cross entropy loss $\ell_y^\text{ce}$ with label $y$ by

$$\ell_y^\text{ce}(Wx) \doteq -\log \frac{\exp((Wx)_y)}{\sum_{y'} \exp((Wx)_{y'})} \quad (E.1)$$

The derivatives of the cross-entropy loss are given as follows:

$$D_v \ell_y^\text{ce}[v] = p(v) - \bar{1}_y \quad (E.2)$$

where $p(v)$ is the softmax probability vector given by

$$(p(v))_{y'} = \frac{\exp(v_{y'})}{\sum_{y''} \exp(v_{y''})} \quad (E.3)$$

Furthermore, it also holds that

$$D^2_v \ell_y^\text{ce}[v] = D^2_v p[v] = \text{diag}(p(v)) - p(v)p(v)^\top \quad (E.4)$$

Furthermore, we have the following relationship between the first and second derivatives of the loss:

Claim E.1. $D^2 \ell_y^\text{ce}[v] = E_{\hat{y} \sim p(v)}[D \ell_{\hat{y}}^\text{ce}[v]^\top D \ell_{\hat{y}}^\text{ce}[v]]$. 

Proof. We have

$$E_{\hat{y} \sim p(v)}[D \ell_{\hat{y}}^\text{ce}[v]^\top D \ell_{\hat{y}}^\text{ce}[v]] = E_{\hat{y} \sim p(v)}[(p(v) - \bar{1}_\hat{y})(p(v) - \bar{1}_\hat{y})^\top] \quad (E.5)$$

Note that the expectation of $\bar{1}_\hat{y}$ for $\hat{y} \sim p(v)$ is simply $p(v)$. Thus, the right hand side simplifies to the desired statement. $\square$