The Implicit and Explicit Regularization Effects of Dropout

Colin Wei [1] **Sham Kakade** [2] **Tengyu Ma** [1]

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 ℓ_2 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 adressed 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 example 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 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.

[1]Stanford University. [2]Microsoft Research & University of Washington. Correspondence to: Colin Wei <colinwei@stanford.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[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.

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:

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

Here F_{drop} denotes the dropout model and 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]Our code is available at https://github.com/cwein3/dropout-analytical.

Algorithm 1 DROPOUT$_k$, mini-batch dropout update using k samples of noise.

Input: Training examples $\{x_i\}_{i=1}^m$.
Sample noise η_{ij} for $i \in [m], j \in [k]$.
Compute $g = \nabla_W \left(\frac{1}{m} \sum_{i=1}^m \widehat{\ell}_{\text{drop},k}(F, x_i, \{\eta_{ij}\}_{j=1}^k) \right)$.
$\triangleright$ Use g for optimization algorithm.

2. Preliminaries

Notation. For a function $f(a) : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$ on a vector a, we use $D_{\boldsymbol{a}}^k f[b] \in \mathbb{R}^{d_2 \times d_1}$ 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 $\boldsymbol{a}$ when the emphasis is unnecessary. Let $f \circ g$ to denote the composition of f with g. For vector v, let $\text{diag}(v)$ denote the matrix with entries of v on its diagonal and 0 elsewhere. Let $\text{tr}(M)$ 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 $v_1^{\odot 2} \triangleq v_1 \odot v_1$. Let $\vec{1}$ 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 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:

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

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

$$h_{\text{drop}} = (\vec{1} + \eta) \odot h$$

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

3. Disentangling Explicit and Implicit Regularization in Dropout

We now present an experimental study designed to disentangle the two regularization effects, which confirms the

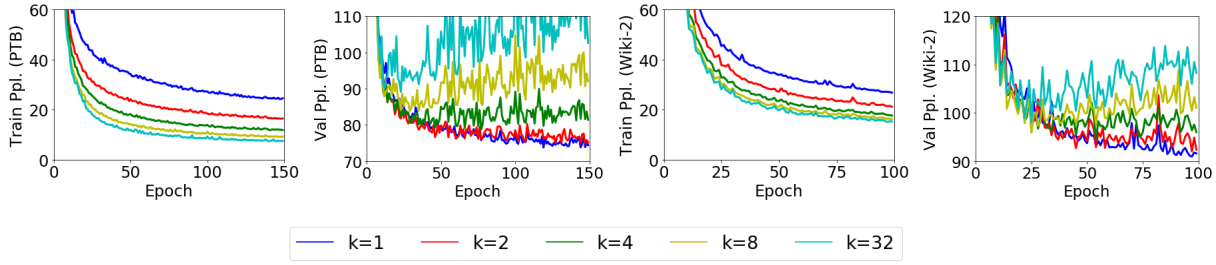

Figure 1. **Averaging dropout noise degrades performance.** Perplexity vs. epoch of LSTMs trained with mini-batch dropout, DROPOUT$_k$ for various k (see Algorithm 1). Training perplexity is evaluated without dropout. Increasing the number of samples of dropout noise, k, reduces the amount of update 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. **Left:** Penn Treebank. **Right:** WikiText-2.

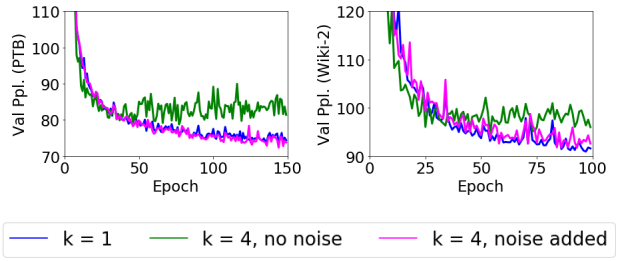

Figure 2. **Confirming implicit regularization effect.** Validation perplexity vs. epoch of LSTMs trained with DROPOUT$_1$, DROPOUT$_4$, and DROPOUT$_4$ with noise added via the procedure in Section 3.1. By adding noise to DROPOUT$_4$, we recover the performance of DROPOUT$_1$. Thus, the noise we add has an implicit regularization effect. **Left:** Penn Treebank. **Right:** WikiText-2.

existence of implicit regularization in dropout. Furthermore, this approach allows us to study each effect in isolation.

Let $L(F) \triangleq \mathbb{E}_x[\ell(F(x))]$ 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 η (defined in Section 2) is

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

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

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

It is natural to define the explicit regularizer as the difference between the expected training objective (averaged over both x and η) 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:

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

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 η 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 γ is the stepsize, x is a randomly sampled datapoint, and η is a randomly sampled dropout noise variable.

We demonstrate that the stochasticity from sampling η 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 $\{\eta_i\}_{i=1}^k$:

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

For training, we now use the stochastic gradient $\nabla_W \widehat{\ell}_{\text{drop},k}$, 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 1.

[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.

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 trained without dropout (for Penn Treebank, the best perplexity without dropout is around 120, which is outside the bounds of the graph). DROPOUT$_k$ and 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 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 DROPOUT$_k$ updates in order to recover the performance of standard dropout. Let ξ_{drop} denote the fluctuation of the stochastic dropout gradient around its mean:

$$\xi_{\text{drop}}(F, x, \eta) \triangleq \nabla_W \ell(F(x, \eta)) - \nabla_W \ell_{\text{drop}}(F, x) \quad (3.3)$$

Note that ξ_{drop} is exactly the gradient noise in standard dropout. Furthermore, we have $\text{Cov}(\nabla_W \widehat{\ell}_{\text{drop},k}) = \frac{1}{k}\text{Cov}(\xi_{\text{drop}})$. To correct the covariance of the DROPOUT$_k$ gradient, we will add mean-zero noise with covariance $\left(1 - \frac{1}{k}\right)\text{Cov}(\xi_{\text{drop}})$. Let η_1 and η_2 be two independent draws of the dropout noise. Define $\widetilde{\xi}_{\text{drop}}$ as:

$$\widetilde{\xi}_{\text{drop}}(F, x, \eta_1, \eta_2) \triangleq \nabla_W \ell(F(x, \eta_1)) - \nabla_W \ell(F(x, \eta_2))$$

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

In Figure 2, we verify that this correction procedure recovers the test performance of DROPOUT$_1$. Thus, we have constructed a (complicated) implicit regularizer which explains the discrepancy between DROPOUT$_k$ and 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 ℓ_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 rewrite the loss after applying dropout on h_i by $\ell(F(x, \eta)) = \ell(F_i(h_i(x) + \delta))$, where $\delta \triangleq \eta_i \odot h_i(x)$ is the perturbation to the i-th layer. We can apply Taylor expansion to analyze the effect of this perturbation.[4] We apply Taylor expansion around $\delta = \vec{0}$:

$$\ell(F(x, \eta)) - \ell(F(x)) \approx D_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i]\delta + \frac{\delta^\top (D^2_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i])\delta}{2} \quad (4.1)$$

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

$$R_{\text{drop}}(F, x) = \mathbb{E}_\eta[\ell(F(x, \eta))] - \ell(F(x)) \approx \frac{1}{2}\mathbb{E}_\delta[\delta^\top (D^2_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i(x)])\delta]$$

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 δ:

$$\mathbb{E}_\delta[\delta^\top D^2_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i]\delta] = \left\langle D^2_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i], \frac{\mathbb{E}[\delta\delta^\top]}{2} \right\rangle = \frac{q}{2(q-1)} \left\langle D^2_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i], \text{diag}(h_i(x)^{\odot 2}) \right\rangle \quad (4.2)$$

[4] Taylor 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.

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

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

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

$$R^i_{\text{approx}}(F, x) \triangleq \left\langle J_{F,i}(x)^\top H_{\text{out}}(x) J_{F,i}(x), \text{diag}(h_i(x)^{\odot 2}) \right\rangle \tag{4.3}$$

Readers may find this reminiscent of the decomposition of the Hessian of neural nets loss (LeCun et al., 2012; Sagun et al., 2017). Indeed, we decompose $D^2_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i]$ 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 (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 δ'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:

$$R_{\text{approx}}(F, x) \triangleq \sum_i \left\langle J_{F,i}(x)^\top H_{\text{out}}(x) J_{F,i}(x), \text{diag}(h_i(x)^{\odot 2}) \right\rangle \tag{4.4}$$

Interpretation. Our regularizer ensures that the Jacobians and hidden layers of the model output are small when measured in the norm of H_{out}. We note that for cross entropy loss, $H_{\text{out}}(x) = \text{diag}(p) - pp^\top \succcurlyeq 0$, where 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* (see (4.2) in our derivation). These quantities are closely related. For example, consider weight matrix Z parametrizing some linear transformation layer, such that $F(x) = F'(Zh(x))$, where h, F' denote the compositions of the layers before and after the application of Z. Then defining $h'(x) = Zh(x)$, we have

$$D_{\boldsymbol{Z}}(\ell \circ F)(x)[Z] = h(x) D_{\boldsymbol{h'}}(\ell \circ F')[Zh(x)]$$

Thus, the loss derivatives with respect to model parameters can be expressed in terms of those with respect to the hidden layers.

We emphasize that one benefit of $R_{\text{approx}}(F, x)$ is that it provides an interpretable and detailed characterization of the explicit regularization effect of dropout. We hope this can help provide theoreticians and practictioners alike with precise intuitions on why dropout works, and, more broadly, how to design effective stability regularizers in practice.

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, ξ_{drop}.

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

We apply Taylor expansion to approximate $\widetilde{\xi}_{\text{drop}}$, 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[5]

$$\begin{aligned}
&\widetilde{\xi}_{\text{drop}}(F, x, \eta_i^{(1)}, \eta_i^{(2)}) \\
&= \nabla_W \left(\ell(F_i(h_i + \delta^{(1)})) - \ell(F_i(h_i + \delta^{(2)})) \right) \\
&\approx \nabla_W \left(J_{\text{loss},i}(x)(\eta_i^{(1)} - \eta_i^{(2)}) \odot h_i(x) \right)
\end{aligned}$$

where $J_{\text{loss},i}(x)$ denotes the Jacobian of the loss with respect to the hidden layers: $J_{\text{loss},i}(x) \triangleq D_{\boldsymbol{h_i}}(\ell \circ F_i)[h_i(x)]$. 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 ξ_{drop}, we obtain the following

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

analytic form for update noise:

$$\xi^i_{\text{approx}}(F, x, \eta_i) \triangleq \nabla_W \left(J_{\text{loss},i}(x)(\eta_i \odot h_i(x)) \right) \quad (4.5)$$

Multi-layer Dropout. To handle multi-layer dropout, we Taylor expand over all the layers, obtaining a sum of (4.5) over the layers:[6]

$$\xi_{\text{approx}}(F, x, \{\eta_i\}) \triangleq \nabla_W \left(\sum_i J_{\text{loss},i}(x)(\eta_i \odot h_i(x)) \right) \quad (4.6)$$

To replace the implicit effect of dropout, we add the mean-zero noise ξ_{approx} to the gradients of the objective.

Interpretation: It is a major open question in deep learning theory to understand the regularization effects of noise (Li et al., 2019). For example, it is even unclear why mini-batch noise in SGD empirically helps in general. Prior works (Yaida, 2018; Wei & Schwab, 2019) have (heuristically) suggested that the noise encourages the algorithm to find a solution that minimizes the trace of the covariance of the noise. As the covariance of ξ_{approx} is some function of $\{J_{\text{loss},i}\}, \{h_i\}$, and their gradients with respect to W, the induced regularizer controls some data-dependent stability of the model. Note the conceptual difference with the explicit regularizer, which multiplies the model Jacobian with the loss Hessian, whereas ξ_{approx} multiplies the model Jacobian with the loss Jacobian. 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, 2019a;b) 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.

[6] To make tuning slightly simpler, we compute the noise by sampling the coordinates of η_i uniformly from $\{-1, +1\}$ and scaling by $\sqrt{\frac{q}{q-1}}$, as this preserves the covariance.

Let ℓ^{ce}_y 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

$$\ell^{\text{ce}}_y(Wx) \triangleq -\log \frac{\exp((Wx)_y)}{\sum_{y'} \exp((Wx)_{y'})}$$

Let $\overline{\ell^{\text{ce}}} = \min\{B, \ell^{\text{ce}}\}$ 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: $\|M\|_{2,1} \triangleq \sum_j \sqrt{\sum_i (M^2_{ij})}$. 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 $\|W^\top\|_{2,1} \leqslant A$, the following holds:*

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

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

$$\mu(W) \triangleq \mathbb{E}_{P_n}[\|D\ell^{\text{ce}}_y[Wx]\|_2]$$
$$\nu(W) \triangleq \mathbb{E}_{P_n}[\text{tr}(D^2\ell^{\text{ce}}_y[Wx])] \quad (4.7)$$

Additionally, we define $\theta \triangleq \log^3(nc)\max_i \|x_i\|_2^2$ and $\zeta \triangleq \frac{B(\log(1/\delta)+\log\log n)}{n}$ is a low order term.

We provide the proof in Section B. 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 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 $O\left(\frac{B\log^3(nc)}{n}\right)$ (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 ℓ_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.

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 ξ, regularization strengths λ_1, λ_2.
Compute $g = \frac{1}{m}\sum_{i=1}^m \nabla_W(\ell(F(x_i)) + \lambda_1 R(F, x_i))$.
Update $g = g + \frac{1}{m}\sum_{i=1}^m \lambda_2 \xi(F, x_i)$.
$\triangleright$ Use g for optimization algorithm.

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 matricess. 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_{approx} only, adding our noise ξ_{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).

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 ℓ_{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 our update noise derived in (4.6) can effectively replicate the implicit regularization effect of dropout. We inject appropriately scaled ξ_{approx} noise into the DROPOUT_k training procedure. As the covariance of $\nabla_W \widehat{\ell}_{\text{drop},k}$ scales with $\frac{1}{k}$, we scale ξ_{approx} by a factor $\sqrt{1-\frac{1}{k}}$. Thus, if ξ_{approx} and ξ_{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 ξ_{approx} captures essential properties of ξ_{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

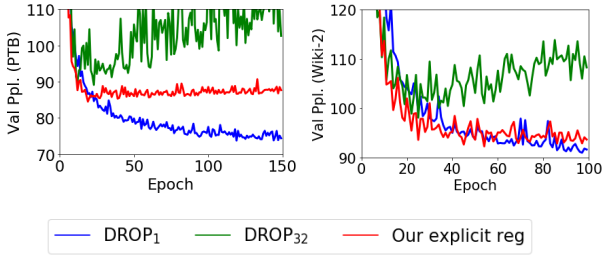

Figure 3. **Our explicit regularizer v.s. dropout.** Validation perplexity vs. epoch of LSTMs trained with DROPOUT_1, DROPOUT_{32} (see Algorithm 1), and our explicit regularizer only. Our explicit regularizer (4.4) outperforms DROPOUT_{32} but does not match DROPOUT_1 since it is missing the implicit regularization benefit of dropout. **Left:** Penn Treebank. **Right:** WikiText-2.

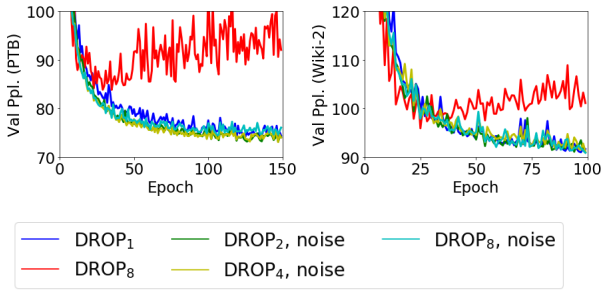

Figure 4. **Our implicit regularizer v.s. dropout.** Validation perplexity vs. epoch of LSTMs trained using mini-batch dropout, DROPOUT_k, with injection of noise ξ_{approx} (Algorithm 4). For reference, we also plot DROPOUT_1 and DROPOUT_8 with no noise (Algorithm 1). DROPOUT_k with noise injection matches DROPOUT_1 for $k = 2, 4, 8$, affirming that our noise distribution ξ_{approx} captures the implicit regularization effect of dropout noise. **Left:** Penn Treebank. **Right:** WikiText-2.

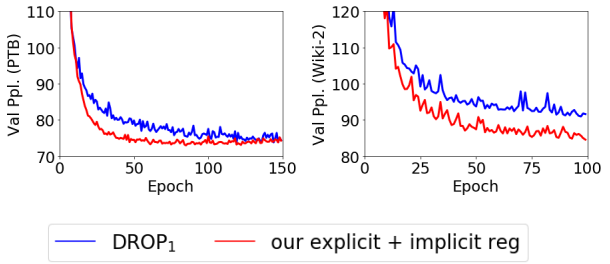

Figure 5. **Our combined regularizer v.s. dropout.** Validation perplexity vs. epoch of LSTMs trained with our regularizers vs standard dropout. Our regularizers can match dropout and appear to improve the validation perplexity faster. **Left:** Penn Treebank. **Right:** WikiText-2.

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.

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_{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 ℓ_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: $H_{\text{out}}(x) \approx D_{\boldsymbol{F}}\ell^{\text{ce}}[F(x)]^\top D_{\boldsymbol{F}}\ell^{\text{ce}}[F(x)]$ (see Section C.1). Substituting this approximation into our explicit regularizer gives the following alternative regularizer:

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

On Penn Treebank, we find that this regularizer is much more effective than ℓ_2 regularization but cannot match dropout. We test whether $\widetilde{R}_{\text{approx}}$ performs well as R_{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. $\widetilde{R}_{\text{approx}}$ on its own significantly outperforms ℓ_2 regularization and can match R_{approx} after tuning. However, with update noise it does not match dropout or R_{approx} (even after tuning).

Table 1. Regularization effect of $\widetilde{R}_{\text{approx}}$ (see (5.1)) on Penn Treebank with and without implicit regularization. $\widetilde{R}_{\text{approx}}$ can significantly outperform ℓ_2 regularization but does not match dropout even with implicit regularization, whereas R_{approx} can.

	Training Method	Best Val. Ppl.
No implicit	ℓ_2 reg (tuned)	112.04
	$\widetilde{R}_{\text{approx}}$ (tuned)	84.06
	R_{approx} (4.4)	84.52
With implicit	DROPOUT$_1$	73.76
	$\widetilde{R}_{\text{approx}}$ (tuned) and ξ_{approx}	79.54
	R_{approx} and ξ_{approx} (4.5)	72.99

Table 2. Experimental results on WikiText-103 dataset for Transformer architecture. The implicit regularization effect of dropout noise appears to decrease with dataset size.

Dataset Size	Training Method	Best Val. Ppl.
Full Dataset	No regularization	29.45
	DROPOUT$_1$	23.39
	DROPOUT$_2$	23.13
	DROPOUT$_4$	23.14
	R_{approx}	24.12
$0.2\times$ Dataset	DROPOUT$_1$	46.05
	DROPOUT$_2$	46.07
	DROPOUT$_4$	47.40

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.

6. Related Work

Dropout has been the focus of several theoretical works studying its properties for both optimization and generalization (Wager et al., 2013; 2014; 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). Jastrzębski et al. (2017); Zhu et al. (2018); Wen et al. (2019) study how the noise distribution in SGD affects generalization. Wen et al. (2019) inject noise with an appropriate covariance structure into the updates of large-batch SGD, making it match the behavior of small-batch SGD. We inject appropriate noise to make large-sample dropout updates match standard 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 (2019a;b) 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); Melis 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, 2019b) 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

Arora, R., Bartlett, P. L., Mianjy, P., and Srebro, N. Dropout: Explicit forms and capacity control, 2020. URL https://openreview.net/forum?id=Bylthp4Yvr.

Arora, S., Ge, R., Neyshabur, B., and Zhang, Y. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 7411–7422, 2019.

Bach, F. et al. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4:384–414, 2010.

Baldi, P. and Sadowski, P. J. Understanding dropout. In *Advances in neural information processing systems*, pp. 2814–2822, 2013.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pp. 6240–6249, 2017.

Bousquet, O. Concentration inequalities and empirical processes theory applied to the analysis of learning algorithms. 2002.

Bradbury, J., Merity, S., Xiong, C., and Socher, R. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.

Cavazza, J., Morerio, P., Haeffele, B., Lane, C., Murino, V., and Vidal, R. Dropout as a low-rank regularizer for matrix factorization. *arXiv preprint arXiv:1710.05092*, 2017.

Chaudhari, P. and Soatto, S. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10. IEEE, 2018.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016a.

Gal, Y. and Ghahramani, Z. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pp. 1019–1027, 2016b.

Gao, H., Pei, J., and Huang, H. Demystifying dropout. In *The 36th International Conference on Machine Learning (ICML 2019)*, 2019.

Golowich, N., Rakhlin, A., and Shamir, O. Size-independent sample complexity of neural networks. *arXiv preprint arXiv:1712.06541*, 2017.

Grave, E., Joulin, A., Cissé, M., Jégou, H., et al. Efficient softmax approximation for gpus. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1302–1310. JMLR. org, 2017.

Gunasekar, S., Woodworth, B. E., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pp. 6151–6159, 2017.

Gunasekar, S., Lee, J. D., Soudry, D., and Srebro, N. Implicit bias of gradient descent on linear convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 9461–9471, 2018a.

Gunasekar, S., Lee, t., Soudry, D., and Srebro, N. Characterizing implicit bias in terms of optimization geometry. *arXiv preprint arXiv:1802.08246*, 2018b.

Helmbold, D. P. and Long, P. M. On the inductive bias of dropout. *The Journal of Machine Learning Research*, 16 (1):3403–3454, 2015.

Helmbold, D. P. and Long, P. M. Surprising properties of dropout in deep networks. *The Journal of Machine Learning Research*, 18(1):7284–7311, 2017.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Hoffman, J., Roberts, D. A., and Yaida, S. Robust learning with jacobian regularization. *arXiv preprint arXiv:1908.02729*, 2019.

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Jastrzebski, S., Kenton, Z., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*, 2018.

Kakade, S. M., Sridharan, K., and Tewari, A. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. In *Advances in neural information processing systems*, pp. 793–800, 2009.

Keskar, N. S. and Socher, R. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Krueger, D. and Memisevic, R. Regularizing rnns by stabilizing activations. *arXiv preprint arXiv:1511.08400*, 2015.

LeCun, Y. A., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

Li, Y., Ma, T., and Zhang, H. Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations. *arXiv preprint arXiv:1712.09203*, 2017.

Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, pp. 11669–11680, 2019.

Ma, X., Gao, Y., Hu, Z., Yu, Y., Deng, Y., and Hovy, E. Dropout with expectation-linear regularization. *arXiv preprint arXiv:1609.08017*, 2016.

Maeda, S.-i. A bayesian encourages dropout. *arXiv preprint arXiv:1412.7003*, 2014.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pp. 114–119. Association for Computational Linguistics, 1994.

Melis, G., Dyer, C., and Blunsom, P. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017a.

Merity, S., McCann, B., and Socher, R. Revisiting activation regularization for language rnns. *arXiv preprint arXiv:1708.01009*, 2017b.

Merity, S., Keskar, N. S., and Socher, R. An Analysis of Neural Language Modeling at Multiple Scales. *arXiv preprint arXiv:1803.08240*, 2018.

Mianjy, P. and Arora, R. On dropout and nuclear norm regularization. *arXiv preprint arXiv:1905.11887*, 2019.

Mianjy, P., Arora, R., and Vidal, R. On the implicit bias of dropout. *arXiv preprint arXiv:1806.09777*, 2018.

Nagarajan, V. and Kolter, J. Z. Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience. *arXiv preprint arXiv:1905.13344*, 2019.

Neyshabur, B., Bhojanapalli, S., and Srebro, N. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. Towards understanding the role of over-parametrization in generalization of neural networks. *arXiv preprint arXiv:1805.12076*, 2018.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. *arXiv preprint arXiv:1802.08760*, 2018.

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

Semeniuta, S., Severyn, A., and Barth, E. Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*, 2016.

Smith, S. L. and Le, Q. V. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Sokolić, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Srebro, N., Sridharan, K., and Tewari, A. Smoothness, low noise and fast rates. In *Advances in neural information processing systems*, pp. 2199–2207, 2010.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Wager, S., Wang, S., and Liang, P. S. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, pp. 351–359, 2013.

Wager, S., Fithian, W., Wang, S., and Liang, P. S. Altitude training: Strong bounds for single-layer dropout. In *Advances in Neural Information Processing Systems*, pp. 100–108, 2014.

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pp. 1058–1066, 2013.

Wang, S. and Manning, C. Fast dropout training. In *international conference on machine learning*, pp. 118–126, 2013.

Wei, C. and Ma, T. Data-dependent sample complexity of deep neural networks via lipschitz augmentation. In *Advances in Neural Information Processing Systems*, pp. 9722–9733, 2019a.

Wei, C. and Ma, T. Improved sample complexities for deep networks and robust classification via an all-layer margin. *arXiv preprint arXiv:1910.04284*, 2019b.

Wei, M. and Schwab, D. J. How noise affects the hessian spectrum in overparameterized neural networks. *ArXiv*, abs/1910.00195, 2019.

Wen, Y., Luk, K., Gazeau, M., Zhang, G., Chan, H., and Ba, J. Interplay between optimization and generalization of stochastic gradient descent with covariance noise. *arXiv preprint arXiv:1902.08234*, 2019.

Woodworth, B., Gunasekar, S., Lee, J., Soudry, D., and Srebro, N. Kernel and deep regimes in overparametrized models. *arXiv preprint arXiv:1906.05827*, 2019.

Xing, C., Arpit, D., Tsirigotis, C., and Bengio, Y. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

Yaida, S. Fluctuation-dissipation relations for stochastic gradient descent. *arXiv preprint arXiv:1810.00004*, 2018.

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, pp. 4949–4959, 2018.

Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, T. Covering number bounds of certain regularized linear function classes. *Journal of Machine Learning Research*, 2(Mar):527–550, 2002.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. The anisotropic noise in stochastic gradient descent: Its behavior of escaping from minima and regularization effects. *arXiv preprint arXiv:1803.00195*, 2018.