Implicit Regularization of Dropout

Zhongwang Zhang and Zhi-Qin John Xu

***Abstract*—It is important to understand how dropout, a popular regularization method, aids in achieving a good generalization solution during neural network training. In this work, we present a theoretical derivation of an implicit regularization of dropout, which is validated by a series of experiments. Additionally, we numerically study two implications of the implicit regularization, which intuitively rationalizes why dropout helps generalization. First, we find that input weights of hidden neurons tend to condense on isolated orientations trained with dropout. Condensation is a feature in the non-linear learning process, which makes the network less complex. Second, we find that the training with dropout leads to the neural network with a flatter minimum compared with standard gradient descent training, and the implicit regularization is the key to finding flat solutions. Although our theory mainly focuses on dropout used in the last hidden layer, our experiments apply to general dropout in training neural networks. This work points out a distinct characteristic of dropout compared with stochastic gradient descent and serves as an important basis for fully understanding dropout.**

***Index Terms*—Neural networks, dropout, condensation, flatness, implicit regularization.**

I. Introduction

DROPOUT is used with gradient-descent-based algorithms for training neural networks (NNs) [1], [2], which can improve the generalization in deep learning [3], [4]. For example, common neural network frameworks such as PyTorch default to utilizing dropout during transformer training. Dropout works by multiplying the output of each neuron by a random variable with probability p being $1/p$ and $1-p$ being zero during training. Note that every time the concerning quantity is calculated, the variable is randomly sampled at each feedforward operation.

Manuscript received 10 April 2023; revised 24 December 2023; accepted 13 January 2024. Date of publication 23 January 2024; date of current version 7 May 2024. This work was supported in part by the National Key R&D Program of China under Grant 2022YFA1008200, in part by the National Natural Science Foundation of China under Grants 92270001 and 12371511, in part by the Shanghai Municipal of Science and Technology Major Project under Grant 2021SHZDZX0102, and in part by the HPC of School of Mathematical Sciences and the Student Innovation Center, and the Siyuan-1 cluster supported in part by the Center for High Performance Computing at Shanghai Jiao Tong University. Recommended for acceptance by R. Giryes. *(Corresponding author: Zhi-Qin John Xu.)*

Zhongwang Zhang is with the Institute of Natural Sciences, School of Mathematical Sciences, MOE-LSC and Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: 0123zzw666@sjtu.edu.cn).

Zhi-Qin John Xu is with the Institute of Natural Sciences, School of Mathematical Sciences, MOE-LSC and Qing Yuan Research Institute, Shanghai Jiao Tong University, Shanghai 200240, China, and also with Shanghai Seres Information Technology Company, Ltd, Shanghai 200040, China (e-mail: xuzhiqin@sjtu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2024.3357172, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2024.3357172

A series of works have studied the explicit and implicit regularizations of dropout from various perspectives, such as complexity analysis [5], [6] and low-rank bias for matrix completion [7], etc. Detailed discussions can be found in Section II. In this work, we aim to study the implicit regularization of dropout by analyzing the properties of neural network weights, instead of connecting dropout with generalization directly. The approach used in this work, consisting of phenomenon-driven analysis and theory-driven analysis, reveals more details of dropout and provides more perspectives for future study.

The effect of dropout is equivalent to adding a specific noise to the gradient descent training. Theoretically, based on the method of the modified gradient flow [8], we derive implicit regularization terms of the dropout training for networks with dropout on the last hidden layer. The implicit regularization of dropout can lead to two important implications, condensed weights and flat solutions, verified by a series of experiments[1] under general settings.

First, we study weight feature learning in dropout training. Previous works [9], [10], [11] find that, in the nonlinear training regime, input weights of hidden neurons (the input weight of a hidden neuron is a vector consisting of the weight from its input layer to the hidden layer and its bias term) are clustered into several groups under gradient flow training. The weights in each group have similar orientations, which is called *condensation*. By analyzing the implicit regularization terms, we theoretically find that dropout tends to find solutions with weight condensation. To verify the effect of dropout on condensation, we conduct experiments in the linear regime, such as neural tangent kernel initialization [12], where the weights are in proximity to the random initial values and condensation does not occur in common gradient descent training. We find that even in the linear regime, with dropout, weights show clear condensation in experiments, and for simplicity, we only show the output here (Fig. 1(a)). As condensation reduces the complexity of the NN, dropout may help the generalization by constraining the model's complexity.

Second, we study the flatness of the solution in dropout training. We theoretically show that the implicit regularization terms of dropout lead to a flat minimum. We experimentally verify the effect of the implicit regularization terms on flatness (Fig. 1(b)). As suggested by many existing works [13], [14], [15], flatter minima have a higher probability of better generalization and stability.

This work provides a comprehensive investigation into the implicit regularization of dropout and its associated implications.

[1]Code can be found at: https://github.com/sjtuzzw/torch_code_frame

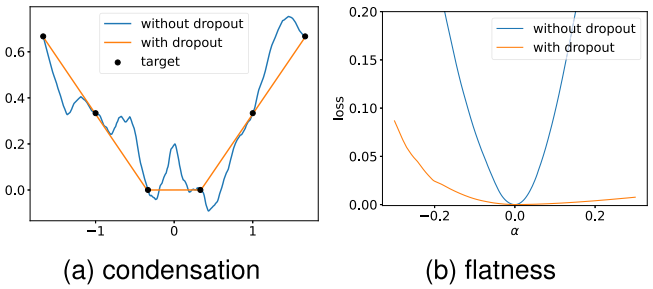


Fig. 1. Experimental results of training two-layer ReLU NNs trained with and without dropout. The width of the hidden layers is 1000, and the learning rate for all experiments is 1×10^{-3}. (a) The output of NNs with or without dropout. The black points represent the target points. (b) The loss value obtained by perturbing the network with or without dropout in a given random direction. α is the step size moving in the above direction.

Although our theoretical analysis mainly focuses on the dropout used in the last hidden layer, our experimental results extend to the general use of dropout in training NNs. Our results show that dropout has a distinct implicit regularization for facilitating weight condensation and finding flat minima, which may jointly improve the generalization performance of NNs.

II. Related Works

Dropout is proposed as a simple approach to prevent overfitting in the training of NNs, thus improving the generalization [1], [2]. Many works aim to understand the regularization of dropout.

Mcallester et al. [16] present PAC-Bayesian bounds for training with dropout. Wan et al. [17], Mou et al. [5], Zhai and Wang [18] and Arora et al. [6] derive Rademacher generalization bounds for training with dropout. Mianjy et al. [19] demonstrate that dropout training, coupled with logistic loss, achieves ε-suboptimality in test error in $O(1/\varepsilon)$ iterations, relying on an assumption that the data distribution is separable with a margin in a particular reproducing kernel Hilbert space. In our work, we focus on the training process of the model and the behavior of neurons rather than the complexity analysis.

Baldi et al. [20] introduce a general formalism for studying dropout on either units or connections, which finds an explicit formula for the expected loss for networks with sigmoid activation and dropout in any layer. Wager et al. [21] analyze the formulation of the implicit regularization of dropout for generalized linear models, such as logistic regression, and then they propose a regularization method that replaces dropout but can achieve results better than dropout. Wager et al. [22] show that dropout training can improve the exponent in the generalization bound for empirical risk minimization. Helmbold et al. [4] theoretically study the dropout regularization in logistic regression for linear classification, such as its convexity and monotonicity. Senen-Cerda and Sanders [23] demonstrate that, for deep neural networks with polynomially bounded activations with continuous derivatives and subjected to squared loss, the network weights converge to a unique stationary set of a projected system of ODEs. Zhang et al. [24] derive the stochastic modified equations that provide a weak approximation to the dynamics of two-layer NNs with dropout in training. In this work, we do not utilize ODE/SDE to study dropout but focus on the modified loss function.

Mianjy et al. [7] study the implicit bias of dropout for single hidden-layer linear neural networks. They show that dropout tends to make the norm of incoming/outgoing weight vectors of all hidden nodes equal. Bank et al. [25] verify that for autoencoders with a linear encoder, optimizing the encoder with dropout regularization leads to an equiangular tight frame. This result is established through an analogy drawn between a denoising autoencoder variant with a linear encoder and a signal encoding scheme. Our results on condensation can be regarded as a generalized result of those in Mianjy et al. [7] and Bank et al. [25] to two-layer non-linear networks. Based on the linear property, Mianjy et al. [7] further provide a complete characterization of the optimization landscape induced by dropout, while our work focuses on the sharpness/flatness of minima found by dropout for non-linear networks. Lengerich et al. [26] show that dropout acts as a regularizer that penalizes more on higher-order interactions among variables. Cavazza et al. [27] and Pal et al. [28] show that in the case of single hidden-layer linear networks, dropout is a regularizer that induces low-rank solutions. Mirzadeh et al. [29] demonstrate that the dropout networks behave like a network with a gating mechanism for continual learning tasks, such that for different tasks, different paths of the network are active. The sparse coding discussed in their work is also a potential behavior to reduce model complexity, which focuses on the number of activated neurons. In our work, we focus on the orientation similarity of different neurons, i.e., the condensation of neurons. Blanchet et al. [30] show that dropout training in generalized linear models corresponds to the minimax solution of a two-player, zero-sum game. This game involves an adversarial nature corrupting a statistician's covariates using a multiplicative nonparametric errors-in-variables model. Under the generalized linear model setting, they justify the ability of dropout training to enhance a predictor's out-of-sample performance, implying that dropout is beneficial to the generalization ability of the model from another perspective.

Wei et al. [31] formalize the expression of the loss function expectation for multi-layer networks with dropout and emphasize its distinctions from the original loss function. Additionally, they present the first-order moment expression of dropout loss based on the Jacobian and the Hessian matrices, intuitively suggesting a preference for the first-order moment to achieve flat solutions, drawing connections between the first-order moment and the Hessian matrix. Our theoretical work only focuses on the case when dropout is applied in the outermost layer, where we are able to derive an exact explicit formulation of the expectation of the dropout loss function, including the first-order moment and the second-order moment. Based on this explicit formulation, our work further enhances the understanding of the impact of dropout on flatness and condensation phenomena.

Several works have studied the disparity between the expected loss values with and without dropout in different situations, which is similar to the definition of R_1 in this study. In particular, Mianjy et al. [7] primarily investigate the disparity in the expected loss values with or without dropout in linear networks,

while Wei et al. [31], in the context of multi-layer nonlinear networks, approximates this disparity using Taylor expansion. In our work, we use experiments to validate the accuracy of the regularization of dropout obtained in our theoretical analysis. We further explore its impact on flatness and condensation both empirically and theoretically, where dropout is applied only in the final layer of the model.

The modified gradient flow is defined as the gradient flow that is close to discrete iterations of the original training path up to some high-order learning rate term [8]. Barrett et al. [32] derive the modified gradient flow of discrete full-batch gradient descent training as $\tilde{R}_{S,\mathrm{GD}}(\boldsymbol{\theta}) = R_S(\boldsymbol{\theta}) + (\varepsilon/4)\|\nabla R_S(\boldsymbol{\theta})\|^2 + O(\varepsilon^2)$, where $R_S(\boldsymbol{\theta})$ is the training loss on dataset S, ε is the learning rate and $\|\cdot\|$ denotes the l_2-norm. In a similar vein, Smith et al. [33] derive the modified gradient flow of stochastic gradient descent training as $\hat{R}_{S,\mathrm{SGD}}(\boldsymbol{\theta}) = R_S(\boldsymbol{\theta}) + (\varepsilon/4)\|\nabla R_S(\boldsymbol{\theta})\|^2 + (\varepsilon/4\,m)\sum_{i=0}^{m-1}\|\nabla R_{S,i}(\boldsymbol{\theta}) - \nabla R_S(\boldsymbol{\theta})\|^2 + O(\varepsilon^2)$, where $R_{S,i}(\boldsymbol{\theta})$ is the ith batch loss and the last term is also called "non-uniform" term [34]. Our work shows that there exist several distinct features between dropout and SGD. Specifically, in the limit of the vanishing learning rate, the modified gradient flow of dropout still has an additional implicit regularization term, whereas that of SGD converges to the full-batch gradient flow [35].

The parameter initialization of the network is important to determine the final fitting result of the network. Luo et al. [9], Zhou et al. [11] mainly identify the linear regime and the condensed regime for two-layer and three-layer wide ReLU NNs, respectively. In the linear regime, the training dynamics of NNs are approximately linear and similar to a random feature model with an exponential loss decay. In the condensed regime, active neurons are condensed at several discrete orientations, which may be an underlying reason why NNs outperform traditional algorithms.

Zhang et al. [36], Zhang et al. [37] show that NNs of different widths often exhibit similar condensation behavior, e.g., stagnating at a similar loss with almost the same output function. Based on this observation, they propose the embedding principle that the loss landscape of an NN contains all critical points of all narrower NNs. The embedding principle provides a basis for understanding why condensation occurs from the perspective of loss landscape.

Several works study the mechanism of condensation at the initial training stage, such as for ReLU network [38], [39] and network with continuously differentiable activation functions [10]. However, studying condensation throughout the whole training process is generally challenging, with dropout training being an exception. The regularization terms we derive in this work show that the dropout training tends to condense in the whole training process.

III. Preliminary

A. Deep Neural Networks

Consider a L-layer ($L \geq 2$) fully-connected neural network (FNN). We regard the input as the 0th layer and the output as the Lth layer. Let m_l represent the number of neurons in the lth layer. In particular, $m_0 = d$ and $m_L = d'$. For any $i, k \in \mathbb{N}$ and $i < k$, we denote $[i:k] = \{i, i+1, \ldots, k\}$. In particular, we denote $[k] := \{1, 2, \ldots, k\}$.

Given weights $W^{[l]} \in \mathbb{R}^{m_l \times m_{l-1}}$ and biases $b^{[l]} \in \mathbb{R}^{m_l}$ for $l \in [L]$, we define the collection of parameters $\boldsymbol{\theta}$ as a $2\,L$-tuple (an ordered list of $2\,L$ elements) whose elements are matrices or vectors

$$\boldsymbol{\theta} = (\boldsymbol{\theta}|_1, \ldots, \boldsymbol{\theta}|_L) = \left(\boldsymbol{W}^{[1]}, \boldsymbol{b}^{[1]}, \ldots, \boldsymbol{W}^{[L]}, \boldsymbol{b}^{[L]}\right),$$

where the lth layer parameters of $\boldsymbol{\theta}$ is the ordered pair $\boldsymbol{\theta}|_l = (\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}), \quad l \in [L]$. We may misuse notation and identify $\boldsymbol{\theta}$ with its vectorization $\mathrm{vec}(\boldsymbol{\theta}) \in \mathbb{R}^M$, where $M = \sum_{l=0}^{L-1}(m_l + 1)m_{l+1}$.

Given $\boldsymbol{\theta} \in \mathbb{R}^M$, the FNN function $\boldsymbol{f}_{\boldsymbol{\theta}}(\cdot)$ is defined recursively. First, we denote $\boldsymbol{f}_{\boldsymbol{\theta}}^{[0]}(\boldsymbol{x}) = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^d$. Then, for $l \in [L-1]$, $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}$ is defined recursively as $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}) = \sigma(\boldsymbol{W}^{[l]}\boldsymbol{f}_{\boldsymbol{\theta}}^{[l-1]}(\boldsymbol{x}) + \boldsymbol{b}^{[l]})$, where σ is a non-linear activation function. Finally, we denote

$$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{f}_{\boldsymbol{\theta}}^{[L]}(\boldsymbol{x}) = \boldsymbol{W}^{[L]}\boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}) + \boldsymbol{b}^{[L]}.$$

For notational simplicity, we denote

$$\boldsymbol{f}_{\boldsymbol{\theta}}^{j}(\boldsymbol{x}_i) = \boldsymbol{W}_j^{[L]} f_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i),$$

where $\boldsymbol{f}_{\boldsymbol{\theta}}^{j}(\boldsymbol{x}_i), \boldsymbol{W}_j^{[L]} \in \mathbb{R}^{m_L}$ is the jth column of $\boldsymbol{W}^{[L]}$, and $f_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i)$ is the jth element of vector $\boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}_i)$. In this work, we denote the l_2-norm as $\|\cdot\|$ for convenience.

B. Loss Function

The training data set is denoted as $S = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $\boldsymbol{y}_i \in \mathbb{R}^{d'}$. For simplicity, we assume an unknown function $\boldsymbol{y}$ satisfying $\boldsymbol{y}(\boldsymbol{x}_i) = \boldsymbol{y}_i$ for $i \in [n]$. The empirical risk reads as

$$R_S(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}), \boldsymbol{y}(\boldsymbol{x}_i)), \tag{1}$$

where the loss function $\ell(\cdot,\cdot)$ is differentiable and the derivative of ℓ with respect to its first argument is denoted by $\nabla\ell(\boldsymbol{y}, \boldsymbol{y}^*)$. The error with respect to data sample $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ is defined as

$$\boldsymbol{e}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) = \boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - \boldsymbol{y}_i.$$

For notation simplicity, we denote $\boldsymbol{e}(\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) = \boldsymbol{e}_{\boldsymbol{\theta},i}$.

C. Dropout

For $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}) \in \mathbb{R}^{m_l}$, we randomly sample a scaling vector $\boldsymbol{\eta} \in \mathbb{R}^{m_l}$ with coordinates of $\boldsymbol{\eta}$ are sampled i.i.d that

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

where $p \in (0,1]$, $k \in [m_l]$ indices the coordinate of $\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x})$. It is important to note that $\boldsymbol{\eta}$ is a zero-mean random variable. We then apply dropout by computing

$$\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{[l]}(\boldsymbol{x}) = (\boldsymbol{1} + \boldsymbol{\eta}) \odot \boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}),$$

and use $\boldsymbol{f}^{[l]}_{\boldsymbol{\theta},\boldsymbol{\eta}}(\boldsymbol{x})$ instead of $\boldsymbol{f}^{[l]}_{\boldsymbol{\theta}}(\boldsymbol{x})$. Here we use $\odot$ for the Hadamard product of two matrices of the same dimension. To simplify notation, we let $\boldsymbol{\eta}$ denote the collection of such vectors over all layers. We denote the output of model $\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$ on input $\boldsymbol{x}$ using dropout noise $\boldsymbol{\eta}$ as $\boldsymbol{f}^{\text{drop}}_{\boldsymbol{\theta},\boldsymbol{\eta}}(\boldsymbol{x})$. The empirical risk associated with the network with dropout layer $\boldsymbol{f}^{\text{drop}}_{\boldsymbol{\theta},\boldsymbol{\eta}}$ is denoted by $R^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta})$, given by

$$R^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta}) = \frac{1}{n}\sum_{i=1}^{n}\ell(\boldsymbol{f}^{\text{drop}}_{\boldsymbol{\theta},\boldsymbol{\eta}}(\boldsymbol{x}_i),\boldsymbol{y}(\boldsymbol{x}_i)). \tag{2}$$

D. Condensation

For a given NN, the input weight of neuron j in the lth layer $(\boldsymbol{W}^{[l]}_j, \boldsymbol{b}^{[l]}_j)$ is vectorized as $\boldsymbol{\theta}^{[l]}_j \in \mathbb{R}^{m_l+1}$. The condensation for any two neurons in the same layer in this work is defined as follows.

Definition 1 (condensation): Neuron i and neuron j in the lth layer are condensed if $\boldsymbol{\theta}^{[l]}_i$ is parallel with $\boldsymbol{\theta}^{[l]}_j$.

To characterize the degree of condensation for a neural network, we define the effective ratio for each layer.

Definition 2 (effective ratio): Let $U^{[l]} = \{\boldsymbol{u}^{[l]}_k\}_{k=1}^{m'_l}$ be the set of vectors such that for any input weight $\boldsymbol{\theta}^{[l]}_j$, there exists an element $\boldsymbol{u} \in U^{[l]}$ satisfying $\boldsymbol{u} \,//\, \boldsymbol{\theta}^{[l]}_j$. The effective neuron number m^{eff}_l of the lth layer is defined as the minimal size of all possible $U^{[l]}$. The effective ratio is defined as m^{eff}_{l}/m_l.

In our experiments, we use cosine similarity to characterize the condensation.

Definition 3 (cosine similarity): The cosine similarity between two vectors $\boldsymbol{u} \in \mathbb{R}^d$ and $\boldsymbol{v} \in \mathbb{R}^d$ is defined as

$$D_{\cos}(\boldsymbol{u},\boldsymbol{v}) = \frac{\boldsymbol{u}^\intercal\boldsymbol{v}}{(\boldsymbol{u}^\intercal\boldsymbol{u})^{1/2}(\boldsymbol{v}^\intercal\boldsymbol{v})^{1/2}}. \tag{3}$$

In this work, we admit $\boldsymbol{u} \,//\, \boldsymbol{\theta}^{[l]}_j$ if $D_{\cos}(\boldsymbol{u},\boldsymbol{\theta}^{[l]}_j) > 0.95$. Note that the selection of the hyper-parameter 0.95 is not sensitive.

IV. Modified Gradient Flow

In this section, we theoretically analyze the implicit regularization effect of dropout. We derive the modified gradient flow of dropout in the sense of expectation. We first summarize the settings and provide the necessary definitions used for our theoretical results below. *Note that, the settings of our experiments are much more general.*

Setting 1 (dropout structure): Consider an L-layer ($L \geq 2$) FNN with only one dropout layer after the $(L-1)$th layer of the network,

$$\boldsymbol{f}^{\text{drop}}_{\boldsymbol{\theta},\boldsymbol{\eta}}(\boldsymbol{x}) = \boldsymbol{W}^{[L]}(\boldsymbol{1}+\boldsymbol{\eta}) \odot \boldsymbol{f}^{[L-1]}_{\boldsymbol{\theta}}(\boldsymbol{x}) + \boldsymbol{b}^{[L]}.$$

Setting 2 (loss function): Take the mean squared error (MSE) as our loss function,

$$R_S(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}(\boldsymbol{f}(\boldsymbol{x}_i,\boldsymbol{\theta}) - \boldsymbol{y}_i)^2.$$

Setting 3 (network structure): For convenience, we set the model output dimension to one, i.e., $m_L = 1$.

In the following, we introduce two key terms that play an important role in our theoretical results:

$$R_1(\boldsymbol{\theta}) := \frac{1-p}{2np}\sum_{i=1}^{n}\sum_{j=1}^{m_{L-1}}\|\boldsymbol{W}^{[L]}_j f^{[L-1]}_{\boldsymbol{\theta},j}(\boldsymbol{x}_i)\|^2, \tag{4}$$

$$R_2(\boldsymbol{\theta}) := \frac{\varepsilon}{4}\mathbb{E}_{\boldsymbol{\eta}}\|\nabla_{\boldsymbol{\theta}} R^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta})\|^2, \tag{5}$$

where $\boldsymbol{W}^{[L]}_j \in \mathbb{R}^{m_L}$ is the jth column of $\boldsymbol{W}^{[L]}$, $f^{[L-1]}_{\boldsymbol{\theta},j}(\boldsymbol{x}_i)$ is the jth element of $\boldsymbol{f}^{[L-1]}_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$, $\mathbb{E}_{\boldsymbol{\eta}}$ is the expectation with respect to $\boldsymbol{\eta}$, and ε is the learning rate.

Based on the above settings, we obtain a modified equation based on dropout gradient flow.

Lemma 1 (the expectation of dropout loss): Given an L-layer FNN with dropout $\boldsymbol{f}^{\text{drop}}_{\boldsymbol{\theta},\boldsymbol{\eta}}(\boldsymbol{x})$, under Settings 1–3, we have the expectation of dropout MSE

$$\mathbb{E}_{\boldsymbol{\eta}}(R^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta})) = R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta}).$$

Based on the above lemma, we proceed to study the discrete iterate training of gradient descent with dropout, resulting in the derivation of the modified gradient flow of dropout training.

Modified Gradient Flow of Dropout: Under Settings 1–3, the mean iterate of $\boldsymbol{\theta}$, with a learning rate $\varepsilon \ll 1$, stays close to the path of gradient flow on a modified loss $\dot{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{\theta}}\tilde{R}^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta})$, where the modified loss $\tilde{R}^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta})$ satisfies

$$\mathbb{E}_{\boldsymbol{\eta}}\tilde{R}^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta}) \approx R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta}) + R_2(\boldsymbol{\theta}). \tag{6}$$

Contrary to SGD [33], the $R_1(\boldsymbol{\theta})$ term is independent of the learning rate ε, thus the implicit regularization of dropout still affects the gradient flow even as the learning rate ε approaches zero. In Section VI, we show the $R_1(\boldsymbol{\theta})$ term makes the network tend to find solutions with lower complexity, that is, solutions with weight condensation, which is also illustrated and supported numerically. In Section VII, we show the $R_1(\boldsymbol{\theta})$ term plays a more important role in improving the generalization and flatness of the model than the $R_2(\boldsymbol{\theta})$ term.

V. Numerical Verification of Implicit Regularization Terms

In this section, we numerically verify the validity of two implicit regularization terms, i.e., $R_1(\boldsymbol{\theta})$ defined in (4) and $R_2(\boldsymbol{\theta})$ defined in (5), under more general settings than out theoretical results. The detailed experimental settings can be found in Appendix A, available online.

A. Validation of the Effect of $R_1(\boldsymbol{\theta})$

As $R_1(\boldsymbol{\theta})$ is independent of the learning rate and $R_2(\boldsymbol{\theta})$ vanishes in the limit of zero learning rate, we select a small learning rate to verify the validity of $R_1(\boldsymbol{\theta})$. According to (6), the modified equation of dropout training dynamics can be approximated by $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$ when the learning rate ε is sufficiently small. Therefore, we verify the validity of $R_1(\boldsymbol{\theta})$ through the similarity of the NN trained by the two loss functions, i.e., $R^{\text{drop}}_S(\boldsymbol{\theta},\boldsymbol{\eta})$ and $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$ under a small learning rate.

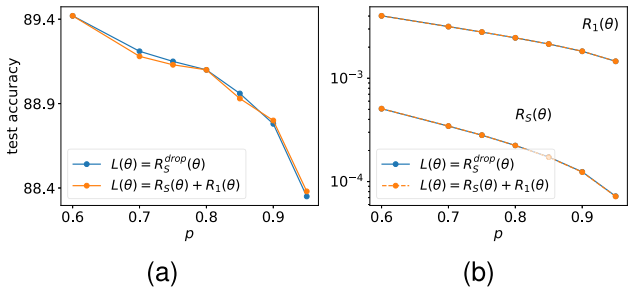


Fig. 2. Two-layer NNs of width 1000 for the classification of the first 1000 images of MNIST dataset, utilizing two distinct loss functions: $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ and $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$. To study the impact of different dropout rates on the performance of the networks, we conduct experiments with varying dropout rates while maintaining a constant learning rate of $\varepsilon = 5 \times 10^{-3}$. (a) The test accuracy. (b) The value of $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ and $R_1(\boldsymbol{\theta})$.

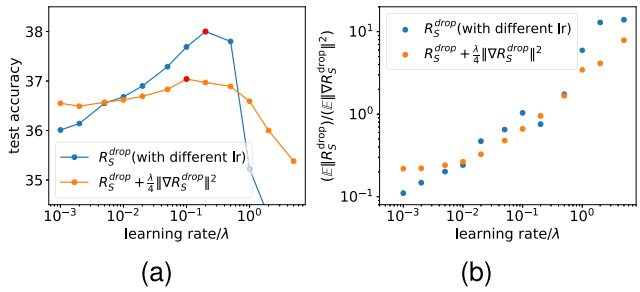


Fig. 3. Classify the first 1000 images of CIFAR-10 by training VGG-9 under a specific loss function by GD. For loss function $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$, we train the NNs with various learning rates ε. For loss function $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) + (\lambda/4)\|\nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|^2$, we train the NNs with various regularization coefficient λ, while keeping the learning rate fixed at a small value of $\varepsilon = 5 \times 10^{-3}$. (a) The test accuracy of the network under different learning rates and regularization coefficients. The red dots indicate the location of the maximum test accuracy of the NNs obtained by training with both two loss functions. (b) The $(\mathbb{E}_{\boldsymbol{\eta}}\|R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|)/(\mathbb{E}_{\boldsymbol{\eta}}\|\nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|^2)$ value of the resulting model in (a) under different learning rates ε or regularization coefficients λ.

Fig. 2(a) presents the test accuracy of two losses trained under different dropout rates. For the network trained with $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$, there is no dropout layer, and the dropout rate affects the weight of $R_1(\boldsymbol{\theta})$ in the loss function. For different dropout rates, the networks obtained by the two losses above exhibit similar test accuracy. It is worth mentioning that for the network trained with $R_S(\boldsymbol{\theta})$, the obtained accuracy is only 79%, which is significantly lower than the accuracy of the network trained through the two loss functions above (over 88% in Fig. 2(a)). In Fig. 2(b), we show the values of $R_S(\boldsymbol{\theta})$ and $R_1(\boldsymbol{\theta})$ for the two networks at different dropout rates. Note that for the network obtained by $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ training, we can calculate the two terms through the network's parameters. It can be seen that for different dropout rates, the values of $R_S(\boldsymbol{\theta})$ and $R_1(\boldsymbol{\theta})$ of the two networks are almost indistinguishable.

B. Validation of the Effect of $R_2(\boldsymbol{\theta})$

As shown in Theorem IV, the modified loss $\tilde{R}_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ satisfies the equation

$$\mathbb{E}_{\boldsymbol{\eta}} \tilde{R}_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \mathbb{E}_{\boldsymbol{\eta}} \left(R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) + \frac{\varepsilon}{4} \|\nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|^2 \right).$$

In order to validate the effect of $R_2(\boldsymbol{\theta})$ in the training process, we verify the equivalence of the following two training methods: (i) training networks with a dropout layer by MSE $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ with different learning rates ε; (ii) training networks with a dropout layer by MSE with an explicit regularization

$$R_S^{\text{regu}}(\boldsymbol{\theta}, \boldsymbol{\eta}) := R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) + (\lambda/4)\|\nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|^2,$$

with different values of λ and a fixed learning rate much smaller than ε. The exact form of $R_2(\boldsymbol{\theta})$ has an expectation with respect to $\boldsymbol{\eta}$, but in this subsection, we ignore this expectation in experiments for convenience.

As shown in Fig. 3, we train the NNs by the MSE $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ with different learning rates (blue), and the regularized MSE $R_S^{\text{regu}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ with a fixed small learning rate and different values of λ (orange). In Fig. 3(a), the learning rate ε and the regularization coefficient λ are close when they reach their corresponding maximum test accuracy (red point). In addition, as shown in Fig. 3(b), we study the value of $\mathbb{E}_{\boldsymbol{\eta}}\|R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|/\mathbb{E}_{\boldsymbol{\eta}}\|\nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|^2$ under different learning rates (blue) and regularization coefficients (orange). In practical experiments, we take 3,000 different dropout noises $\boldsymbol{\eta}$ to approximate the expectation after the training process. The results indicate that the same learning rate ε and regularization coefficient λ result in similar ratios.

Due to the computational cost of full-batch GD, we only use a few training samples in the above experiments. We conduct similar experiments with dropout under different learning rates and regularization coefficients using SGD as detailed in Appendix D-A, available online.

VI. Dropout Facilitates Condensation

A condensed network, which refers to a network with neurons having aligned input weights, is equivalent to another network with a reduced width [9], [10]. Therefore, the effective complexity of the network is smaller than its superficial appearance. Such low effective complexity may be an underlying reason for good generalization. In addition, the embedding principle [36], [37], [40] shows that although the condensed network is equivalent to a smaller one in the sense of approximation, it has more degeneracy and more descent directions that may lead to a simpler training process.

In this section, we experimentally and theoretically study the effect of dropout on the condensation phenomenon.

A. Experimental Results

To empirically validate the effect of dropout on condensation, we examine ReLU and tanh activations in one-dimensional and high-dimensional fitting problems, as well as image classification problems. Due to space limitations, detailed experimental settings and some experimental results are left in Appendices A, D, available online.

1) Network With One-Dimensional Input: We train a tanh NN with 1,000 hidden neurons for the one-dimensional fitting

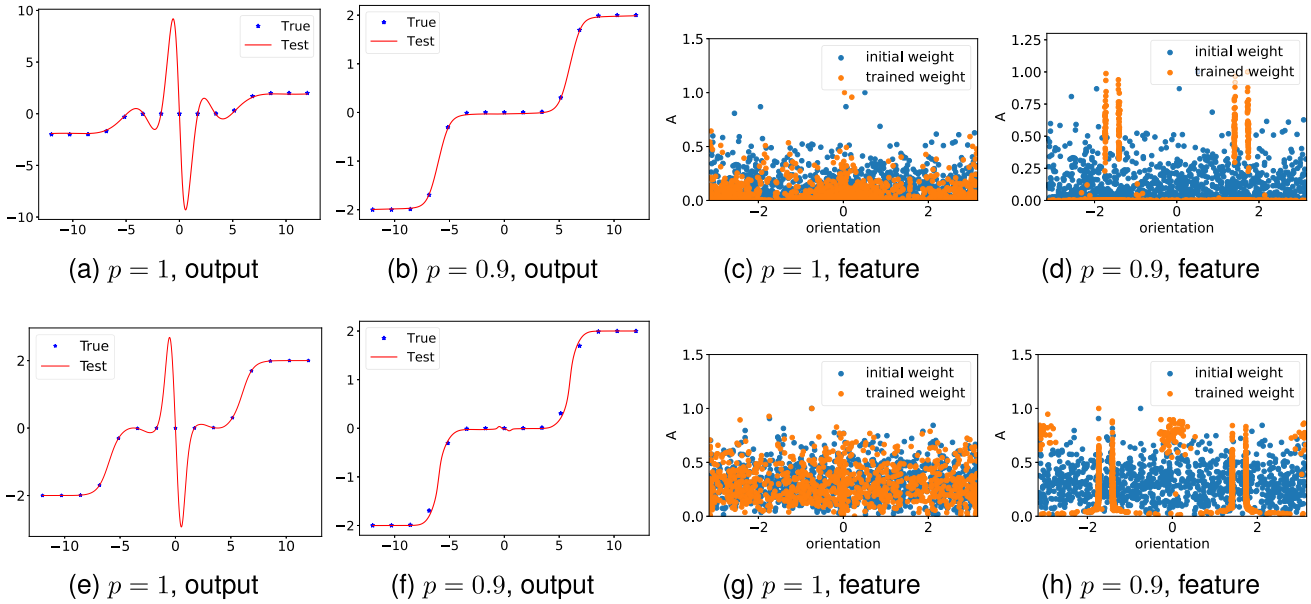


Fig. 4. Tanh NNs outputs and features under different dropout rates. The width of the hidden layers is 1,000, and the learning rate for different experiments is 1×10^{-3}. In (c, d, g, h), blue dots and orange dots are for the weight feature distribution at the initial and final training stages, respectively. The top row is the result of two-layer networks, with the dropout layer after the hidden layer. The bottom row is the result of three-layer networks, with the dropout layer between the two hidden layers and after the last hidden layer. Refer to Appendix D-B, available online, for further experiments on ReLU NNs.

problem to fit the data shown in Fig. 4 with MSE. Additional experimental verifications on ReLU NNs are provided in Appendix D, available online. The experiments performed with and without dropout under the same initialization can both well fit the training data. In order to clearly study the effect of dropout on condensation, we take the parameter initialization distribution in the linear regime [9], where condensation does not occur without additional constraints. The dropout layer is used after the hidden layer of the two-layer network (top row) and used between the hidden layers and after the last hidden layer of the three-layer network (bottom row). Upon close inspection of the fitting process, we find that the output of NNs trained without dropout in Fig. 4(a) and (e) has much more oscillation than the output of NNs trained with dropout in Fig. 4(b) and (f). To better understand the underlying effect of dropout, we study the feature of parameters.

The parameter pair $(a_j, \boldsymbol{w}_j)$ of each neuron can be separated into a unit orientation feature $\hat{\boldsymbol{w}}_j = \boldsymbol{w}_j / \|\boldsymbol{w}_j\|_2$ and an amplitude $A_j = |a_j| \|\boldsymbol{w}_j\|_2$ indicating its contribution[2] to the output, i.e., $(\hat{\boldsymbol{w}}_j, A_j)$. For a one-dimensional input, $\boldsymbol{w}_j$ is two-dimensional due to the incorporation of bias. Therefore, we use the angle to the x-axis in $[-\pi, \pi)$ to indicate the orientation of each $\hat{\boldsymbol{w}}_j$. For simplicity, for the three-layer network with one-dimensional input, we only consider the input weight of the first hidden layer. The scatter plots of $\{(\hat{\boldsymbol{w}}_j, |a_j|)\}_{j=1}^m$ and $\{(\hat{\boldsymbol{w}}_j, \|\boldsymbol{w}_j\|_2)\}_{j=1}^m$ of tanh activation are presented in Appendix D-C, available online, to eliminate the impact of the non-homogeneity of tanh activation.

[2]Due to the homogeneity of ReLU neurons, this amplitude can accurately describe the contribution of ReLU neurons. For tanh neurons, the amplitude has a certain positive correlation with the contribution of each neuron.

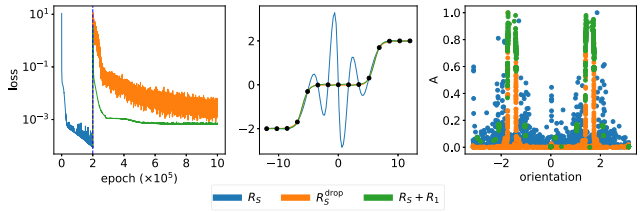
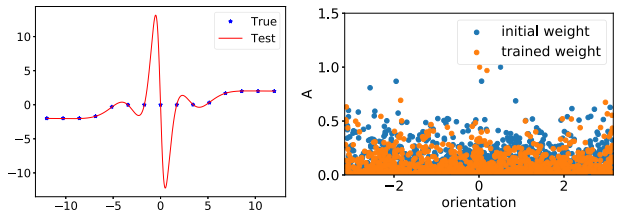


Fig. 5. Compared dynamics are initialized at model found by $R_S(\boldsymbol{\theta})$, marked by the vertical dashed line in iteration 200000 with two-layer tanh NN. Left: The loss trajectory under different loss functions. MIddle: The output of the model trained by $R_S(\boldsymbol{\theta})$ (blue) and the model trained by $R_S^{\text{drop}}(\boldsymbol{\theta})$ (orange) and $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$ (green) initialized at model found by $R_S(\boldsymbol{\theta})$. The black points are the target points. Right: The feature of the model trained by $R_S(\boldsymbol{\theta})$ (blue) and the model trained by $R_S^{\text{drop}}(\boldsymbol{\theta})$ (orange) and $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$ (green) initialized at model found by $R_S(\boldsymbol{\theta})$.

The scatter plots of $\{(\hat{\boldsymbol{w}}_j, A_j)\}_{j=1}^m$ of the NNs are shown in Fig. 4(c), (d), (g), and (h). For convenience, we normalize the feature distribution of each model parameter such that the maximum amplitude of neurons in each model is 1. Compared with the initial weight distribution (blue), the weight trained without dropout (orange) is close to its initial value. However, for the NNs trained with dropout, the parameters after training are significantly different from the initialization, and the non-zero parameters tend to condense on several discrete orientations, showing a condensation tendency.

In addition, we study the stability of the model trained with the loss function $R_S(\boldsymbol{\theta})$ under the two loss functions $R_S^{\text{drop}}(\boldsymbol{\theta})$ and $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$. As shown in the left panel of Fig. 5, we use $R_S(\boldsymbol{\theta})$ as the loss function to train the model before the dashed line where when $R_S(\boldsymbol{\theta})$ is small, and we then replace



(a) batch size $= 2$, output (b) batch size $= 2$, feature

Fig. 6. Two-layer tanh NN output and feature with a batch size of 2. The width of the hidden layer is 1,000, and the learning rate is 1×10^{-3}. In (b), blue dots and orange dots are for the weight feature distribution at the initial and final training stages, respectively.

the loss function by $R_S^{\text{drop}}(\boldsymbol{\theta})$ or $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$. The outputs and features of the models trained with these three loss functions are shown in the middle and right panels of Fig. 5, respectively. The results reveal that dropout ($R_1(\boldsymbol{\theta})$ term) aids the training process in escaping from the minima obtained by $R_S(\boldsymbol{\theta})$ training and finding a condensed solution.

One may wonder if any noise injected into the training process could lead to condensation. We also perform similar experiments for SGD. As shown in Fig. 6, no significant condensation occurs even in the presence of noise during training. Therefore, the experiments in this section reveal the special characteristic of dropout that facilitate condensation.

2) Network With High-Dimensional Input: We conducted further investigation into the effect of dropout on high-dimensional two-layer tanh NNs under the teacher-student setting. Specifically, we utilize a two-layer tanh NN with only one hidden neuron and 10-dimensional input as the target function. The orientation similarity of two neurons is calculated by taking the inner product of their normalized weights. As shown in Fig. 7(a) and (b), for the NN with dropout, the neurons of the network have only two orientations, indicating the occurrence of condensation, while the NN without dropout does not exhibit such a phenomenon.

To visualize the condensation during the training process, we define the ratio of effective neurons as follows.

We study the training process of using ResNet-18 to learn CIFAR-10. As shown in Fig. 7(c), NNs with dropout tend to have lower effective ratios, and thus tend to exhibit condensation.

3) Dropout Improves Generalization: As the effective neuron number of a condensed network is much smaller than its actual neuron number, it is expected to generalize better. To verify this, we use a two-layer tanh network with 1,000 neurons to learn a teacher two-layer tanh network with two neurons. The number of free parameters in the teacher network is 6. As shown in Fig. 8, the model with dropout generalizes well when the number of samplings is larger than 6, while the model without dropout generalizes badly. This result is consistent with the rank analysis of non-linear models [41].

4) Dropout is an Ideal Way to Induce Condensation: Existing literatures show that condensation is an important and general phenomenon in the non-linear training process of neural networks. Condensed neurons have highly similar outputs. Therefore, a network with a large number of condensed neurons has a rather small effective complexity that can control the generalization of a model. However, previous studies require the network parameters with very small initialization to achieve condensation, where the learning trajectory will experience regions that are very close to saddle points, resulting in an extremely long training time. For example in Fig. 9(a), the model with a small initialization is trapped by a saddle point and takes 9×10^5 epochs to escape by gradient descent training.

Dropout offers an ideal training method to induce condensation phenomena without suffering from the long training process. For example in Fig. 9(a), with the same setup as Fig. 4, the model trained by gradient descent with dropout reaches a small loss (10^{-5}) in approximately 1×10^5 epochs, which is much less than that trained by gradient descent with small initialization. Meanwhile, the model with dropout exhibits a flat output function, as shown in Fig. 9(b). In contrast, as illustrated in Fig. 4, a model with such a large initialization without dropout produces an oscillating output function with no condensation.

B. The Effect of $R_1(\boldsymbol{\theta})$ on Condensation

As can be seen from the implicit regularization term $R_1(\boldsymbol{\theta})$, dropout regularization imposes an additional l_2-norm constraint on the output of each neuron. The constraint has an effect on condensation. We illustrate the effect of $R_1(\boldsymbol{\theta})$ by a toy example of a two-layer ReLU network.

We use the following two-layer ReLU network to fit a one-dimensional function:

$$f_{\boldsymbol{\theta}}(x) = \sum_{j=1}^{m} a_j \sigma(\boldsymbol{w}_j \cdot \boldsymbol{x}) = \sum_{j=1}^{m} a_j \sigma(w_j x + b_j),$$

where a_j, w_j, b_j are the trainable parameters of the two-layer network, $\boldsymbol{x} := (x, 1)^\intercal \in \mathbb{R}^2$, $\boldsymbol{w}_j := (w_j, b_j) \in \mathbb{R}^2$, $\sigma(x) = \text{ReLU}(x)$. For simplicity, we set $m = 2$, and suppose the network can perfectly fit a training data set of two data points generated by a target function of $\sigma(\boldsymbol{w}^* \cdot \boldsymbol{x})$, denoted as $\boldsymbol{o}^* := (\sigma(\boldsymbol{w}^* \cdot \boldsymbol{x}_1), \sigma(\boldsymbol{w}^* \cdot \boldsymbol{x}_2))$. We further assume $\boldsymbol{w}^* \cdot \boldsymbol{x}_i > 0, i = 1, 2$. Denote the output of the jth neuron over samples as

$$\boldsymbol{o}_j = (a_j \sigma(\boldsymbol{w}_j \cdot \boldsymbol{x}_1), a_j \sigma(\boldsymbol{w}_j \cdot \boldsymbol{x}_2)).$$

The network output should equal to the target on the training data points after long enough training, i.e.,

$$\boldsymbol{o}^* = \boldsymbol{o}_1 + \boldsymbol{o}_2.$$

There are infinitely many pairs of $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ that can fit $\boldsymbol{o}^*$ well. However, the $R_1(\boldsymbol{\theta})$ term leads the training to a specific pair. $R_1(\boldsymbol{\theta})$ can be written as

$$R_1(\boldsymbol{\theta}) = \|\boldsymbol{o}_1\|^2 + \|\boldsymbol{o}_2\|^2,$$

and the components of $\boldsymbol{o}_j$ perpendicular to $\boldsymbol{o}^*$ need to cancel each other at the well-trained stage to minimize $R_1(\boldsymbol{\theta})$. As a result, $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ need to be parallel with $\boldsymbol{o}^*$, i.e., $\boldsymbol{w}_1 // \boldsymbol{w}_2 // \boldsymbol{w}^*$, which is the condensation phenomenon.

In the following, we show that minimizing $R_1(\boldsymbol{\theta})$ term can lead to condensation under several settings. We first give some

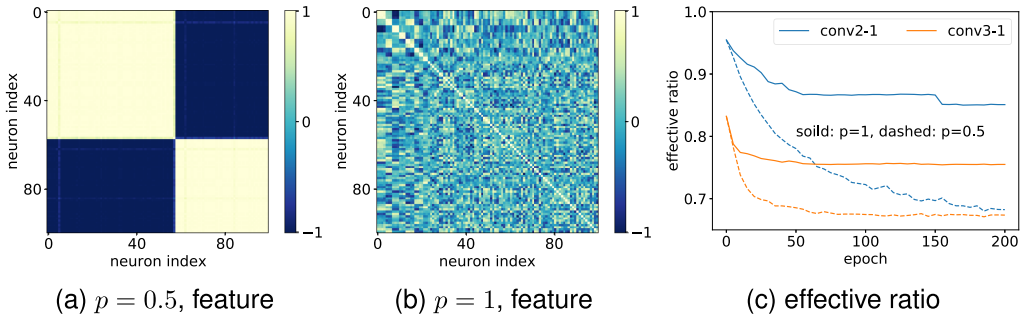


Fig. 7. Sparsity in High-Dimensional NNs with different dropout rates. (a, b) Parameter features of the two-layer tanh NNs with and without dropout. (c) Effective ratio with and without dropout under the task of CIFAR-10 classification with ResNet-18. Conv2-1 and conv3-1 represent the parameters of the first convolutional layer of the second block and the third block of the ResNet, respectively.

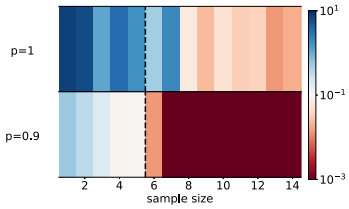


Fig. 8. Average test error of the two-layer tanh NNs (color) versus the number of samples (abscissa) for different dropout rates (ordinate). For all experiments, the width of the hidden layer is 1,000, and the learning rate is 1×10^{-4} with the Adam optimizer. Each test error is averaged over 10 trials with random initialization. Refer to Appendix D-B, available online, for further experiments on ReLU NNs.

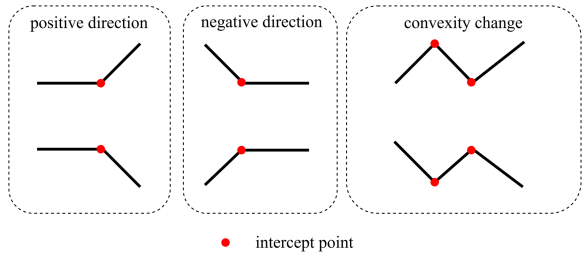


Fig. 10. Schematic diagram of some definitions associated with ReLU neurons.

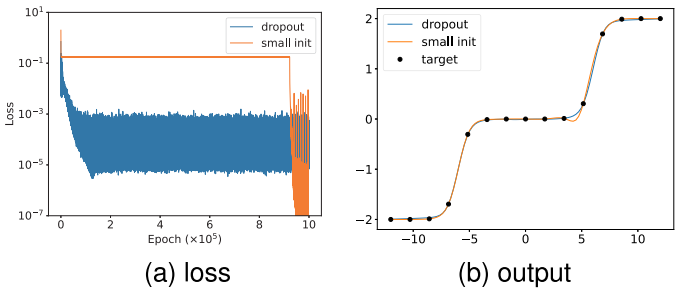


Fig. 9. Comparison of loss and output between the model trained by gradient descent with small initialization (orange) and the model trained by dropout with normal scale initialization (blue). The setup is the same as Fig. 4.

definitions that capture the characteristic of ReLU neurons (also shown in Fig. 10).

Definition 4 (convexity change of ReLU NNs): Consider piecewise linear function $f(t), t \in \mathbb{R}$, and its linear interval sets $\{[t_i, t_{i+1}]\}_{i=1}^{T}$. For any two intervals $[t_i, t_{i+2}], [t_{i+1}, t_{i+3}], i \in [T-3]$, if on one of the intervals, f is convex and on the other f is concave, then we call there exists a convexity change.

Definition 5 (direction and intercept point of ReLU neurons): For a one-dimensional ReLU neuron $a_j\sigma(w_j x + b_j)$, its direction is defined as $\text{sign}(w_j)$, and its intercept point is defined as $x = -\frac{b_j}{w_j}$.

Drawing inspiration from the methodology employed to establish the regularization effect of label noise SGD [42], we show that under the setting of two-layer ReLU NN and one-dimensional input data, the implicit bias of $R_1(\boldsymbol{\theta})$ term corresponds to "simple" functions that satisfy two conditions: (i) they have the minimum number of convexity changes required to fit the training points, and (ii) if the intercept points of neurons are in the same inner interval, and the neurons have the same direction, then their intercept points are identical.

*Theorem 1 (*the effect of $R_1(\boldsymbol{\theta})$ on facilitating condensation*).* Consider the following two-layer ReLU NN,

$$f_{\boldsymbol{\theta}}(x) = \sum_{j=1}^{m} a_j \sigma(w_j x + b_j) + ax + b,$$

trained with a one-dimensional dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$, where $x_1 < x_2 < \cdots < x_n$. When the MSE of training data $R_S(\boldsymbol{\theta}) = 0$, if any of the following two case happens:

(i) the number of convexity changes of NN in (x_1, x_n) can be reduced while $R_S(\boldsymbol{\theta}) = 0$;

(ii) there exist two neurons with indexes $k_1 \neq k_2$, such that they have the same sign, i.e., $\text{sign}(w_{k_1}) = \text{sign}(w_{k_2})$, and different intercept points in the same interval, i.e., $-b_{k_1}/w_{k_1}, -b_{k_2}/w_{k_2} \in [x_i, x_{i+1}]$, and $-b_{k_1}/w_{k_1} \neq -b_{k_2}/w_{k_2}$ for some $i \in [2 : n-2]$;

then there exists parameters $\boldsymbol{\theta}'$, an infinitesimal perturbation of $\boldsymbol{\theta}$, s.t.,

(i) $R_S(\boldsymbol{\theta}') = 0$;

(ii) $R_1(\boldsymbol{\theta}') < R_1(\boldsymbol{\theta})$.

Remark 1: Theorem 1 implies that for an over-parameterized network with many global minima, if one of the two cases above occurs, there will be a global minimum with a smaller value of

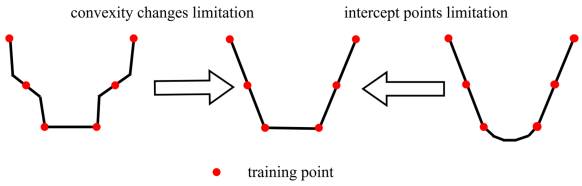


Fig. 11. Schematic diagram of the effect of $R_1(\boldsymbol{\theta})$ on the constraint of the convexity changes (left figure, case (i) in Theorem 1) and the intercept points (right figure, case (ii) in Theorem 1).

R_1 than the current one. In other words, if a global minimum minimizes R_1 over all minima, then, both cases in the theorem can not happen, that is, the number of convexity changes of NN in (x_1, x_n) is minimal, and the model has at most 2 intercept points between any interval $[x_i, x_{i+1}]$, for any $i \in [2 : n-2]$.

Remark 2: The additional linear function $ax + b$ in Theorem 1 is added for maintaining zero loss when studying certain situations in case (i). In this theorem, we reduce R_1 by moving the intercept points of both neurons in different directions simultaneously. Since the two neurons have opposite directions in some situations, i.e., their input weights have opposite signs, the movement of the intercept points of the two neurons will affect the network output globally (one neuron will affect one side). To cancel this global effect in order to maintain zero loss, one has to add a global function. We prove by construction that a linear function can do the trick. Due to the additivity of linear functions, the linear functions required for reducing multiple convexity changes can be equivalent to one linear function. It should be noted that this linear function can be composed of two ReLUs, which means that the original network is still essentially a two-layer ReLU neural network. However, for the convenience of proof, we use the network with $ax + b$ in the output. More discussion about the conditions in the theorem can be found in Appendix B, available online.

The example in the left curve in Fig. 11 shows the case (i) of Theorem 1, that is, the convexity change can be reduced while the loss stays zero. The example in the right curve in Fig. 11 shows the case (ii) of Theorem 1, that is, there are more than two intercept points without convexity change between the third and the fourth points. Both right and left curves can be reduced to the middle one with smaller R_1 and zero training loss. Therefore, the situation in which both case (i) and case (ii) do not happen can facilitate condensation. It's also worth noting that condensation can occur in either case. However, the effective ratio can be further reduced (more condensed) when either case is reduced to the middle one. Meanwhile, not all functions trained with dropout exhibit obvious condensation. For example, a function with dropout shows no condensation when trained using only one data point. However, for general datasets, such as the example shown in Figs. 1 and 4, NNs reach the condensed solution due to the constraint of the convexity changes and the intercept points (also illustrated in Fig. 11).

Although the current study only demonstrates the result for ReLU NNs, it is expected that for general activation functions, such as tanh, the $R_1(\boldsymbol{\theta})$ term also has the effect on facilitating condensation, which is left for future work. This is also confirmed by the experimental results conducted on the tanh NNs above. Furthermore, it is believed that the linear term $ax + b$ utilized to ensure $R_S(\boldsymbol{\theta}) = 0$ in certain cases is not a fundamental requirement. Our experiments illustrate that neural networks without the linear term also exhibit the condensation phenomenon.

VII. Implicit Regularization of Dropout on the Flatness of Solution

Understanding the mechanism by which dropout improves the generalization of NNs is of great interest and significance. In this section, we study the flatness of the minima found by dropout as inspired by the study of SGD on generalization [13]. Our primary focus is to study the effect of $R_1(\boldsymbol{\theta})$ and $R_2(\boldsymbol{\theta})$ on the flatness of loss landscape and network generalization.

A. Dropout Finds Flatter Minima

We first study the effect of dropout on model flatness and generalization. For a fair comparison of the flatness between different models, we employ the approach used in Li et al. [43] as follows. To obtain a direction for a network with parameters $\boldsymbol{\theta}$, we begin by producing a random Gaussian direction vector $\boldsymbol{d}$ with dimensions compatible with $\boldsymbol{\theta}$. Then, we normalize each filter in $\boldsymbol{d}$ to have the same norm as the corresponding filter in $\boldsymbol{\theta}$. For FNNs, each layer can be regarded as a filter, and the normalization process is equivalent to normalizing the layer, while for convolutional neural networks (CNNs), each convolution kernel may have multiple filters, and each filter is normalized individually. Thus, we obtain a normalized direction vector $\boldsymbol{d}$ by replacing $\boldsymbol{d}_{i,j}$ with $\frac{\boldsymbol{d}_{i,j}}{\|\boldsymbol{d}_{i,j}\|}\|\boldsymbol{\theta}_{i,j}\|$, where $\boldsymbol{d}_{i,j}$ and $\boldsymbol{\theta}_{i,j}$ represent the jth filter of the ith layer of the random direction $\boldsymbol{d}$ and the network parameters $\boldsymbol{\theta}$, respectively. Here, $\|\cdot\|$ denotes the Frobenius norm. It is crucial to note that j refers to the filter index. We use the function $L(\alpha) = R_S(\boldsymbol{\theta} + \alpha\boldsymbol{d})$ to characterize the loss landscape around the minima obtained with and without dropout layers.

For all network structures shown in Fig. 12, dropout improves the generalization of the network and finds flatter minima. In Fig. 12(a) and (b), for both networks trained with and without dropout layers, the training loss values are all close to zero, but their flatness and generalization are still different. In Fig. 12(c) and (d), due to the complexity of the dataset, i.e., CIFAR-100 and Multi30 k, and network structures, i.e., ResNet-20 and transformer, networks with dropout do not achieve zero training error but the ones with dropout find flatter minima with much better generalization. The accuracy of different network structures is shown in Table I.

B. The Effect of $R_1(\boldsymbol{\theta})$ on Flatness

In this subsection, we study the effect of $R_1(\boldsymbol{\theta})$ on flatness under the two-layer ReLU NN setting. Different from the flatness described above by loss interpolation, we define the flatness of the minimum as the sum of the eigenvalues of the Hessian matrix H in this section, i.e., $\mathrm{Tr}(H)$. Note that when $R_S(\boldsymbol{\theta}) = 0$, we

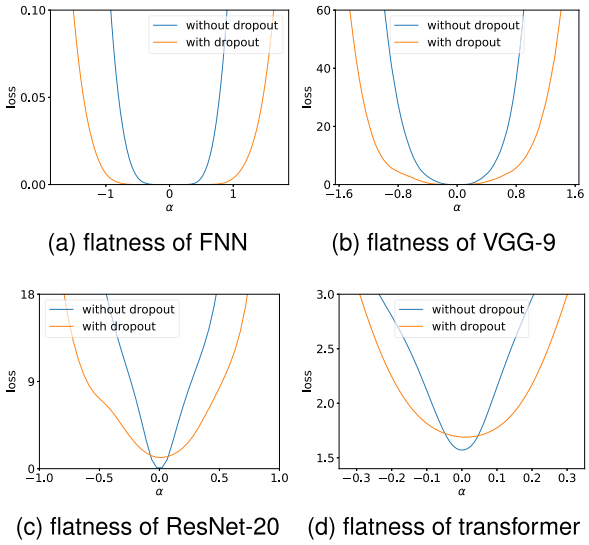


Fig. 12. 1D visualization of solutions of different network structures obtained with or without dropout layers. (a) The FNN is trained on MNIST dataset. The test accuracy for the model with dropout layers is 98.7% while 98.1% for the model without dropout layers. (b) The VGG-9 network is trained on CIFAR-10 dataset using the first 2048 examples as the training dataset. The test accuracy for the model with dropout layers is 60.6% while 59.2% for the model without dropout layers. (c) The ResNet-20 network is trained on CIFAR-100 dataset using all examples as the training dataset. The test accuracy for the model with dropout layers is 54.7% while 34.1% for the model without dropout layers. (d) The transformer is trained on Multi30 k dataset using the first 2048 examples as the training dataset. The test accuracy for the model with dropout layers is 49.3% while 34.7% for the model without dropout layers.

TABLE I
THE EFFECT OF DROPOUT ON MODEL ACCURACY

Structure	Dataset	With Dropout	Without Dropout
FNN	MNIST	98.7%	98.1%
VGG-9	CIFAR-10	60.6%	59.2%
ResNet-20	CIFAR-100	54.7%	34.1%
Transformer	Multi30k	49.3%	34.7%

have,

$$\mathrm{Tr}(H) = \mathrm{Tr}\left(\frac{1}{n}\sum_{i=1}^{n}\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\nabla_{\boldsymbol{\theta}}^{\intercal} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) = \frac{1}{n}\sum_{i=1}^{n}\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2,$$

thus the definition of flatness above is equivalent to $\frac{1}{n}\sum_{i=1}^{n}\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\|_2^2$.

Theorem 2 (the effect of $R_1(\boldsymbol{\theta})$ on facilitating flatness): Consider a two-layer ReLU NN,

$$f_{\boldsymbol{\theta}}(x) = \sum_{j=1}^{m} a_j \sigma(\boldsymbol{w}_j \boldsymbol{x}),$$

trained with dataset $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ with MSE loss. Under the gradient flow training with the loss function $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$, if $\boldsymbol{\theta}_0$ satisfying $R_S(\boldsymbol{\theta}_0) = 0$ and $\nabla_{\boldsymbol{\theta}} R_1(\boldsymbol{\theta}_0) \neq 0$, we have

$$\frac{\mathrm{d}\left(\frac{1}{n}\sum_{i=1}^{n}\|\nabla_{\boldsymbol{\theta}} f_{\boldsymbol{\theta}_0}(\boldsymbol{x}_i)\|_2^2\right)}{\mathrm{dt}} < 0.$$

The regularization effect of $R_2(\boldsymbol{\theta})$ also has a positive effect on flatness by constraining the norm of the gradient. In the next subsection, we compare the effect of these two regularization terms on generalization and flatness.

C. Effect of Two Implicit Regularization Terms on Generalization and Flatness

Although the modified gradient flow is noise-free during training, the model trained with the modified gradient flow can also find a flat minimum that generalizes well, due to the effect of $R_1(\boldsymbol{\theta})$ and $R_2(\boldsymbol{\theta})$. However, the magnitude of their impact on flatness is not yet fully understood. In this subsection, we study the effect of each regularization term through training networks by the following four loss functions:

$$\begin{aligned}
L_1(\boldsymbol{\theta}) &:= R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta}) \\
&:= R_S(\boldsymbol{\theta}) + \frac{1-p}{2np}\sum_{i=1}^{n}\sum_{j=1}^{m_{L-1}}\|\boldsymbol{W}_j^{[L]} f_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i)\|^2, \\
L_2(\boldsymbol{\theta},\boldsymbol{\eta}) &:= R_S(\boldsymbol{\theta}) + \tilde{R}_2(\boldsymbol{\theta},\boldsymbol{\eta}) \\
&:= R_S(\boldsymbol{\theta}) + \frac{\varepsilon}{4}\left\|\nabla_{\boldsymbol{\theta}} R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})\right\|^2, \\
L_3(\boldsymbol{\theta},\boldsymbol{\eta}) &:= R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta}) - \tilde{R}_2(\boldsymbol{\theta},\boldsymbol{\eta}) \\
&:= R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta}) - \frac{\varepsilon}{4}\left\|\nabla_{\boldsymbol{\theta}} R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})\right\|^2, \\
L_4(\boldsymbol{\theta},\boldsymbol{\eta}) &:= R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta}) - R_1(\boldsymbol{\theta}) \\
&:= R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta}) - \frac{1-p}{2np}\sum_{i=1}^{n}\sum_{j=1}^{m_{L-1}}\|\boldsymbol{W}_j^{[L]} f_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i)\|^2,
\end{aligned} \tag{7}$$

where $\tilde{R}_2(\boldsymbol{\theta},\boldsymbol{\eta})$ is defined as $(\varepsilon/4)\|\nabla_{\boldsymbol{\theta}} R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})\|^2$ for convenience, and we have $\mathbb{E}_{\boldsymbol{\eta}}\tilde{R}_2(\boldsymbol{\theta},\boldsymbol{\eta}) = R_2(\boldsymbol{\theta})$. For each $L_i, i \in [4]$, we explicitly add or subtract the penalty term of either $R_1(\boldsymbol{\theta})$ or $\tilde{R}_2(\boldsymbol{\theta},\boldsymbol{\eta})$ to study their effect on dropout regularization. Therefore, $L_1(\boldsymbol{\theta})$ and $L_3(\boldsymbol{\theta},\boldsymbol{\eta})$ are used to study the effect of $R_1(\boldsymbol{\theta})$, while $L_2(\boldsymbol{\theta},\boldsymbol{\eta})$ and $L_4(\boldsymbol{\theta},\boldsymbol{\eta})$ are for $R_2(\boldsymbol{\theta})$.

We first study the effect of two regularization terms on the generalization of NNs. As shown in Fig. 13, we compare the test accuracy obtained by training with the above four distinct loss functions under different dropout rates and utilize the results of $R_S(\boldsymbol{\theta})$ and $R_S^{\mathrm{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})$ as reference benchmarks. Two different learning rates are considered, with the solid and dashed lines corresponding to $\varepsilon = 0.05$ and $\varepsilon = 0.005$, respectively. As shown in Fig. 13(a), both approaches show that the training with the $R_1(\boldsymbol{\theta})$ regularization term finds a solution that almost has the same test accuracy as the training with dropout. For $\tilde{R}_2(\boldsymbol{\theta},\boldsymbol{\eta})$, as shown in Fig. 13(b), the effect of $\tilde{R}_2(\boldsymbol{\theta},\boldsymbol{\eta})$ only marginally improves the generalization ability of full-batch gradient descent training in comparison to the utilization of $R_1(\boldsymbol{\theta})$.

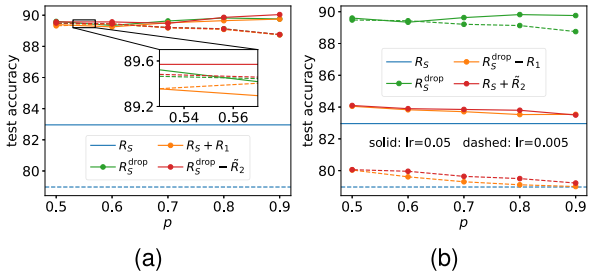


Fig. 13. Classification task on the MNIST dataset (the first 1000 images) using the FNN with size 784-1000-10. The test accuracy obtained by training with $L_1(\boldsymbol{\theta}), \ldots, L_4(\boldsymbol{\theta})$ and $R_S(\boldsymbol{\theta})$, $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ under different dropout rates and learning rates. The solid line represents the test accuracy of the network under a large learning rate ($\varepsilon = 0.05$), and the dashed line represents the test accuracy of the network under a small learning rate ($\varepsilon = 0.005$).

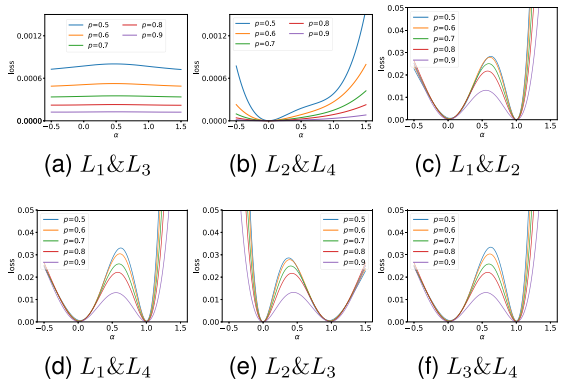


Fig. 14. Classification task on the MNIST dataset (the first 1,000 images) using the FNN with size 784-1000-10. The $R_S(\boldsymbol{\theta})$ value for interpolation between models with α interpolation factor. For $L_i\&L_j$, there is one trained model at $\alpha = 0$ (trained by loss function L_i), and the other is at $\alpha = 1$ (trained by loss function L_j). Different curves represent different dropout rates used for training.

Then we study the effect of two regularization terms on flatness. To this end, we show a one-dimensional cross-section of the loss $R_S(\boldsymbol{\theta})$ by the interpolation between two minima found by the training of two different loss functions. For either $R_1(\boldsymbol{\theta})$ or $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$, we use addition or subtraction to study its effect. As shown in Fig. 14(a), for $R_1(\boldsymbol{\theta})$, the loss value of the interpolation between the minima found by the addition approach (L_1) and the subtraction approach (L_3) stays near zero, which is similar for $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ in Fig. 14(b), showing that the higher-order terms of the learning rate ε in the modified equation have less influence on the training process. We then compare the flatness of minima found by the training with $R_1(\boldsymbol{\theta})$ and $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ as illustrated in Fig. 14(c)–(f). The results indicate that the minima obtained by the training with $R_1(\boldsymbol{\theta})$ exhibit greater flatness than those obtained by training with $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$.

The experiments in this section show that, compared with SGD, the unique implicit regularization of dropout, $R_1(\boldsymbol{\theta})$, plays a significant role in improving the generalization and finding flat minima.

VIII. Conclusion and Discussion

In this work, we theoretically study the implicit regularization of dropout and its role in improving the generalization performance of neural networks. Specifically, we derive two implicit regularization terms, $R_1(\boldsymbol{\theta})$ and $R_2(\boldsymbol{\theta})$, and validate their efficacy through numerical experiments. One important finding of this work is that the unique implicit regularization term $R_1(\boldsymbol{\theta})$ in dropout, unlike SGD, is a key factor in improving the generalization and flatness of the dropout solution. We also found that $R_1(\boldsymbol{\theta})$ can facilitate the weight condensation during training, which may establish a link among weight condensation, flatness, and generalization for further study. This work reveals rich and unique properties of dropout, which are fundamental to a comprehensive understanding of dropout.

Our study also sheds light on the broader issue of simplicity bias in deep learning. We observed that dropout regularization tends to impose a bias toward simple solutions during training, as evidenced by the weight condensation and flatness effects. This is consistent with other perspectives on simplicity bias in deep learning, such as the frequency principle [44], [45], [46], [47], [48], [49], [50], [51], which reveals that neural networks often learn data from low to high frequency. Our analysis of dropout regularization provides a detailed understanding of how simplicity bias works in practice, which is essential for understanding why over-parameterized neural networks can fit the training data well and generalize effectively to new data. So far, we do not have any concrete evidence to establish a connection between these two implicit biases. However, we believe that studying their connection would be an interesting topic. We have some clues to suggest their connection may not be simple. For example, it has been observed that SGD can find a flat solution [13]. However, as shown in Fig. 18 in the appendix, available online, this solution does not appear to be a condensed one.

Finally, our work highlights the potential benefits of dropout regularization in training neural networks, particularly in the linear regime. As we have shown, dropout regularization can induce weight condensation and avoid the slow training speed often encountered in highly nonlinear networks due to the fact that the training trajectory is close to the stationary point [36], [37]. This may have important implications for the development of more efficient and effective deep learning algorithms.

References

[1] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*.
[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[3] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 6105–6114.
[4] D. P. Helmbold and P. M. Long, "On the inductive bias of dropout," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 3403–3454, 2015.
[5] W. Mou, Y. Zhou, J. Gao, and L. Wang, "Dropout training, data-dependent regularization, and generalization bounds," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 3645–3653.
[6] R. Arora, P. Bartlett, P. Mianjy, and N. Srebro, "Dropout: Explicit forms and capacity control," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2021, pp. 351–361.
[7] P. Mianjy, R. Arora, and R. Vidal, "On the implicit bias of dropout," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 3540–3548.
[8] E. Hairer, C. Lubich, and G. Wanner, "Geometric numerical integration illustrated by the Störmer–Verlet method," *Acta Numerica*, vol. 12, pp. 399–450, 2003.

[9] T. Luo, Z.-Q. J. Xu, Z. Ma, and Y. Zhang, "Phase diagram for two-layer ReLU neural networks at infinite-width limit," *J. Mach. Learn. Res.*, vol. 22, no. 71, pp. 1–47, 2021.
[10] H. Zhou, Q. Zhou, T. Luo, Y. Zhang, and Z.-Q. J. Xu, "Towards understanding the condensation of neural networks at initial training," 2021, *arXiv:2105.11686*.
[11] H. Zhou, Q. Zhou, Z. Jin, T. Luo, Y. Zhang, and Z.-Q. J. Xu, "Empirical phase diagram for three-layer neural networks with infinite width," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 26021–26033.
[12] A. Jacot, C. Hongler, and F. Gabriel, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8580–8589.
[13] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," 2016, *arXiv:1609.04836*.
[14] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," 2017, *arXiv:1706.08947*.
[15] Z. Zhu, J. Wu, B. Yu, L. Wu, and J. Ma, "The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects," 2018, *arXiv:1803.00195*.
[16] D. McAllester, "A PAC-Bayesian tutorial with a dropout bound," 2013, *arXiv:1307.2118*.
[17] L. Wan, M. Zeiler, S. Zhang, Y. Lecun, and R. Fergus, "Regularization of neural networks using DropConnect," in *Proc. Int. Conf. Mach. Learn.*, Citeseer, 2013, pp. 1058–1066.
[18] K. Zhai and H. Wang, "Adaptive dropout with rademacher complexity regularization," in *Proc. Int. Conf. Learn. Representations*, 2018.
[19] P. Mianjy and R. Arora, "On convergence and generalization of dropout training," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 21151–21161.
[20] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2814–2822.
[21] S. Wager, S. Wang, and P. S. Liang, "Dropout training as adaptive regularization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 351–359.
[22] S. Wager, W. Fithian, S. Wang, and P. S. Liang, "Altitude training: Strong bounds for single-layer dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 100–108.
[23] A. Senen-Cerda and J. Sanders, "Almost sure convergence of dropout algorithms for neural networks," 2020, *arXiv:2002.02247*.
[24] Z. Zhang, Y. Li, T. Luo, and Z.-Q. J. Xu, "Stochastic modified equations and dynamics of dropout algorithm," 2023, *arXiv:2305.15850*.
[25] D. Bank and R. Giryes, "An ETF view of dropout regularization," in *Proc. Brit. Mach. Vis. Conf.*, 2020.
[26] B. J. Lengerich, E. Xing, and R. Caruana, "Dropout as a regularizer of interaction effects," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2022, pp. 7550–7564.
[27] J. Cavazza, P. Morerio, B. Haeffele, C. Lane, V. Murino, and R. Vidal, "Dropout as a low-rank regularizer for matrix factorization," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2018, pp. 435–444.
[28] A. Pal, C. Lane, R. Vidal, and B. D. Haeffele, "On the regularization properties of structured dropout," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7671–7679.
[29] S. I. Mirzadeh, M. Farajtabar, and H. Ghasemzadeh, "Dropout as an implicit gating mechanism for continual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 232–233.
[30] J. Blanchet, Y. Kang, J. L. M. Olea, V. A. Nguyen, and X. Zhang, "Dropout training is distributionally robust optimal," *J. Mach. Learn. Res.*, vol. 24, no. 180, pp. 1–60, 2023.
[31] C. Wei, S. Kakade, and T. Ma, "The implicit and explicit regularization effects of dropout," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 10181–10192.
[32] D. Barrett and B. Dherin, "Implicit gradient regularization," in *Proc. Int. Conf. Learn. Representations*, 2020.
[33] S. L. Smith, B. Dherin, D. Barrett, and S. De, "On the origin of implicit regularization in stochastic gradient descent," in *Proc. Int. Conf. Learn. Representations*, 2020.
[34] L. Wu, C. Ma, and W. E, "How SGD selects the global minima in over-parameterized learning: A dynamical stability perspective," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 8289–8298.
[35] S. Yaida, "Fluctuation-dissipation relations for stochastic gradient descent," in *Proc. Int. Conf. Learn. Representations*, 2018.
[36] Y. Zhang, Z. Zhang, T. Luo, and Z. J. Xu, "Embedding principle of loss landscape of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 14848–14859.
[37] Y. Zhang, Y. Li, Z. Zhang, T. Luo, and Z.-Q. J. Xu, "Embedding principle: A hierarchical structure of loss landscape of deep neural networks," *J. Mach. Learn.*, vol. 1, pp. 1–45, 2022.
[38] H. Maennel, O. Bousquet, and S. Gelly, "Gradient descent quantizes ReLU network features," 2018, *arXiv:1803.08367*.
[39] F. Pellegrini and G. Biroli, "An analytic theory of shallow networks dynamics for hinge loss classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 5356–5367.
[40] Z. Bai, T. Luo, Z.-Q. J. Xu, and Y. Zhang, "Embedding principle in depth for the loss landscape analysis of deep neural networks," 2022, *arXiv:2205.13283*.
[41] Y. Zhang, Z. Zhang, L. Zhang, Z. Bai, T. Luo, and Z.-Q. J. Xu, "Linear stability hypothesis and rank stratification for nonlinear models," 2022, *arXiv:2211.11623*.
[42] G. Blanc, N. Gupta, G. Valiant, and P. Valiant, "Implicit regularization for deep neural networks driven by an Ornstein-Uhlenbeck like process," in *Proc. Conf. Learn. Theory*, PMLR, 2020, pp. 483–513.
[43] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," 2017, *arXiv:1712.09913*.
[44] Z.-Q. J. Xu, Y. Zhang, and Y. Xiao, "Training behavior of deep neural network in frequency domain," in *Proc. Int. Conf. Neural Inf. Process.*, Springer, 2019, pp. 264–274.
[45] N. Rahaman et al., "On the spectral bias of deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 5301–5310.
[46] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, "Frequency principle: Fourier analysis sheds light on deep neural networks," *Commun. Comput. Phys.*, vol. 28, no. 5, pp. 1746–1767, 2020.
[47] B. Ronen, D. Jacobs, Y. Kasten, and S. Kritchman, "The convergence rate of neural networks for learned functions of different frequencies," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 4761–4771.
[48] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, and Q. Gu, "Towards understanding the spectral bias of deep learning," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021, pp. 2205–2211.
[49] Y. Zhang, T. Luo, Z. Ma, and Z.-Q. J. Xu, "A linear frequency principle model to understand the absence of overfitting in neural networks," *Chin. Phys. Lett.*, vol. 38, no. 3, 2021, Art. no. 038701.
[50] T. Luo, Z. Ma, Z.-Q. J. Xu, and Y. Zhang, "Theory of the frequency principle for general deep neural networks," *CSIAM Trans. Appl. Math.*, vol. 2, no. 3, pp. 484–507, 2021.
[51] Z.-Q. J. Xu, Y. Zhang, and T. Luo, "Overview frequency principle/spectral bias in deep learning," 2022, *arXiv:2201.07395*.
[52] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[55] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

Zhongwang Zhang received the bachelor's degree from the Zhiyuan College of Shanghai Jiao Tong University, in 2021. He is currently working toward the doctoral degree with the School of Mathematical Sciences, Shanghai Jiao Tong University. His interests include understanding deep learning from the training process, loss landscape, generalization, and applications.

Zhi-Qin John Xu received the graduated degree from the Zhiyuan College of Shanghai Jiao Tong University, in 2012, and the doctor's degree in applied mathematics from Shanghai Jiao Tong University, in 2016. He is an Associate Professor with the Institute of Natural Sciences/School of Mathematical Sciences, Shanghai Jiao Tong University. From 2016 to 2019, he was a postdoctoral fellow with NYU ABU Dhabi and the Courant Institute. He and collaborators discovered the frequency principle, parameter condensation, and embedding principles in deep learning, and developed multi-scale neural networks. His interests include understanding deep learning from the training process, loss landscape, generalization, and applications.