理解深度学习

2023 年 11 月 13 日

目录

1 Dropout的隐式正则化 **2**

1.1 Dropout 2
1.2 Dropout促进神经元凝聚 3
1.3 Dropout及其隐式正则化的显示表达 6
1.4 直观理解隐式正则项对凝聚的影响 8
1.5 非常强的凝聚正则化例子 9
1.6 Dropout促进凝聚有利于模型泛化性能 10
1.7 Dropout对解的平坦性的影响 11

Chapter 1

Dropout的隐式正则化

凝聚现象是神经网络非线性训练过程中非常重要的一个现象，也体现了训练过程的一种隐式正则化，即，逐步增加神经网络的有效神经元数目。在前面的介绍中，我们介绍了在梯度流下，凝聚现象的发生需要比较小的初始化。在介绍嵌入原则的部分，我们更仔细观察了训练过程，当发生凝聚的时候，训练轨迹会经历距离鞍点非常近的区域，使得训练变得非常慢，在损失函数关于训练步数的曲线中，观察到损失函数会停留在一个值附近很久。

因此，这里有一个平衡需要选择，小初始化会使得凝聚更明显，但同时会使训练变慢，而大初始化，会让神经网络趋于随机特征模型，但训练会快一些。事实上，一般的训练并不会完全用梯度下降的方法，而是会加一些正则化方法。在这一章节中，我们将介绍Dropout的训练技术，它可以使任意初始化下的神经网络在整个训练过程趋于凝聚。

Dropout的基本想法是在每一个训练步中，随机地丢弃部分神经元。比如，在某一步训练中，某一层中一半的神经元被丢弃了，那这一层的输出强度肯定受到很大的影响，于是剩下的神经元的输出乘了2以弥补丢弃带来的损失。然后再进行梯度下降训练。如果训练到最后，神经网络收敛到一个稳定的解，那最佳的方案就是那些丢弃的神经元和留下的神经元尽量保持一致，这样弥补的效果是最好的。这种神经元一致的情况就是我们谈到的凝聚现象。

下面，我们将首先从实验论证Dropout具有凝聚的隐式正则化，然后在理论上论证(Zhang & Xu 2022)。

1.1 Dropout

Dropout是一种用于训练神经网络的常见技巧Srivastava et al. (2014)，它可以提高模型的泛化能力Tan & Le (2019)。Dropout 的工作机制是将每个神经元的输出乘一个以概率 p 为 $1/p$ 或者概率 $1-p$ 为零的随机变量，且在每次前馈操作中，都对该随机变量进行采样。

对于神经网络的第l层输出$\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x}) \in \mathbb{R}^{m_l}$,我们采样一个独立随机向量$\boldsymbol{\eta} \in \mathbb{R}^{m_l}$,

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

其中$k \in [m_l]$是$\boldsymbol{\eta}$的一个索引。注意η是一个零均值随机变量。我们然后通过计算

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

来对第l层应用dropout，同时用$\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{[l]}(\boldsymbol{x})$代替$\boldsymbol{f}_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{x})$。这里我们用$\odot$表示两个同维矩阵的哈达马乘积。为了表达的简便性,我们令$\boldsymbol{\eta}$表示所有层上的这样的向量集合。$\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x})$表示在输入$\boldsymbol{x}$和dropout噪声$\boldsymbol{\eta}$下模型$\boldsymbol{f}_{\boldsymbol{\theta}}(\boldsymbol{x})$的输出。$R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$表示具有dropout层$f_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\text{drop}}$的网络的损失函数,即

$$R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) = \frac{1}{n} \sum_{i=1}^{n} \ell \left(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}_i), \boldsymbol{y}(\boldsymbol{x}_i) \right) = \mathbb{E}_S \ell \left(\boldsymbol{f}_{\boldsymbol{\theta},\boldsymbol{\eta}}^{\text{drop}}(\boldsymbol{x}), \boldsymbol{y} \right).$$

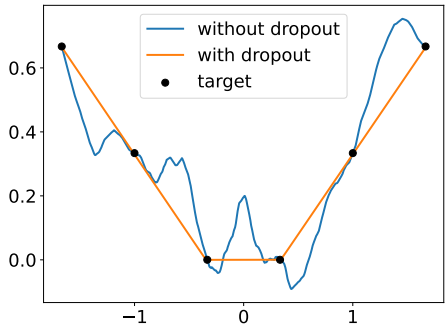


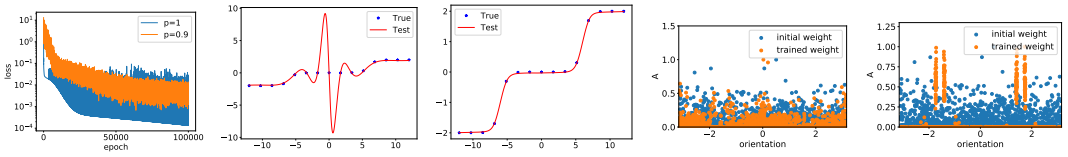
图 1.1: 训练有无dropout的两层ReLU神经网络的实验结果。隐藏层的宽度为1000，所有实验的学习率均为1×10^{-3}。有dropout及没有dropout的神经网络的输出，其中黑点表示目标点。

首先，我们对一个线性初始化下Jacot et al. (2018)的神经网络进行训练，来看一下有或者没有Dropout的差异。如图1.1所示，在没有Dropout的情况下，神经网络用一个非常振荡地曲线拟合了给定的几个离散点。这符合我们在相图分析中看到的结果，也符合传统的学习理论，也就是，模型过于复杂，会带来过拟合。但当我们加入Dropout时，模型就显示出非线性的特征，可以看到，此时神经网络用一个非常有特征的曲线拟合了给定的点，这种非常明显的转折暗示了参数内部有一定的结构。

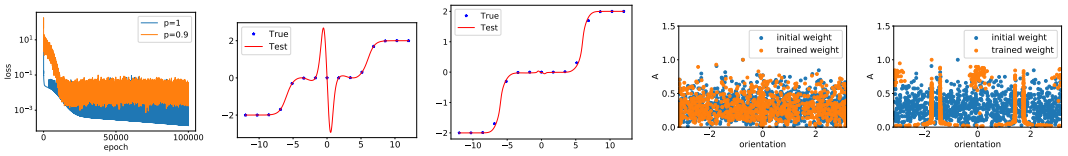
1.2 Dropout促进神经元凝聚

在这一节中，我们在不同的设定下研究dropout对神经元凝聚的促进作用，并对这种产生

凝聚的机制做出解释。首先，我们着眼于一维函数的拟合问题。

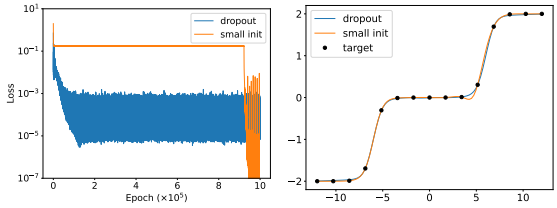


(a) 损失值 (b) $p = 1$, 输出 (c) $p = 0.9$, 输出 (d) $p = 1$, 参数特征 (e) $p = 0.9$, 参数特征



(f) 损失值 (g) $p = 1$, 输出 (h) $p = 0.9$, 输出 (i) $p = 1$, 参数特征 (j) $p = 0.9$, 参数特征

图 1.2: tanh 神经网络在不同丢弃频率下的输出和参数特征。隐藏层的宽度为1000，不同实验的学习率为1×10^{-3}。 (d,e,i,j)中，蓝点和橙点分别表示初始和最终训练阶段的权重特征分布。上面一行是两层网络的结果，dropout 层在隐藏层之后。下面一行是三层网络的结果，在两个隐藏层之间和最后一个隐藏层之后有 dropout 层。



(a) loss (b) output

图 1.3: 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. 1.2.

如图1.2所示，我们取目标函数

$$f(x) = \sigma(x - 6) + \sigma(x + 6),$$

其中$\sigma(x) = \tanh(x)$。我们由上述一维函数生成的数据训练具有 1000 个隐藏神经元的 tanh 神经网络，使用 MSE进行拟合。在相同初始化下使用和不使用 dropout 进行的实验都可以很好地拟合训练数据。为了清楚地研究 dropout 对凝聚现象的影响，我们采用线性区域 Luo et

al. (2021) 的参数初始化分布，它在没有附加约束的情况下不会发生凝聚现象。 dropout层用在两层网络的隐藏层之后（图1.2顶行），在三层网络的两个隐藏层之间和最后一个隐藏层之后（图1.2底行）使用。首先，图1.2(a,e)明显看到，loss下降过程没有经历明显的平台阶段，即参数没有被鞍点吸引，这使得使用dropout网络的训练速度要比小初始化网络的训练速度更快。通过观察拟合结果，我们发现图1.2(b,g)中没有 dropout 训练的神经网络的输出比图1.2(c,h)中使用 dropout 训练的神经网络的输出有更多的振荡。为了更好地理解 dropout 的对凝聚现象影响，我们进一步研究了参数的分布特征。

为了深入探究它们的参数空间表示的细节，我们注意到对于ReLU激活函数，每个神经元的参数对$(a_k, \boldsymbol{w}_k)$可以分离为一个单位方向特征$\hat{\boldsymbol{w}} = \boldsymbol{w}/|\boldsymbol{w}|_2$和一个表示其对输出贡献的振幅$A = |a||\boldsymbol{w}|_2$,即$(A, \hat{\boldsymbol{w}})$。对于一维输入，由于加入了偏置项，$\boldsymbol{w}$是二维的。因此，我们使用每个$\hat{\boldsymbol{w}}$相对于x轴的$[-\pi, \pi)$内的角度来表示其方向。因此$(A_k, \hat{\boldsymbol{w}}_k)_{k=1}^m$的散点图可以表示参数分布。受ReLU激活函数的启发，对于tanh激活函数，我们也使用此方法对参数分布进行表达。

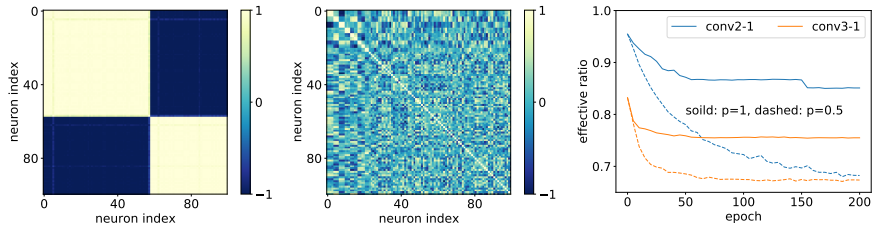
神经网络$\{(\hat{\boldsymbol{w}}_j, A_j)\}_{j=1}^m$的散点图如图 1.2(d,e,i,j)。为了方便起见，我们对每个模型参数的特征分布进行归一化，使得每个模型中神经元的最大幅度为1。与初始权重分布（蓝色）相比，没有使用 dropout 训练的网络权重（橙色）与其初始值接近。而对于使用dropout训练的神经网络，训练后的参数与初始化有显着差异，非零参数倾向于聚集在几个离散方向上，呈现凝聚趋势。

Dropout offers an ideal training method to induce condensation phenomena without suffering from the long training process. For example in Fig. 1.3(a), with the same setup as Fig. 1.2, 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. 1.3(b). In contrast, as illustrated in Fig. 1.2, a model with such a large initialization without dropout produces an oscillating output function with no condensation.

接下来，我们研究高维数据设定下，dropout对凝聚现象的影响。我们进一步研究了teacher-student设定下 dropout 对高维两层 tanh 神经网络的影响。具体来说，我们利用只有一个隐藏层神经元和 10 维输入的两层 tanh 神经网络作为目标函数。对于神经网络模型，两个神经元之间方向的相似度是通过其归一化权重的内积，即余弦相似度来计算的。如图 1.4(a,b) 所示，对于有 dropout 的神经网络，网络的神经元只有两个方向，表明发生了神经元凝聚现象，而没有 dropout 的神经网络则没有表现出类似的现象。

对于复杂的高维输入问题，我们该如何研究凝聚程度随训练进行的变化情况？为了可视化训练过程中的凝聚，我们定义神经元的有效比率如下。

定义 1 (**神经元有效比率**). *对于给定的神经网络，第 l 层中神经元 j 的输入权重向量化为*



(a) $p = 0.5$, 参数特征　　(b) $p = 1$, 参数特征　　(c) 神经元有效比率

图 1.4: 具有不同丢失概率的高维神经网络的稀疏性。 (a,b) 有和没有 dropout 的两层 tanh 神经网络的参数特征。 (c) 使用 ResNet-18 的 CIFAR-10 分类任务下，有和没有 dropout 的神经元有效比率。其中Conv2-1和conv3-1分别表示ResNet的第二块和第三块的第一卷积层的参数。

$\boldsymbol{\theta}_j^{[l]} \in \mathbb{R}^{m_l}$。设 $U^{[l]} = \{\boldsymbol{u}_k^{[l]}\}_{k=1}^{m_l'}$ 为单位向量集合，使得对于任何 $\boldsymbol{\theta}_j^{[l]}$，存在元素$\boldsymbol{u} \in U^{[l]}$满足$\boldsymbol{u} \cdot \boldsymbol{\theta}_j^{[l]}/||\boldsymbol{\theta}_j^{[l]}|| > 0.95$。第$l$层的有效神经元数量$m_l^{\text{eff}}$被定义为所有可能的$U^{[l]}$中元素个数最少的那个集合的元素个数。有效比率定义为m_l^{eff}/m_l。

我们研究使用ResNet-18学习CIFAR-10的训练过程。如图 1.4(c) 所示，具有 dropout 的神经网络往往具有较低的有效比率，从而往往表现得更凝聚。

具体的，我们使用带有和不带有 dropout 层的 ResNet-18 对 CIFAR-10 进行分类。对于使用dropout的网络，我们在每个块(block)的最后一个激活函数后添加 dropout 层，取$p = 0.5$。我们使用 Adam 训练网络，学习率为 0.001，批量大小为 128。对于网络参数初始化分布，我们均使用kaiming初始化。

1.3 Dropout及其隐式正则化的显示表达

在本节中，我们基于dropout机制所诱发的噪声结构，给出其隐式正则化的显式表达，即，dropout的隐式正则化会给每个神经元的输出带来一个L_2的限制。具体推导过程详见Zhang & Xu (2022)。

为了便于叙述与计算，我们考虑一个L层($L \geq 2$)全连接神经网络,其中dropout层仅在网络的第$L-1$层后面添加。同时我们以均方差损失(MSE)作为我们的损失函数。基于上述设定，我们可以得到损失函数($R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$)关于随机变量$\boldsymbol{\eta}$的期望

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

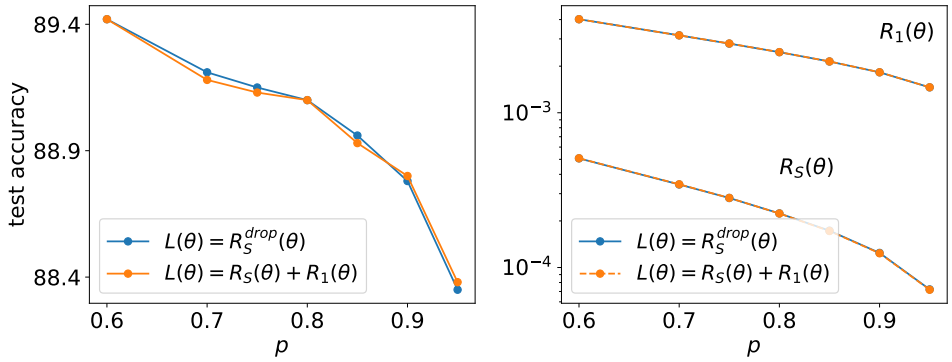
其中，$R_1(\boldsymbol{\theta})$是一个由dropout噪声所诱导的额外正则项，其形式为

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

其中 $\boldsymbol{W}_j^{[L]} \in \mathbb{R}^{m_L}$ 是 $\boldsymbol{W}^{[L]}$ 的第 j 列，$f_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i)$ 是 $\boldsymbol{f}_{\boldsymbol{\theta}}^{[L-1]}(\boldsymbol{x}_i)$ 中的第 j 个元素。

不同于SGD等随机算法的隐式正则化项，dropout所诱导的$R_1(\boldsymbol{\theta})$项是一个与学习率无关的项，也就是说，即使在梯度流(学习率足够小)的设定下，模型训练依旧会受到该正则项的限制。$R_1(\boldsymbol{\theta})$项本质上是在限制网络最外层的每个神经元输出二范数的平方和的大小，在下一节中，我们将展示这种学习率无关的L_2限制，对神经元凝聚及提高模型损失景观平坦性的效用、下面，我们先通过实验来验证$R_1(\boldsymbol{\theta})$项的准确性。

由于$R_1(\boldsymbol{\theta})$与学习率无关，因此我们选择一个较小的学习率通过近似梯度流来验证$R_1(\boldsymbol{\theta})$的有效性。我们通过由$R_S^{\text{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})$ and $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$两个损失函数训练的神经网络的相似度来验证$R_1(\boldsymbol{\theta})$的有效性。



(a) (b)

图 1.5: 宽度为 1000 的两层神经网络，用于对 MNIST 数据集的前 1000 张图像进行分类。我们使用两种不同的损失函数：$R_S^{\text{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})$ 和 $R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$. 为了研究不同的丢弃概率对网络性能的影响，我们在保持 $\varepsilon = 5 \times 10^{-3}$ 的恒定学习率的同时，使用不同的丢弃概率进行实验。(a)模型的测试精度。(b) $R_S^{\text{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})$ 和 $R_1(\boldsymbol{\theta})$ 的值。

图 1.5(a) 展示了在不同丢弃概率下训练的两种损失的测试精度。其中对于使用$R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$训练的网络，网络中并没有dropout层，丢弃概率影响损失函数中的$R_1(\boldsymbol{\theta})$。对于不同的丢弃概率，上述两种loss得到的网络表现出相似的测试精度。值得一提的是，对于使用$R_S(\boldsymbol{\theta})$训练的网络，获得的准确率仅为79%，明显低于通过上面两个损失函数训练的网络的准确率（图1.5(a)中超过 88%）。在图 1.5(b) 中，我们展示了不同丢弃概率下通过不同损失函数训练的两个网络的$R_S(\boldsymbol{\theta})$和$R_1(\boldsymbol{\theta})$的值。其中，对于由$R_S^{\text{drop}}(\boldsymbol{\theta},\boldsymbol{\eta})$训练的网络，我们仍然可以通过网络

的参数来计算这两项。可以看出，对于不同的丢弃概率，两个网络中$R_S(\boldsymbol{\theta})$和$R_1(\boldsymbol{\theta})$的值几乎无法区分。

1.4 直观理解隐式正则项对凝聚的影响

从隐式正则化项 $R_1(\boldsymbol{\theta})$ 可以看出，dropout 正则化对每个神经元的输出施加了额外的 l_2-范数约束。该约束对凝聚现象有影响。我们通过一个两层 ReLU 网络的简单示例来说明 $R_1(\boldsymbol{\theta})$ 的效果。

我们使用以下两层 ReLU 网络来拟合一维函数：

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

其中 $\boldsymbol{x} := (x, 1)^\intercal \in \mathbb{R}^2$, $\boldsymbol{w}_j := (w_j, b_j) \in \mathbb{R}^2$, $\sigma(x) = \mathrm{ReLU}(x)$。为简单起见，我们设置$m = 2$，并假设网络可以完美拟合由目标函数$\sigma(\boldsymbol{w}^* \cdot \boldsymbol{x})$生成的两个数据点的训练数据集，表示为$\boldsymbol{o}^* := (\sigma(\boldsymbol{w}^* \cdot \boldsymbol{x}_1), \sigma(\boldsymbol{w}^* \cdot \boldsymbol{x}_2))$。我们进一步假设 $\boldsymbol{w}^* \cdot \boldsymbol{x}_i > 0, i = 1, 2$。将样本上第 j 个神经元的输出表示为

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

经过足够长的训练后，网络输出应等于训练数据点上的目标值，即

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

有无数对 $\boldsymbol{o}_1$ 和 $\boldsymbol{o}_2$ 可以很好地拟合 $\boldsymbol{o}^*$。然而，$R_1(\boldsymbol{\theta})$ 项将训练引导到特定的对。 $R_1(\boldsymbol{\theta})$ 可以写成

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

正如图1.6所示，$\boldsymbol{o}_j$ 垂直于 $\boldsymbol{o}^*$ 的分量需要在训练良好的阶段相互抵消，以最小化 $R_1(\boldsymbol{\theta})$。因此，$\boldsymbol{o}_1$和$\boldsymbol{o}_2$需要与$\boldsymbol{o}^*$平行，即$\boldsymbol{w}_1 // \boldsymbol{w}_2 // \boldsymbol{w}^*$，这就是凝聚现象。

图 1.6: $R_1(\boldsymbol{\theta})$项促进凝聚现象发生原因的直观理解示意图。

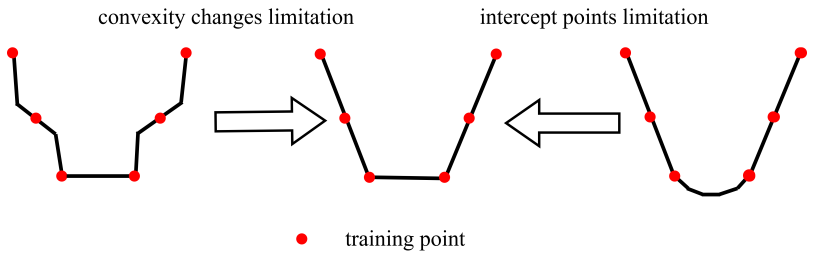


图 1.7: Schematic diagram of the effect of $R_1(\boldsymbol{\theta})$.

The example in the left curve in Fig. 1.7 shows the case (i) of Theorem **??**, that is, the convexity change can be reduced while the loss stays zero. The example in the right curve in Fig. 1.7 shows the case (ii) of Theorem **??**, 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 Fig. 1.1 and Fig. 1.2, NNs reach the condensed solution due to the constraint of the convexity changes and the intercept points (also illustrated in Fig. 1.7).

1.5 非常强的凝聚正则化例子

既然对于初始化在线性区域的网络，dropout对其有促进凝聚的作用。那如果对于已经近似拟合好的没有凝聚现象的模型，即陷入一个没有凝聚现象的极小值点附近，dropout算法会帮助模型跳出这个“坏的”极小点吗？为了回答这个问题，我们研究了使用损失函数$R_S(\boldsymbol{\theta})$训练的模型在两个损失函数$R_S^{\text{drop}}(\boldsymbol{\theta})$和$R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$下的稳定性。如图1.8左图所示，我们使用$R_S(\boldsymbol{\theta})$作为损失函数来训练虚线之前的模型。当$R_S(\boldsymbol{\theta})$的值很小时，我们用 $R_S^{\text{drop}}(\boldsymbol{\theta})$ 或 $R_S(\boldsymbol{\theta})+R_1(\boldsymbol{\theta})$ 替换损失函数。使用这三个损失函数训练的模型的输出和参数特征分别展示在图1.8的中间和右侧子图中。实验结果表明，dropout（$R_1(\boldsymbol{\theta})$ 项）有助于训练过程摆脱线性区域下 $R_S(\boldsymbol{\theta})$ 训练获得的最小值并找到有凝聚现象的模型。

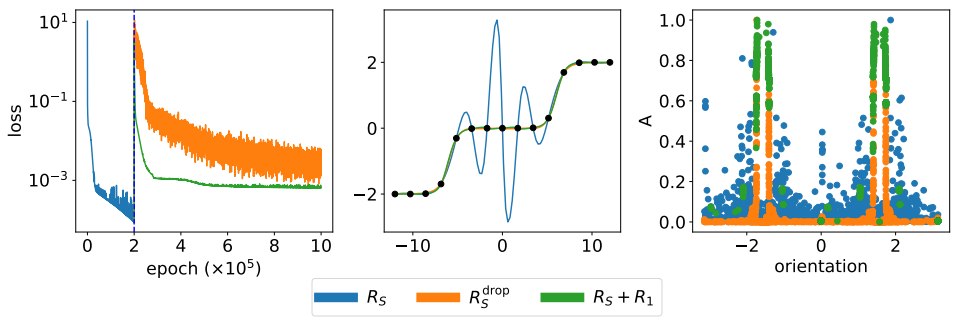


图 1.8: 两层tanh网络解在不同设定下的比较。蓝色：在线性区域初始化以$R_S(\boldsymbol{\theta})$为目标函数训练。橙色：将以$R_S(\boldsymbol{\theta})$为目标函数训练得到的解作为初始化参数（左图虚线位置），以$R_S^{\text{drop}}(\boldsymbol{\theta})$为损失函数进行训练。绿色：将以$R_S(\boldsymbol{\theta})$为目标函数训练得到的解作为初始化参数（左图虚线位置），以$R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$为损失函数进行训练左图：不同损失函数下的损失轨迹。中间图：$R_S(\boldsymbol{\theta})$训练的模型（蓝色）以及以$R_S(\boldsymbol{\theta})$ 得到模型作为初始化，以$R_S^{\text{drop}}(\boldsymbol{\theta})$（橙色）和$R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$（绿色）训练得到的模型的输出，其中黑点是目标点。右图：$R_S(\boldsymbol{\theta})$训练的模型（蓝色）以及以$R_S(\boldsymbol{\theta})$ 得到模型作为初始化，以$R_S^{\text{drop}}(\boldsymbol{\theta})$（橙色）和$R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta})$（绿色）训练得到的模型的参数特征分布。

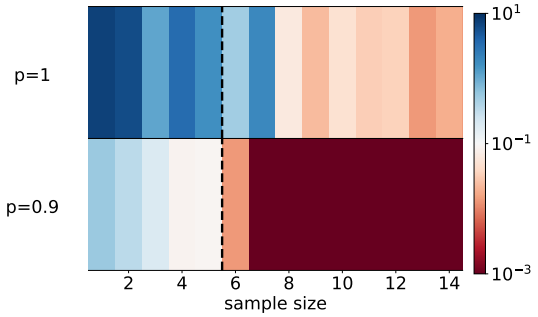


图 1.9: 不同丢弃概率（纵坐标）下两层 tanh 神经网络（颜色）的平均测试误差与样本数量（横坐标）的关系。对于所有实验，隐藏层的宽度为 1000，Adam 优化器的学习率为 1×10^{-4}。每个测试误差是随机初始化的 10 次独立试验的平均值。

1.6 Dropout促进凝聚有利于模型泛化性能

由于发生凝聚的网络的有效神经元数量远小于其实际神经元数量，因此往往具有更好的

泛化能力。为了验证这一点，我们取目标函数

$$f(x) = \sigma(x-6) + \sigma(x+6),$$

其中$\sigma(x) = \tanh(x)$。我们使用具有 1000 个神经元的两层 tanh 网络来学习这个具有两个神经元的两层 tanh 网络（教师-学生设定）。教师网络中的自由参数数量为6。如图1.9所示，当采样数大于6时，带有dropout的模型泛化能力很好，而没有dropout的模型泛化能力很差。

1.7 Dropout对解的平坦性的影响

了解 dropout 提高神经网络泛化能力的机制对我们理解神经网络是很重要的。许多实验发现最小值周围的平坦性与模型的泛化能力有很高的相关性。在本节中，我们研究了 Dropout 得到的最小值的平坦性。

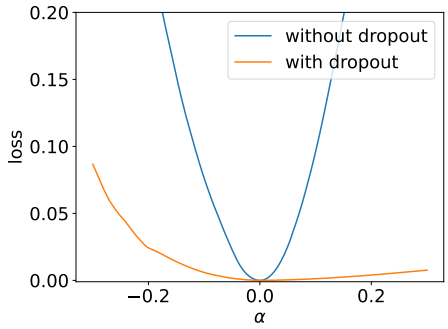


图 1.10: 训练有无dropout的两层ReLU神经网络的实验结果（与图1.1相同的实验）。隐藏层的宽度为1000，所有实验的学习率均为1×10^{-3}。有dropout及没有dropout的网络在给定随机方向上扰动网络得到的损失值。 α 是向上述方向移动的步长。

为了公平比较不同模型之间的平坦度，我们采用?中使用的方法，如下所示。为了获得具有参数 $\boldsymbol{\theta}$ 的网络方向，我们首先生成一个随机高斯方向向量 $\boldsymbol{d}$，其尺寸与 $\boldsymbol{\theta}$ 一致。然后，我们对 $\boldsymbol{d}$ 中的每个滤波器进行标准化，使其与 $\boldsymbol{\theta}$ 中相应的滤波器具有相同的范数。对于全连接网络来说，每一层都可以看作一个滤波器，归一化过程相当于对层进行归一化，而对于卷积神经网络来说，每个卷积核可能有多个滤波器，每个滤波器都单独进行归一化。因此，我们通过将 $\boldsymbol{d}_{i,j}$ 替换为 $\frac{\boldsymbol{d}_{i,j}}{\|\boldsymbol{d}_{i,j}\|}\|\boldsymbol{\theta}_{i,j}\|$ 来获得归一化方向向量 $\boldsymbol{d}$，其中$\boldsymbol{d}_{i,j}$和$\boldsymbol{\theta}_{i,j}$表示随机方向第$i$层的第$j$个滤波器分别为$\boldsymbol{d}$ 和网络参数$\boldsymbol{\theta}$。这里，$\|\cdot\|$表示Frobenius范数。需要注意的是，j 指的是过滤器索引。我们使用函数 $L(\alpha) = R_S(\boldsymbol{\theta} + \alpha\boldsymbol{d})$ 来表征使用和不使用 dropout 层时获得的最小值周围的损失景观。

表 1.1: Dropout对模型准确性的影响

网络结构	数据集	有dropout	没有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%

我们先在一维简单例子上验证dropout对平坦性的效果。我们使用宽度为 1000 的 ReLU FNN 来拟合目标函数，如下所示，

$$f(x) = \frac{1}{2}\sigma(-x - \frac{1}{3}) + \frac{1}{2}\sigma(x - \frac{1}{3}),$$

其中 $\sigma(x) = \mathrm{ReLU}(x)$. 我们使用 Adam 训练网络，学习率为 1×10^{-4}。我们令参数初始化在线性区域，$\boldsymbol{\theta} \sim N\left(0, \frac{1}{m^{0.2}}\right)$，其中$m = 1000$是隐藏层的宽度。

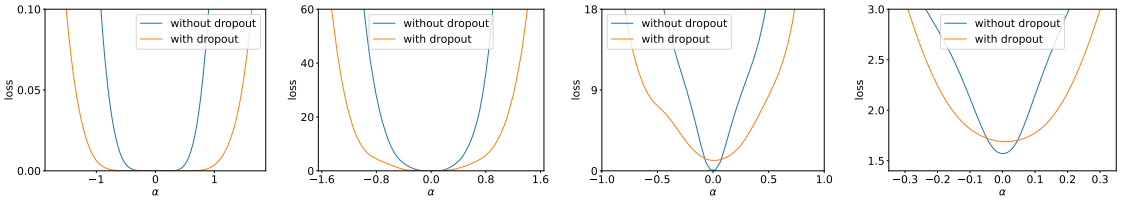
容易看到，对于使用dropout的网络，其最小值点附近的损失景观更加平坦。下面，我们着眼于更复杂的任务，并比较添加与不添加dropout层网络的平坦性及其泛化能力。

对于图 1.11 中所示的所有网络结构，dropout 提高了网络的泛化性并找到更平坦的最小值。在图 1.11(a, b) 中，对于有和没有 dropout 层训练的网络，训练损失值都接近于零，但它们的平坦度和泛化能力仍然不同。在图 1.11(c, d) 中，由于数据集（即 CIFAR-100 和 Multi30k）以及网络结构（即 ResNet-20 和 Transformer）的复杂性，具有 dropout 的网络难以实现零训练误差，但带有 dropout 层的网络依旧能找到更平坦的最小值，具有更好的泛化能力。不同网络结构的准确率如表1.1所示。

那么平坦性的提升是否由$R_1(\boldsymbol{\theta})$引起？下面我们细致研究了$R_1(\boldsymbol{\theta})$项对平坦性的贡献。为了更好的进行比较，我们有必要引入另一个dropout算法诱导的隐式正则化项$R_2(\boldsymbol{\theta})$，其中

$$R_2(\boldsymbol{\theta}) := \frac{\varepsilon}{4}\mathbb{E}_{\boldsymbol{\eta}}\|\nabla_{\boldsymbol{\theta}} R_S^{\mathrm{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})\|^2,$$

$R_2(\boldsymbol{\theta})$的产生是由于梯度流的离散化，因此它是一个与学习率有关的项。$R_2(\boldsymbol{\theta})$的影响在学习率趋于零时（梯度流下）就会消失。Dropout迭代过程主要受$R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta}) + R_2(\boldsymbol{\theta})$的影响，以及一些可忽略的关于学习率的高阶项。可以看到，$R_2(\boldsymbol{\theta})$也存在提高模型平坦性的正则化效果，因此我们主要通过控制变量比较二者对提升模型平坦性的贡献。我们通过训练网络通过以下四个损失函数来研究每个正则化项的效果：



(a) FNN的损失景观 (b) VGG-9的损失景观 (c) ResNet-20的损失景观 (d) transformer的损失景观

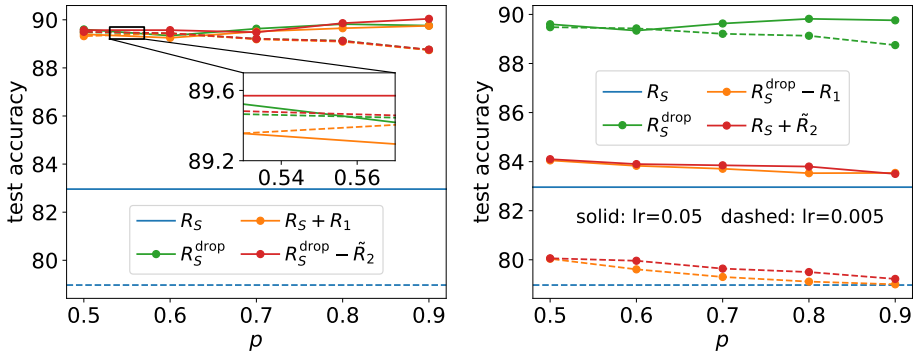
图 1.11: 使用或不使用 dropout 层获得的不同网络结构的解的损失景观一维可视化。(a) FNN 在 MNIST 数据集上进行训练。具有 dropout 层的模型的测试精度为 98.7%，而没有 dropout 层的模型的测试精度为 98.1%。(b) 使用前 2048 个示例作为训练数据集，在 CIFAR-10 数据集上训练 VGG-9 网络。具有 dropout 层的模型的测试精度为 60.6%，而没有 dropout 层的模型的测试精度为 59.2%。(c) 使用所有示例作为训练数据集，在 CIFAR-100 数据集上训练 ResNet-20 网络。具有 dropout 层的模型的测试准确度为 54.7%，而没有 dropout 层的模型的测试准确度为 34.1%。(d) 使用前 2048 个示例作为训练数据集，在 Multi30k 数据集上训练 Transformer。具有 dropout 层的模型的测试准确度为 49.3%，而没有 dropout 层的模型的测试准确度为 34.7%。

$$
\begin{aligned}
L_1(\boldsymbol{\theta}) &:= R_S(\boldsymbol{\theta}) + R_1(\boldsymbol{\theta}) \\
&:= R_S(\boldsymbol{\theta}) + \frac{1-p}{2np} \sum_{i=1}^{n} \sum_{j=1}^{m_{L-1}} \|\boldsymbol{W}_j^{[L]} f_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i)\|^2, \\
L_2(\boldsymbol{\theta}, \boldsymbol{\eta}) &:= R_S(\boldsymbol{\theta}) + \tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta}) \\
&:= R_S(\boldsymbol{\theta}) + \frac{\varepsilon}{4} \left\| \nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) \right\|^2, \\
L_3(\boldsymbol{\theta}, \boldsymbol{\eta}) &:= R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) - \tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta}) \\
&:= R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) - \frac{\varepsilon}{4} \left\| \nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) \right\|^2, \\
L_4(\boldsymbol{\theta}, \boldsymbol{\eta}) &:= R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) - R_1(\boldsymbol{\theta}) \\
&:= R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) - \frac{1-p}{2np} \sum_{i=1}^{n} \sum_{j=1}^{m_{L-1}} \|\boldsymbol{W}_j^{[L]} f_{\boldsymbol{\theta},j}^{[L-1]}(\boldsymbol{x}_i)\|^2,
\end{aligned}
\tag{1.1}
$$

其中，方便起见，$\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ 被定义为 $(\varepsilon/4) \left\| \nabla_{\boldsymbol{\theta}} R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta}) \right\|^2$，我们有 $\mathbb{E}_{\boldsymbol{\eta}} \tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta}) = R_2(\boldsymbol{\theta})$。对于每个 $L_i, i \in [4]$，我们显式地添加或减去 $R_1(\boldsymbol{\theta})$ 或 $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ 的惩罚项来研究它们的正则化效果。其中，$L_1(\boldsymbol{\theta})$ 和 $L_3(\boldsymbol{\theta}, \boldsymbol{\eta})$ 用于研究 $R_1(\boldsymbol{\theta})$ 的效果，而 $L_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ 和 $L_4(\boldsymbol{\theta}, \boldsymbol{\eta})$

适用于 $R_2(\boldsymbol{\theta})$。

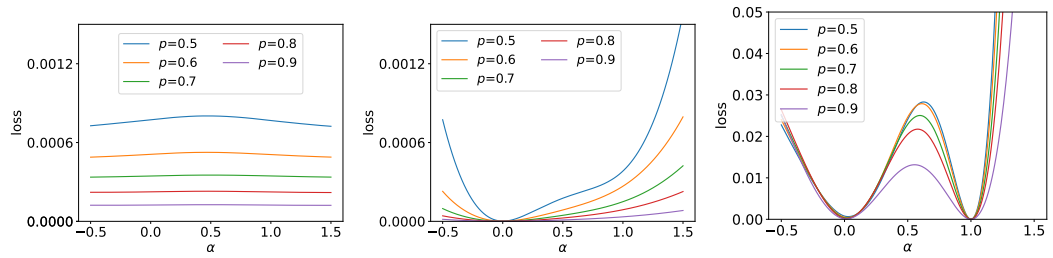
我们首先研究两个正则化项对神经网络泛化的影响。如图1.12所示，我们比较了在不同丢弃概率下用上述四种不同的损失函数训练得到的测试精度，并利用$R_S(\boldsymbol{\theta})$和 $R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$ 作为参考基准。考虑两种不同的学习率，实线和虚线分别对应于 $\varepsilon = 0.05$ 和 $\varepsilon = 0.005$。如图1.12(a) 所示，两种方法都表明使用 $R_1(\boldsymbol{\theta})$ 正则化项进行训练找到的解几乎与使用 dropout 进行训练具有相同的测试精度。对于$\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$，如图1.12(b)所示，$\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$的效果与使用 $R_1(\boldsymbol{\theta})$ 相比，仅略微提高了全批次梯度下降训练的泛化能力。



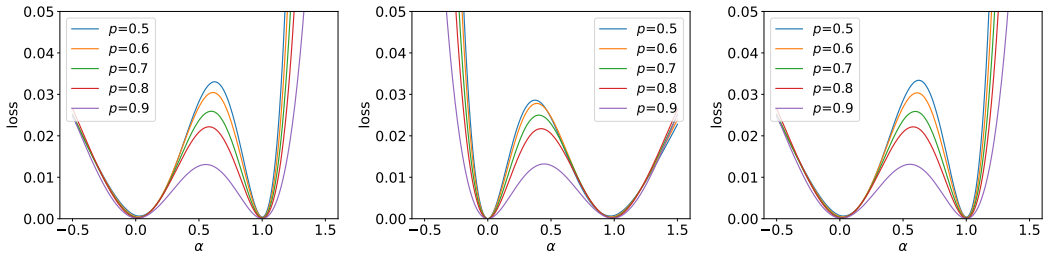
(a) (b)

图 1.12: 使用大小为 784-1000-10 的 FNN 对 MNIST 数据集（前 1000 张图像）进行分类任务。在不同的丢弃概率和学习率下, 通过 $L_1(\boldsymbol{\theta}), \cdots, L_4(\boldsymbol{\theta})$ 和 $R_S(\boldsymbol{\theta})$、$R_S^{\text{drop}}(\boldsymbol{\theta}, \boldsymbol{\eta})$进行训练。实线表示网络在大学习率（$\varepsilon = 0.05$）下的测试精度，虚线表示网络在小学习率（$\varepsilon = 0.005$）下的测试精度。

然后我们研究两个正则化项对平坦度的影响。为此，我们通过由两个不同损失函数找到的最小值之间的插值来显示损失 $R_S(\boldsymbol{\theta})$ 的一维横截面。对于 $R_1(\boldsymbol{\theta})$ 或 $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$，我们使用加法或减法来研究其效果。如图1.13(a)所示，对于$R_1(\boldsymbol{\theta})$，通过加法(L_1)和减法(L_3)，容易看出损失截面的损失值一直保持在零附近，这与图 1.13(b) 中的 $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ 类似，表明学习率的高阶项对训练过程的影响较小。然后，我们将由$R_1(\boldsymbol{\theta})$正则化限制训练得到的最小值的平坦度与$\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ 进行比较，如图 1.13(c-f)所示。结果表明，通过 $R_1(\boldsymbol{\theta})$ 训练获得的最小值比通过 $\tilde{R}_2(\boldsymbol{\theta}, \boldsymbol{\eta})$ 训练获得的最小值表现出更大的平坦度。



(a) L_1&L_3 (b) L_2&L_4 (c) L_1&L_2



(d) L_1&L_4 (e) L_2&L_3 (f) L_3&L_4

图 1.13: 以插值因子α 对两个模型$R_S(\boldsymbol{\theta})$进行插值。使用大小为 784-1000-10 的 FNN 对 MNIST 数据集（前 1000 张图像）进行分类任务。对于L_i&L_j，一个训练模型在$\alpha = 0$（由损失函数L_i训练），另一个在$\alpha = 1$（由损失函数L_j训练）。不同的曲线代表用于训练的不同丢弃概率。

参考文献

Jacot, A., Gabriel, F. & Hongler, C. (2018), Neural tangent kernel: Convergence and generalization in neural networks, *in* 'Advances in neural information processing systems', pp. 8571–8580.

Luo, T., Xu, Z.-Q. J., Ma, Z. & Zhang, Y. (2021), 'Phase diagram for two-layer relu neural networks at infinite-width limit', *Journal of Machine Learning Research* **22**(71), 1–47.

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

Tan, M. & Le, Q. (2019), Efficientnet: Rethinking model scaling for convolutional neural networks, *in* 'International conference on machine learning', PMLR, pp. 6105–6114.

Zhang, Z. & Xu, Z.-Q. J. (2022), 'Implicit regularization of dropout', *arXiv preprint arXiv:2207.05952* .