利用潜在二元变量和归一化流稀疏化贝叶斯神经网络

Lars Skaaret-Lund LARS.SKAARET-LUND@NMBU.NO
Geir Storvik GEIRS@MATH.UIO.NO
Aliaksandr Hubin ALIAKSANDR.HUBIN@NMBU.NO

arXiv:2305.03395v1 [stat.ML] 5 May 2023

摘要

人工神经网络（ANN）是功能强大的机器学习方法，广泛应用于人脸识别、机器翻译和癌症诊断等诸多现代领域。人工神经网络的一个常见问题是，它们通常具有数百万甚至数十亿个可训练参数，因此往往容易对训练数据产生过拟合。在需要获得可靠的不确定性估计的应用中，这一问题尤为突出。贝叶斯神经网络（BNN）能够改善这一问题，因为它们将参数不确定性纳入了考虑。此外，潜在二元贝叶斯神经网络（LBBNN）还通过允许权重开启或关闭，将结构不确定性纳入考虑，从而能够在权重与结构的联合空间中进行推断。本文将探讨对LBBNN方法的两项扩展：首先，通过使用局部重参数化技巧（LRT）直接对隐藏单元进行采样，我们获得了一种计算效率更高的算法。更重要的是，通过在LBBNN参数的变分后验分布上使用归一化流，网络能够学习到比均值场高斯分布更加灵活的变分后验分布。实验结果表明，与LBBNN方法相比，这种方法不仅提高了预测能力，还获得了更加稀疏的网络。我们进行了两项模拟研究。在第一项研究中，我们考虑了逻辑回归场景下的变量选择问题，其中更加灵活的变分分布带来了更好的结果。在第二项研究中，我们比较了基于二维高斯分布生成的数据所得到的预测不确定性。在这一研究中，我们论证了贝叶斯方法能够产生更加符合实际的预测不确定性估计。

1. 引言

使用数学模型来模拟大脑工作方式的想法，最早由 McCulloch 和 Pitts 提出 [1943]。然而，直到近些年，人们才通过反向传播的思想，真正发挥出这些模型的潜力：利用梯度下降训练模型 [Rumelhart 等人， 1986] 。随着现代 GPU 架构的出现，深度神经网络得以扩展至大数据规模，并已在多种任务中展现出卓越表现，包括计算机视觉 [Voulodimos 等人， 2018], 自然语言处理 [Young 等人， 2018] 以及强化学习 [Li， 2017]。现代深度学习架构可以包含数十亿个可训练参数 [Khan 等人， 2020]。由于模型中的参数数量庞大，网络具有过拟合的倾向，因此可能无法很好地泛化到未见过的数据。人们采用各种正则化方法来应对这一问题，例如提前停止 [Prechelt， 1998], Dropout [Srivastava 等人， 2014] 或数据增强 [Shorten 和 Khoshgoftaar， 2019]。这些技术属于启发式方法，因此其使用方式及效果并不总是明晰，尤其是在

实践中。还可以通过剪枝来减少网络中的参数数量。通常采用稠密到稀疏方法 [Han 等人, 2017]。具体而言，先训练一个稠密网络，同时记录权重的重要性（即其大小）。随后，移除低于稀疏度阈值（一个超参数）的权重。在 Frankle 和 Carbin [2018], 提出的假设中，随机初始化的稠密网络中存在一个稀疏子网络（中奖彩票），它可以单独训练，并达到与原始稠密网络相同的测试准确率。不同于只训练和剪枝一次（即单次剪枝），这一过程会依次重复多次，每次移除一定比例的剩余权重，从而得到比单次剪枝所得网络稀疏度更高的网络。然而，这会带来更高的计算成本。Evci 等人对此作了进一步改进 [2020], ：网络从稠密状态开始，在动态移除幅值最小的权重的同时，依据梯度信息添加新的连接。同样，这些方法都属于启发式方法，缺乏坚实的理论基础。深度学习模型的另一个问题是，它们经常会做出过度自信的预测。Szegedy 等人 [2013], 证明，只需向图像添加少量噪声，就能诱使分类器做出完全错误的预测（且置信度很高），尽管在人眼看来图像毫无变化。反过来也可能发生：白噪声图像几乎总是会被确信无疑地归入某个特定类别 [Nguyen 等人, 2015]。

贝叶斯神经网络（贝叶斯神经网络）由 Neal [1992], MacKay [1995], 和Bishop[1997]提出。它们采用严谨的贝叶斯方法来处理参数和预测的不确定性，并纳入先验知识。在许多情况下，这能带来更可靠、过拟合程度更低的解决方案；然而，代价是极高的计算成本。直到最近，由于标准马尔可夫链蒙特卡罗（MCMC）方法的局限性——这种方法是用于复杂贝叶斯推断的主要定量程序——贝叶斯神经网络的推断还无法扩展到大型多元数据。近期，变分贝叶斯方法 [Gal, 2016] 使我们能够近似感兴趣的后验分布，并带来更具可扩展性的方法。

然而，贝叶斯神经网络往往参数过多且难以解释。因此，从贝叶斯视角考虑稀疏诱导方法通常很有意义。这通常通过使用稀疏诱导先验来实现，例如变分 dropout [King ma 等人，2015年，Molchanov 等人， 2017], 其在权重上使用独立对数均匀先验。这是一个非正规先验，意味着它不可积，因此不是有效的概率分布。正如 Hron 等人 [2017], 所指出的，将该先验与常用的似然函数结合使用会导致非正规后验分布，这意味着所得结果无法从贝叶斯建模的角度解释。有人认为，变分 dropout 应被解释为对变分参数进行的惩罚最大似然估计。此外，Gale 等人 [2019] 发现，尽管变分 dropout 在较小的网络上表现良好，但在较大的网络上，其性能不如启发式（非贝叶斯）方法。另一种稀疏诱导先验是独立尺度混合先验，其中 Blundell 等人 [2015]提出了两个高斯密度的混合模型：对第二个混合分量使用较小的方差，会使许多权重的先验集中在 0 附近。另一种可能性是使用独立尖峰与板块先验，这种先验最常用于贝叶斯线性回归模型。该先验用于

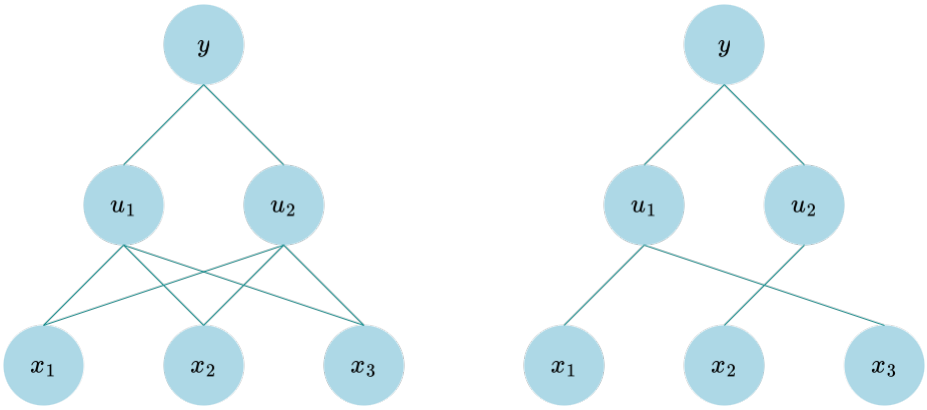

图 1：左侧为稠密网络，右侧为一种可能的稀疏结构。

由Hubin 和 Storvik [2019,2023]提出的潜在二元贝叶斯神经网络（LBBNN）。该先验从理论角度在Polson 和 Ro˘ckov´a [2018]中进行了研究。在Hubin 和 Storvik [2019] 中，实证表明，使用该先验会产生非常稀疏的网络（约 90 % 的权重被移除），同时保持良好的预测能力。LBBNN方法考虑了每个权重是否纳入的不确定性（结构不确定性），以及给定结构时纳入权重的不确定性（参数不确定性），从而实现完全贝叶斯的网络稀疏化方法（见图 1）。本文表明，与Hubin 和 Storvik[2019]中的平均场方法相比，使用归一化流变换变分后验分布，能够得到更加稀疏的网络，同时提升预测能力。此外，我们还证明，流网络能够很好地处理预测不确定性，并且在变量高度相关的逻辑回归场景中，其变量选择表现优于平均场方法。

2. 模型

给定解释变量 $\boldsymbol{x} \in \mathrm{R}^n$ 和响应变量 $\boldsymbol{y} \in \mathrm{R}^m$，神经网络对函数进行建模

$$\boldsymbol{y} \sim f(\boldsymbol{\eta}(\boldsymbol{x})).$$

均值向量 $\boldsymbol{\eta}$ 通过半仿射变换的复合得到：

$$u_j^{(l)} = \sigma^{(l)}\left(\sum_{i=1}^{n^{(l-1)}} u_i^{(l-1)} \gamma_{ij}^{(l)} w_{ij}^{(l)} + b_j^{(l)}\right), j = 1, \ldots, n^{(l)}, l = 1, \ldots, L, \tag{1}$$

其中 $\eta_j = u_j^{(L)}$。此外，$\boldsymbol{u}^{(l-1)}$ 表示来自前一层的输入（其中 $\boldsymbol{u}^0 = \boldsymbol{x}$对应解释变量），$w_{ij}^{(l)}$ 表示权重，$b_j^{(l)}$ 表示偏置项，而 $n^{(l)}$ （以及 $n^{(0)} = n$）表示总共 L 层中第 l 层的输入数量。此外，我们还有逐元素非线性激活函数 $\sigma^{(l)}$。额外的参数

$\gamma_{ij}^{(l)} \in \{0,1\}$ 表示相应权重的二元纳入变量。为便于阅读，从这里开始我们省略层记法，因为各层始终被视为相互独立。

接下来Hubin 和 Storvik [2019], Polson 和 Ro˘ckov´a [2018], 我们将结构定义为二元向量 $\boldsymbol{\gamma}$的配置，以及在此配置条件下每种结构的权重。为同时考虑结构和权重的不确定性，我们采用尖峰与厚尾先验。对于网络的每个（独立）层，我们也假设权重相互独立，从而得到 LBBNN模型：

$$p(w_{ij}|\gamma_{ij}) = \gamma_{ij}\mathcal{N}(0,\sigma^2) + (1-\gamma_{ij})\delta(w_{ij})$$
$$p(\gamma_{ij}) = \text{Bernoulli}(\alpha).$$

这里， $\delta(\cdot)$ 表示狄拉克δ函数，除零点处存在一个尖峰外，在其他所有位置均被视为零。此外， σ^2 和 α 分别表示权重的先验方差和先验纳入概率。在实际应用中，我们对所有层和权重使用相同的方差和纳入概率，但这并非严格必要的。

3. 贝叶斯推断

使用 LBBNN 的主要动机在于，我们能够同时考虑结构不确定性和参数不确定性，而标准贝叶斯神经网络只关注参数不确定性。通过后验预测分布进行推断，我们对所有可能的结构配置和参数取平均。对于给定数据 $\mathcal{D}$ 的新观测$\tilde{\boldsymbol{y}}$ ，我们有：

$$p(\tilde{\boldsymbol{y}}|\mathcal{D}) = \sum_{\boldsymbol{\gamma}} \int_{\boldsymbol{w}} p(\tilde{\boldsymbol{y}}|\boldsymbol{w},\boldsymbol{\gamma},\mathcal{D}) p(\boldsymbol{w},\boldsymbol{\gamma}|\mathcal{D})\, d\boldsymbol{w}.$$

由于 $\boldsymbol{w}$ 和 $\boldsymbol{\gamma}$的维度极高，这个表达式难以计算；而且，由于难以从后验分布中获取样本，使用蒙特卡罗抽样进行近似也很有挑战性， $p(\boldsymbol{w},\ \boldsymbol{\gamma}|\mathcal{D})$。我们不尝试从真实后验分布中抽样，而是将其转化为一个优化问题，采用变分推断[VI, Blei等人， 2017]。核心思想是用一个近似分布替代真实后验分布， $q_{\boldsymbol{\theta}}(\boldsymbol{w},\ \boldsymbol{\gamma})$，其中 $\boldsymbol{\theta}$ 表示一些变分参数。我们学习这些变分参数，使近似后验尽可能接近真实后验。我们通过 Kullback-Leibler（KL）散度来衡量这种接近程度：

$$\text{KL}\left[q_{\boldsymbol{\theta}}(\boldsymbol{w},\boldsymbol{\gamma})||p(\boldsymbol{w},\boldsymbol{\gamma}|\mathcal{D})\right] = \sum_{\boldsymbol{\gamma}} \int_{\boldsymbol{w}} q_{\boldsymbol{\theta}}(\boldsymbol{w},\boldsymbol{\gamma}) \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{w},\boldsymbol{\gamma})}{p(\boldsymbol{w},\boldsymbol{\gamma}|\mathcal{D})}\, d\boldsymbol{w}.$$

最小化 KL 散度（关于 $\boldsymbol{\theta}$）等价于最大化证据下界（ELBO）：

$$\text{ELBO}(q_{\boldsymbol{\theta}}) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{w},\boldsymbol{\gamma})}\left[\log p(\mathcal{D}|\boldsymbol{w},\boldsymbol{\gamma})\right] - \text{KL}\left[q_{\boldsymbol{\theta}}(\boldsymbol{w},\boldsymbol{\gamma})||p(\boldsymbol{w},\boldsymbol{\gamma})\right]. \tag{2}$$

因此，目标是在最大化期望对数似然的同时，根据先验与变分后验之间的 KL 散度进行惩罚。近似的效果取决于所选择的变分分布族 $\{q_{\boldsymbol{\theta}}, \boldsymbol{\theta} \in \Theta\}$ 。

3.1 变分族的选择

一种常见的选择 [Blundell 等人， 2015]对于（稠密）贝叶斯神经网络中的近似后验而言，是平均场高斯分布。为简化记号，现用 $\mathbf{W}$ 表示对应于特定层的权重集合。那么

$$q_{\boldsymbol{\theta}}(\mathbf{W}) = \prod_{i=1}^{n_{in}} \prod_{j=1}^{n_{out}} \mathcal{N}(\tilde{\mu}_{ij}, \tilde{\sigma}_{ij}^2),$$

其中， n_{in} 和 n_{out} 分别表示前一层和当前层中的神经元数量。不同层对应的权重也假设相互独立。

在Hubin 和 Storvik [2019], 中，贝叶斯神经网络的平均场高斯分布被扩展为纳入二元纳入变量，遵循Carbonetto 和 Stephens [2012]:

$$\begin{aligned} q_{\boldsymbol{\theta}}(\mathbf{W}|\boldsymbol{\Gamma}) &= \prod_{i=1}^{n_{in}} \prod_{j=1}^{n_{out}} [\gamma_{ij}\mathcal{N}(\tilde{\mu}_{ij}, \tilde{\sigma}_{ij}^2) + (1-\gamma_{ij})\delta(w_{ij})]; \\ q_{\tilde{\alpha}_{ij}}(\gamma_{ij}) &= \text{Bernoulli}(\tilde{\alpha}_{ij}). \end{aligned} \tag{3}$$

然而，平均场高斯分布通常过于简单，无法捕捉真实后验分布的复杂性。我们遵循 Ranganath 等人 [2016], 的做法，引入一组潜变量 $\boldsymbol{z}$ 来建模权重之间的依赖关系，并采用以下变分后验分布:

$$\begin{aligned} q_{\boldsymbol{\theta}}(\mathbf{W}|\boldsymbol{\Gamma}, \boldsymbol{z}) &= \prod_{i=1}^{n_{in}} \prod_{j=1}^{n_{out}} [\gamma_{ij}\mathcal{N}(z_i\tilde{\mu}_{ij}, \tilde{\sigma}_{ij}^2) + (1-\gamma_{ij})\delta(w_{ij})]; \\ q_{\tilde{\alpha}_{ij}}(\gamma_{ij}) &= \text{Bernoulli}(\tilde{\alpha}_{ij}), \end{aligned} \tag{4}$$

其中， $\boldsymbol{z}$ 服从分布$q_{\boldsymbol{\phi}}$（$\boldsymbol{z}$）。关于两种变分分布（3）和（4）之间差异的示意图，请参见图 2。因此，我们提出的新颖变分分布不仅能够建模权重之间的依赖关系，还能同时考虑权重和结构不确定性。对于$\mathbf{W}$，同样， $\boldsymbol{z}$ 也是一组与特定层相关的变量；对于 $\boldsymbol{z}$，我们也假设各层之间相互独立。为提高变分后验的灵活性，我们将归一化流 [Rezende 和 Mohamed， 2015] 应用于$q_{\boldsymbol{\phi}}$（$\boldsymbol{z}$）。一般而言，归一化流是由某个初始（简单）随机变量 $\mathbf{z}_0$，

$$\boldsymbol{z}_k = f_k(\boldsymbol{z}_{k-1}), \quad k = 1, ..., K.$$

变换后变量 $\boldsymbol{z} = \boldsymbol{z}_K$ 的对数密度为，

$$\log q_K(\boldsymbol{z}_K) = \log q_0(\boldsymbol{z}_0) - \sum_{k=1}^{K} \log \left| \det \frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{z}_{k-1}} \right|. \tag{5}$$

我们通常关注这样的变换：它们不仅具有高度灵活性，而且其雅可比行列式易于计算且计算速度快。对.....进行变换

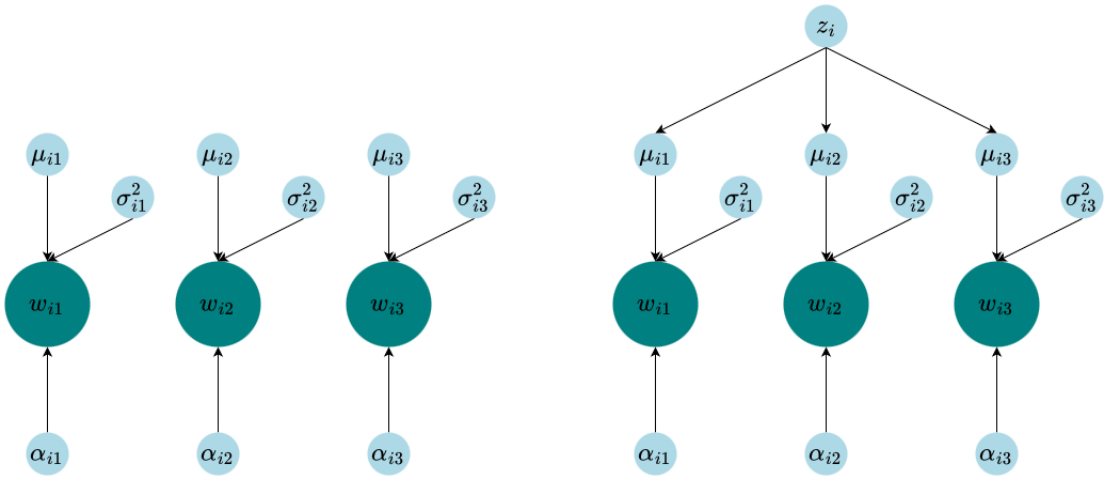

图 2：左侧为均值场变分后验，其中假设权重相互独立。右侧的潜在变分分布 z 可以对权重之间的依赖关系进行建模。

在贝叶斯神经网络中使用归一化流构造变分后验分布，最早由 Louizos 和 Welling [2017]，完成。他们创造了“乘性归一化流”（MNF）这一术语，指的是将变换应用于激活空间而非权重空间。由于权重的维度高得多，流参数的数量——进而变分分布的参数数量——会迅速膨胀。我们在这里也采用同样的方法。我们工作的主要区别在于，使用公式（4）中的变分后验，我们还能得到稀疏网络。

对于归一化流，我们将采用逆自回归流（IAF），并使用数值稳定的更新方法，该方法由Kingma 等人 [2016]提出。其通过以下方式对输入进行变换：

$$\begin{aligned} \boldsymbol{z}_{k-1} &= \text{input} \\ \boldsymbol{m}_k, \boldsymbol{s}_k &= \text{NeuralNetwork}(\boldsymbol{z}_k) \\ \boldsymbol{\kappa}_k &= \text{sigmoid}(\boldsymbol{s}_k) \\ \boldsymbol{z}_k &= \boldsymbol{\kappa}_k \odot \boldsymbol{z}_{k-1} + (1 - \boldsymbol{\kappa}_k) \odot \boldsymbol{m}_k, \end{aligned} \tag{6}$$

其中 $\odot$表示逐元素乘法。假设（6）中的神经网络是自回归的（即 $z_{k,\ i}$ 只能依赖于$z_{k,\ 1:i-1}$），我们得到一个下三角雅可比矩阵，并

$$\log \left| \det \frac{\partial \boldsymbol{z}_k}{\partial \boldsymbol{z}_{k-1}} \right| = \sum_{i=1}^{n_{in}} \log \kappa_{k,i}. \tag{7}$$

3.2 计算变分界

在实践中，我们最小化式（2）中的负证据下界。为了计算这个上界，我们需要从联合后验分布中边缘化掉 $\boldsymbol{z}$ （仍在同一层内，

省略层记法，并且从这里开始，为简化记法，也省略变分参数 $q(\cdot)'s$ 的下标）：

$$q(\boldsymbol{W}, \boldsymbol{\Gamma}) = \int q(\boldsymbol{W}, \boldsymbol{\Gamma}, \boldsymbol{z})\, d\boldsymbol{z}.$$

该表达式通常无法解析，因此我们必须采用近似方法来学习其参数。与Louizos 和 Welling [2017],类似，我们利用

$$\log q(\boldsymbol{W}, \boldsymbol{\Gamma}) = \log q(\boldsymbol{W}, \boldsymbol{\Gamma}|\boldsymbol{z}) + \log q(\boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{\Gamma}).$$

由此得到 KL 散度的如下表达式，

$$\begin{aligned} &\mathrm{KL}\left[q(\boldsymbol{W}, \boldsymbol{\Gamma})||p(\boldsymbol{W}, \boldsymbol{\Gamma})\right] = \\ &\quad \mathbb{E}_{q(\boldsymbol{W}, \boldsymbol{\Gamma}, \boldsymbol{z})}\Big[\mathrm{KL}\left[q(\boldsymbol{W}, \boldsymbol{\Gamma}|\boldsymbol{z})||p(\boldsymbol{W}, \boldsymbol{\Gamma})\right] + \log q(\boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{\Gamma})\Big]. \end{aligned} \tag{8}$$

经过一些代数运算后，第一项可写为：

$$\begin{aligned} &\mathrm{KL}\left[q(\boldsymbol{W}, \boldsymbol{\Gamma}|\boldsymbol{z})||p(\boldsymbol{W}, \boldsymbol{\Gamma})\right] \\ &= \sum_{ij}\left[\tilde{\alpha}_{ij}\left(\log\frac{\sigma_{ij}}{\tilde{\sigma}_{ij}} + \log\frac{\tilde{\alpha}_{ij}}{\alpha_{ij}} - \frac{1}{2} + \frac{\tilde{\sigma}_{ij}^2 + (\tilde{\mu}_{ij} z_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right) + (1 - \tilde{\alpha}_{ij}) \log\frac{1-\tilde{\alpha}_{ij}}{1-\alpha_{ij}}\right]. \end{aligned}$$

第二项则为

$$\log q_K(\boldsymbol{z}) = \log q_0(\boldsymbol{z}_0) - \sum_{i=1}^{n_{in}} \log \kappa_{k,i}.$$

第三项 $q(\boldsymbol{z}|\boldsymbol{W},\ \boldsymbol{\Gamma}$通常无法解析，数值计算也很困难。为解决这一问题，我们引入一个额外的辅助分布 $r_\theta(\boldsymbol{z}|\boldsymbol{W},\ \boldsymbol{\Gamma}$，其参数化形式为 θ，并根据 8 式得到上界，遵循 Ranganath 等人 [2016]的方法。

$$\begin{aligned} &\mathrm{KL}\left[q(\boldsymbol{W}, \boldsymbol{\Gamma})||p(\boldsymbol{W}, \boldsymbol{\Gamma})\right] \leq \\ &\quad \mathbb{E}_{q(\boldsymbol{W}, \boldsymbol{\Gamma}, \boldsymbol{z})}\Big[\mathrm{KL}\left[q(\boldsymbol{W}, \boldsymbol{\Gamma}|\boldsymbol{z})||p(\boldsymbol{W}, \boldsymbol{\Gamma})\right] + \log q(\boldsymbol{z}) - \log r(\boldsymbol{z}|\boldsymbol{W}, \boldsymbol{\Gamma})\Big]. \end{aligned} \tag{9}$$

该上界比原始上界更宽松（证明见 Ranganath 等人 [2016] ），但变分后验分布中的依赖结构可以弥补这一点。对于（9）中的最后一项 $\log\ r(\boldsymbol{z}|\boldsymbol{W},\ \boldsymbol{\Gamma})$，我们遵循 Louizos 和 Welling [2017]，并使用逆归一化流使该分布具有更大的灵活性，其中

$$r_B(\boldsymbol{z}_B|\boldsymbol{W}, \boldsymbol{\Gamma}) = \prod_{i=1}^{n_{in}} \mathcal{N}(\nu_i, \tau_i^2).$$

我们定义对 $\boldsymbol{W}$ 和 $\boldsymbol{\Gamma}$ 的依赖方式与Louizos 和 Welling [2017]类似：

$$\begin{aligned} \boldsymbol{\nu} &= n_{\text{out}}^{-1}(\boldsymbol{d}_1 \boldsymbol{s}^T)\mathbf{1}, \qquad \text{with } \boldsymbol{s} = \sigma(\mathbf{e}^T(\boldsymbol{W} \odot \boldsymbol{\Gamma})) \\ \log \boldsymbol{\tau^2} &= n_{\text{out}}^{-1}(\boldsymbol{d}_2 \boldsymbol{s}^T)\mathbf{1}. \end{aligned} \tag{10}$$

这里，$\mathbf{d}_1$, $\mathbf{d}_2$ 和 $\mathbf{e}$ 是形状与 $\boldsymbol{z}$相同的可训练参数。对于 σ，我们使用硬Tanh，而不是 Tanh（Louizos 和 Welling [2017]中使用的方法），因为这种方法在经验上效果更好。因此，对于（9）的最后一项，我们有：

$$\log r\left(\boldsymbol{z}|\mathbf{W}, \boldsymbol{\Gamma}\right) = \log r_B\left(\boldsymbol{z}_B|\mathbf{W}, \boldsymbol{\Gamma}\right) + \log \left|\det \frac{\partial \boldsymbol{z}_B}{\partial \boldsymbol{z}}\right|.$$

这意味着我们必须使用两个归一化流：一个用于从 $\boldsymbol{z}_0$ 变换到 $\boldsymbol{z} = \boldsymbol{z}_K$，另一个用于从 $\boldsymbol{z}_B$ 变换到 $\boldsymbol{z}$。这里，我们展示的逆归一化流只有一层，但通常可以像（5）那样扩展到任意层数。

对于偏置，我们假设它们彼此独立，且与权重相互独立。我们采用标准正态先验和平均场高斯近似后验。由于偏置不使用归一化流，因此只需计算两个高斯分布之间的 KL 散度：

$$\text{KL}\left[q(\boldsymbol{b})||p(\boldsymbol{b})\right] = \sum_{ij}\left[\log \frac{\sigma_{b_{ij}}}{\tilde{\sigma}_{b_{ij}}} - \frac{1}{2} + \frac{\tilde{\sigma}_{b_{ij}}^2 + (\tilde{\mu}_{b_{ij}} - \mu_{b_{ij}})^2}{2\sigma_{b_{ij}}^2}\right].$$

实践中，证据下界通过（随机）梯度算法进行优化，其中应用了重参数化技巧 [Kingma 和 Welling, 2013]并结合小批量。

4. 使用局部重参数化技巧和 MNF 扩展 LBNNNs

Hubin 和 Storvik 提出的 LBBNN方法的一个缺点是 [2019, 2023], ，训练期间的每次前向传播都需要对大型 $\boldsymbol{\Gamma}$ 和$\mathbf{W}$ 矩阵进行采样，这些矩阵由所有γ_{ij}组成，并且w_{ij}也组成相应矩阵，以便按照Hubin 和 Storvik[2019, 2023]中详细描述的随机变分推断过程，计算网络每一层的激活值。此外，由于γ_{ij}具有二元性质，Hubin 和 Storvik [2019, 2023]中需要进行连续松弛。本文将展示如何通过直接对式（ h_j 1）中的预激活值进行采样来规避这两个问题；这种方法通常称为局部重参数化技巧（LRT） [Kingma 等人， 2015]。我们的情况有所不同，因为还必须考虑二元纳入变量。这里提到h_j时，指的是应用非线性激活函数之前的激活值。随后，我们仍然使用与Hubin 和 Storvik [2019, 2023] 完全相同的随机变分推断优化算法，只是不再需要任何松弛。我们可以计算其均值和方差：

$$\begin{aligned} \mathbb{E}(h_j) &= \mathbb{E}\left[b_j + \sum_{i=1}^{N}[o_i \gamma_{ij} w_{ij}]\right] = \tilde{\mu}_{b_j} + \sum_{i=1}^{N}[o_i \tilde{\alpha}_{ij} \tilde{\mu}_{ij}] \\ \text{Var}(h_j) &= \text{Var}\left[b_j + \sum_{i=1}^{N}[o_i \gamma_{ij} w_{ij}]\right] = \tilde{\sigma}_{b_j}^2 + \sum_{i=1}^{N}[o_i^2 \tilde{\alpha}_{ij}(\tilde{\sigma}_{ij}^2 + (1 - \tilde{\alpha}_{ij})\tilde{\mu}_{ij}^2)]. \end{aligned}$$

这里， o 表示前一层的输出，由 N 个神经元组成。LRT 背后的基本思想是，若将相互独立的高斯随机变量相加，其和也将（精确地）服从高斯分布。在我们的情形中，虽然这些变量是相互独立的高斯分布的混合，但只要满足林德伯格条件，相互独立随机变量之和的中心极限定理仍然成立。我们还通过经验验证，使用 LRT 生成的激活值样本，其分布近似于通过对 $\mathbf{\Gamma}$ 和$\mathbf{W}$进行采样所生成的激活值样本。因此，我们可以根据上述公式给出的均值和方差，将激活值作为（相互独立的）高斯随机变量进行采样。此外，如Kingma 等人 [2015]所示，使用 LRT 还能降低梯度估计的方差。还需注意的是，通过对 $\boldsymbol{h}$ 进行采样而产生的近似，也可以视为直接针对 $p(\boldsymbol{h}|\mathcal{D})$的一种替代性变分近似。

作为第二项扩展，我们在激活空间中应用归一化流，以提高变分后验的灵活性。使用归一化流时，激活值 h_j 的均值和方差为：

$$\mathbb{E}(h_j) = \sum_{i=1}^{N} o_i z_i \tilde{\alpha}_{ij} \tilde{\mu}_{ij} + \tilde{\mu}_{b_j}$$
$$\mathrm{Var}(h_j) = \sum_{i=1}^{N} o_i^2 \tilde{\alpha}_{ij} (\tilde{\sigma}_{ij}^2 + (1 - \tilde{\alpha}_{ij}) z_i^2 \tilde{\mu}_{ij}^2) + \tilde{\sigma}_{b_j}^2,$$

需要注意的是 $\boldsymbol{z}$ 会同时影响高斯近似的均值和方差，而在Louizos 和 Welling [2017] 的方法中，它只影响均值。Louizos 和 Welling [2017] 还会为小批量中的每个观测样本抽取一个 $\boldsymbol{z}$ 。我们发现，从经验上看，只抽取一个向量，并将同一个 $\boldsymbol{z}$ 与每个输入向量相乘，对性能没有影响。我们采用这种做法，因为计算效率更高。

5. 实验

5.1 背景

在本节中，我们主要关注将Hubin 和 Storvik [2023], 提出的基线方法（在他们的论文中称为 LBBNN-GP-MF，本文中简称 LBBNN）与本文详细介绍的两种扩展方法进行比较。第一种方法使用局部重参数化技巧（LRT），记为 LBBNN-LRT；第二种方法在变分后验分布上使用乘性归一化流（以及 LRT），记为 LBBNN-FLOW。在Hubin 和 Storvik [2019, 2023], 的全面分类实验中，结果表明，LBBNN 能够大幅稀疏化贝叶斯神经网络，同时保持较高的预测能力。此外，Hubin 和 Storvik [2023]还考察了一项实验，其中通过在 logit 尺度上对后验纳入参数使用多元正态分布，将依赖关系内置于变分后验分布中。在Hubin 和 Storvik [2023], 所考察的示例中，多元正态选项的表现并不优于平均场方法，因此我们将其从比较方法中剔除。在此，我们展示，通过归一化流提高变分后验的灵活性，相较于

在Hubin 和 Storvik [2019, 2023]所考察的方法的基础上，我们针对一组基线分类问题进行了比较。此外，我们还开展了两项模拟研究。第一项研究考察了逻辑回归情境下的变量选择，其中解释变量之间存在高度相关性。第二项研究从二维高斯分布簇中生成数据，并比较不同方法处理预测不确定性的效果。所有实验均使用 PyTorch 深度学习库在 Python 中编写 [Paszke 等人，2019]。

5.2 分类实验

我们进行了两项分类实验：一项采用全连接架构 [，如Hubin 和 Storvik所示，2019, 2023], 另一项采用卷积架构（具体实现方法见附录A ）。两种情况下，我们都在 MNIST[Deng、 2012], FMNIST（时尚 MNIST） [Xiao 等人、 2017] 以及 KMNIST（草书假名 MNIST） [Clanuwat 等人上进行分类， 2018]。MNIST 是一个包含 0 到 9 手写数字的数据库。FMNIST 包含来自 Zalando（欧洲最大的在线时尚零售商）数据库的十种不同服饰。最后，KMNIST 也包含十个类别，每个类别代表日语假名音节文字平假名中的一行。这些数据集均包含 28x28 的灰度图像，并分别划分为包含 60 000 张图像的训练集和包含 10 000 张图像的验证集。MNIST 和 FMNIST 是广为人知且常用的数据集，因此在测试新算法时很容易比较其性能。KMNIST 则是较新的数据集，被认为比经典的 MNIST 数字数据集更具挑战性，因为每个平假名都可能有许多不同的字符。

对于采用全连接架构的实验，我们使用与 Hubin 和 Storvik [2019, 2023]相同的设置。我们使用两个隐藏层，分别包含 400 和 600 个神经元，采用 ReLU [Agarap， 2018] 激活函数，以及 Adam [Kingma 和 Ba，2014] 优化器。批次大小设为 100，训练 250 轮。所有实验均运行 10 次，我们报告这 10 次运行中预测准确率的最小值、中位数和最大值。报告的密度（1−稀疏度）是这 10 次运行的平均值。对于 LBBNN-LRT 和 LBBNN-FLOW 方法，我们对网络中的所有权重和偏置使用标准正态先验，并将先验纳入概率设为 0.10。对于 $q(\boldsymbol{z})$ 和 $r(\boldsymbol{z}|\mathbf{W}\ \boldsymbol{\Gamma})$，我们使用长度为 2 的流，其中神经网络包含两个各有 250 个神经元的隐藏层。对于第二个分类实验，我们采用 LeNet-5 [LeCun 等人， 1998] 卷积架构，但卷积层使用 32 个和 48 个滤波器。我们使用与前一个实验相同的先验、归一化流和数据集。

为了衡量预测性能，我们考虑两种方法。首先是完全贝叶斯模型平均方法，即对来自变分后验分布的 100 个样本进行平均，同时考虑权重和结构两方面的不确定性。其次，我们考虑中位数概率模型 [Barbieri and Berger， 2004], 在这种方法中，我们仅对后验纳入概率大于 0.5 的权重进行模型平均，遵循 Hubin 和 Storvik [2019, 2023], 的方法，而其他权重则从模型中排除。这使得网络能够实现显著的稀疏化。我们强调，这是可行的，因为在进行推断时，我们可以重新对权重进行采样。我们还报告了密度，即中位数概率模型中所包含权重的比例。

表1：全连接架构在 KMNIST、MNIST、FMNIST 验证数据上的性能指标。对于准确率（%），我们报告十次不同运行中的最小值、最大值和中位数。密度按十次运行的平均值计算。

KMNIST	中位数概率模型				全模型平均			
方法	min	中位数	max	密度	min	中位数	max	密度
LBBNN	89.22	89.59	89.98	0.113	89.43	89.76	90.21	1.000
LBBNN-LRT	90.04	90.26	90.43	0.136	90.23	90.39	90.60	1.000
LBBNN-FLOW	90.64	**91.12**	91.46	0.096	91.16	**91.30**	91.61	1.000
MNIST	中位数概率模型				全模型平均			
方法	min	中位数	max	密度	min	中位数	max	密度
LBBNN	98.01	98.10	98.20	0.098	98.03	98.14	98.23	1.000
LBBNN-LRT	97.84	97.95	98.09	0.103	98.01	98.08	98.11	1.000
LBBNN-FLOW	98.14	**98.36**	99.42	0.074	98.23	**98.42**	98.53	1.000
FMNIST	中位数概率模型				全模型平均			
方法	min	中位数	max	密度	min	中位数	max	密度
LBBNN	88.47	88.76	88.90	0.106	88.60	88.74	88.91	1.000
LBBNN-LRT	87.51	87.82	87.94	0.141	87.88	87.94	88.14	1.000
LBBNN-FLOW	89.49	**89.70**	89.88	0.097	89.52	**89.80**	89.92	1.000

全连接架构的结果见表1 ，卷积架构的结果见表2。首先，我们看到，使用 LBBNN-LRT 得到的结果与基线LBBNN方法相当，但在 FMNIST 上，无论使用全连接架构还是卷积架构，其表现都稍差一些。这些结果相似并不意外，因为使用局部重参数化技巧主要带来的是计算上的优势。其次，我们注意到，在卷积架构和全连接架构上，LBBNN-FLOW方法的表现都优于另外两种方法，同时网络也最为稀疏。我们注意到，我们的方法在采用卷积架构的 MNIST 上取得的准确率与Louizos 和 Welling[2017]（99.30%）报告的准确率相当。这表明，可以在不损失太多预测能力的情况下稀疏化卷积贝叶斯神经网络。总体而言，卷积架构上的密度较高，主要是由于参数初始化略有不同；不过，这些网络也可以稀疏到与全连接网络相近的程度。使用归一化流带来的预测能力提升需要付出计算成本。对于全连接架构，我们观察到，在 NVIDIA A10 GPU 上，使用 LBBNN-LRT 训练一轮约需 4 秒，使用 LBBNN 需 13 秒，使用 LBBNN-FLOW 则需 17 秒。对于卷积架构，使用 LBBNN-LRT 每轮需 7 秒，使用 LBBNN 需 18 秒，使用归一化流则需 28 秒。

<escape>表2：采用卷积架构时，在 KMNIST、MNIST、FMNIST 验证数据上的性能指标。有关更多详细信息，请参见表1 中的说明。</escape>

KMNIST	中位数概率模型				全模型平均			
方法	min	中位数	max	密度	min	中位数	max	密度
LBBNN	95.13	95.52	95.89	0.359	95.21	95.48	95.78	1.000
LBBNN-LRT	94.73	94.94	95.16	0.429	95.07	95.42	95.65	1.000
LBBNN-FLOW	95.73	**95.99**	96.43	0.351	96.00	**96.18**	96.44	1.000
MNIST	中位数概率模型				全模型平均			
方法	min	中位数	max	密度	min	中位数	max	密度
LBBNN	99.22	99.26	99.35	0.353	99.21	99.28	99.33	1.000
LBBNN-LRT	99.11	99.26	99.31	0.406	99.20	99.28	99.34	1.000
LBBNN-FLOW	99.15	**99.27**	99.41	0.338	99.16	**99.29**	99.42	1.000
FMNIST	中位数概率模型				全模型平均			
方法	min	中位数	max	密度	min	中位数	max	密度
LBBNN	91.14	91.31	91.48	0.352	91.10	91.26	91.44	1.000
LBBNN-LRT	90.04	90.40	90.85	0.433	90.52	90.73	91.06	1.000
LBBNN-FLOW	90.52	**91.54**	91.75	0.367	91.38	**91.71**	92.04	1.000

5.3 逻辑回归模拟研究

本节在逻辑回归框架下进行变量选择实验。由于逻辑回归只是具有一个神经元（因而只有一层）的神经网络的特例，因此修改算法十分简单。我们将研究范围限定在逻辑回归场景，以便与Carbonetto 和 Stephens [2012], 提出的原始基线方法进行比较。他们表明，当协变量存在相关性时，平均场变分近似在变量选择任务中的表现会开始变差。我们使用与 Hubin 和 Storvik [2018], 相同的数据，其中包含20个二元和连续变量，以及一个二元结果变量，共有2 000个观测值。协变量 $\boldsymbol{x}$由许多变量之间强而复杂的相关结构生成（见图3）。有关协变量具体生成方式的更多详情，请参见Hubin 和 Storvik [2018]的附录B。响应变量 y根据以下数据生成过程生成：

$$\eta \sim \mathcal{N}(\boldsymbol{\beta x}, 0.5)$$
$$y \sim \text{Bernoulli}\left(\frac{\exp(\eta)}{1+\exp(\eta)}\right)$$

其中回归参数定义为：

$$\boldsymbol{\beta} = (-4, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1.2, 0, 37.1, 0, 0, 50, -0.00005, 10, 3, 0).$$

目标是训练不同的方法，以选择 $\boldsymbol{\beta}$ 中非零的元素。如果后验纳入概率 $\alpha_j > 0.5$，即，我们就认为参数 β_j 被纳入。

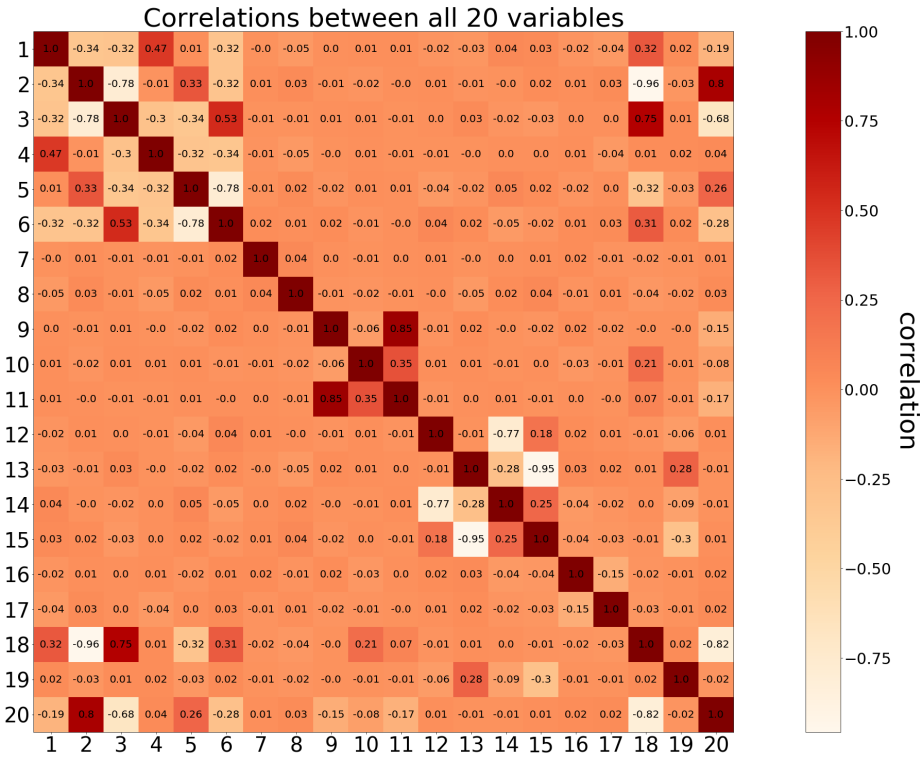

图3：展示逻辑回归模拟研究中不同变量之间相关性的图。

表3：逻辑回归变量选择模拟研究的性能指标。

	CS	LBBNN-LRT	LBBNN-FLOW
平均真正率	0.681	0.838	**0.972**
平均假正率	0.125	0.084	**0.074**

Barbieri and Berger 提出的中位数概率模型Barbieriand Berger [2004]。我们对每种方法（使用相同的数据）拟合 100 次，每次都计算真正率（TPR）和假正率（FPR）：

$$\mathrm{TPR} = \sum_{j=1}^{M} \frac{\mathrm{TP}_j}{\mathrm{TP}_j + \mathrm{FN}_j},$$
$$\mathrm{FPR} = \sum_{j=1}^{M} \frac{\mathrm{FP}_j}{\mathrm{FP}_j + \mathrm{TN}_j}.$$

这里 $n = 20$ 变量，TP = 真正阳性，表示一个非零权重被正确纳入。FN = 假阴性，表示一个非零权重未被纳入。FP =假阳性，表示一个零权重被纳入。TN =真阴性，表示一个零权重未被纳入。因此，TPR 衡量的是非零权重变量被正确纳入的比例，而 FPR 衡量的是零权重变量被错误纳入的比例。

在本实验中，我们将我们的方法 LBBNN-LRT 和 LBBNN-FLOW 与 Carbonetto 和 Stephens [2012], 提出的算法进行比较，下文将其记为 CS。该方法与 LBBNN-LRT 非常相似，因为它使用相同的变分分布，但其优化采用重要性抽样和坐标上升变分推断（不对数据进行子采样）。对于归一化流，我们使用长度为二的流，神经网络包含两个隐藏层，每层有 100 个神经元。我们将批次大小设为 400，并训练 500 个训练轮次。对于三种方法，我们都对权重使用标准正态先验，并对纳入指标使用 0.25 的先验纳入概率。因此，我们处于带有变量选择的贝叶斯逻辑回归设置中。

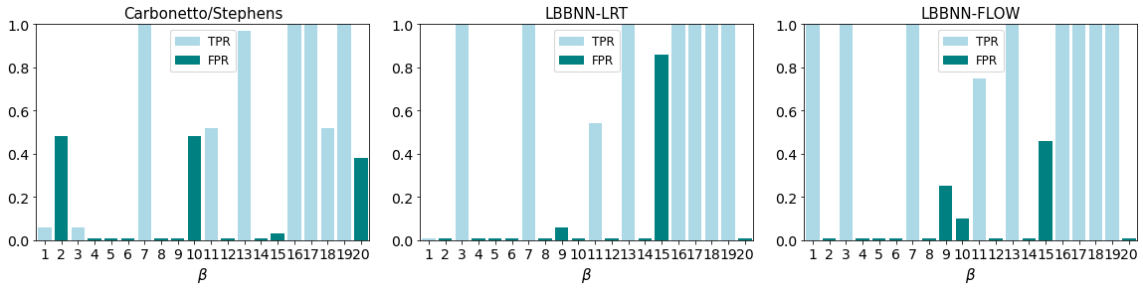

图4：展示100次运行中权重被纳入模型的频率的条形图 s.

结果见 表3。我们还针对 100 次运行中 20 个权重的每一个绘制了柱状图（图4）。我们看到，LBBNN-FLOW 的表现最佳，具有最高的真正率和最低的假正率。在许多变量之间存在高度相关性的情况下（例如 $\beta_1 - \beta_6$），它尤其擅长识别出正确的变量。我们可以将此归因于其更灵活的变分后验分布，而其他三种方法使用的是平均场高斯分布。Carbonetto 和 Stephens [2012] 还讨论了，只有当变量相互独立或至多存在较弱的相关性时，平均场方法才可能被视为一种良好的近似。

5.4 预测不确定性

使用贝叶斯神经网络（BNNs）的一个重要动机在于，与非贝叶斯神经网络相比，它们能够更准确地处理预测不确定性。因此，在本实验中，我们希望展示我们的方法 LBBNN-LRT 和 LBBNN-FLOW，以及蒙特卡洛 dropout [Gal 和 Ghahramani、 2016]，以及常规（稠密）贝叶斯神经网络在预测不确定性方面的表现。因此，本研究旨在进行说明，而非比较，这些方法在此并非相互竞争。对于本实验，我们模拟了来自二维高斯分布的 5 个数据簇。对于这五个高斯分布，我们采用附录 B中报告的均值和协方差。随后，将数据转换到 0 到 1 的范围内，以便于可视化。任务是将数据分类到与特定簇对应的正确类别中。我们生成了三个数据集，分别包含 10、50 和 200

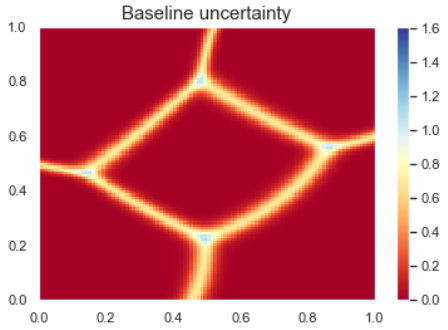

图5：预测不确定性实验中高斯簇的数据生成不确定性。

分别从每个类别中抽取的样本。对于所有方法，我们拟合一个包含单个隐藏层的网络，该隐藏层由 1000 个神经元组成。这意味着我们所处的场景中，可训练参数的数量远大于观测值的数量，这是贝叶斯神经网络（BNN）应用中的典型情况。对于 Dropout，我们将 dropout 概率设为 0.5；对于 LBBNN-LRT 和 LBBNN-FLOW，我们将先验纳入概率设为 0.5。我们使用长度为 2 的流，神经网络由两个隐藏层组成，每个隐藏层包含 50 个神经元。对于所有方法，我们使用 10 个样本进行模型平均。为了衡量预测不确定性，我们在 $[0,1]^2$ 上的网格中生成一个测试集，并计算网格中每个点的预测分布的熵。当预测分布.....时，熵达到最大值。

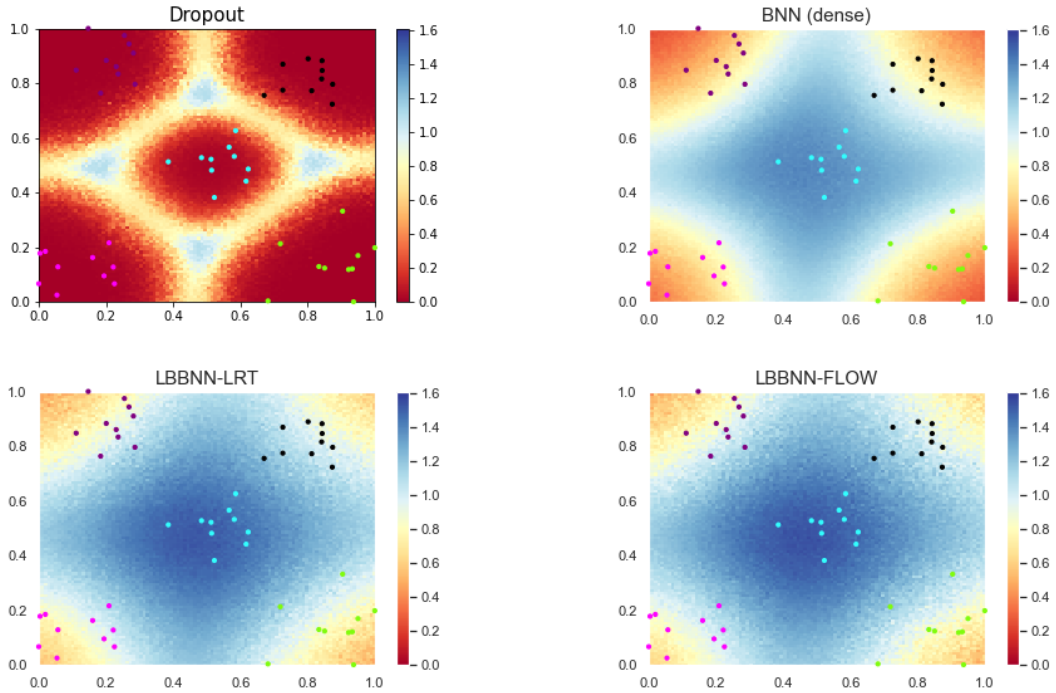

图6：每个簇中10个样本的熵

分布是均匀的，即每个类别的概率均为 0.2。结果如 图6、图7和图8所示。此外，我们在图5中展示了数据生成模型的熵；随着样本量增加，预测熵预计会收敛到该熵值。

在数据较少时，Dropout 与贝叶斯网络之间呈现出显著差异。除了类别之间的决策边界附近，Dropout 的预测在所有区域都高度自信。相比之下，贝叶斯网络在大多数区域都表现出较高的不确定性，尤其是在观测数据较少的区域。随着数据量增加，我们可以看到贝叶斯网络对预测逐渐变得更加自信，而且熵（如期望的那样）开始趋近于数据生成熵；而对于给定比率的 Dropout，不确定性并未降低。需要指出的是，这并非欠拟合，因为所有方法在训练期间的准确率都接近 100%。最后，我们观察到，稠密贝叶斯神经网络的不确定性通常略低于采用局部重参数化技巧和 FLOW 的 LBBNN。不过，我们无法据此判断这种情况是好是坏，因为在 LBBNN 的设置下，通过可逆跳跃马尔可夫链蒙特卡罗难以获得真实的不确定性。

此外，我们进行了一项实验：在使用 50 个样本（每个簇 10 个）进行训练后，生成 10 000 个测试样本（每个簇 2 000 个）。训练完成后，我们计算测试数据上预测分布的熵，并按照熵从低到高对数据进行排序。我们还根据最大类别概率对样本进行排序，并计算累积准确率（每次使用 100 个数据样本）。具体来说，我们首先计算置信度最高的 100 个预测的准确率，随后是置信度稍低的 100 个预测的准确率，

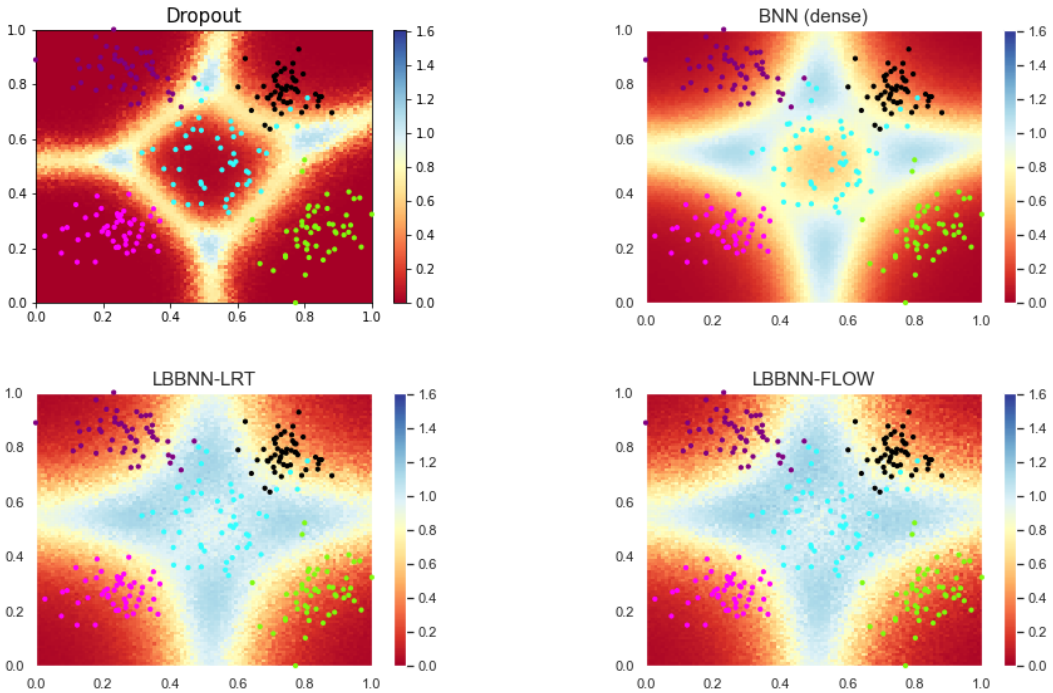

图7：每个簇使用50个样本时的熵

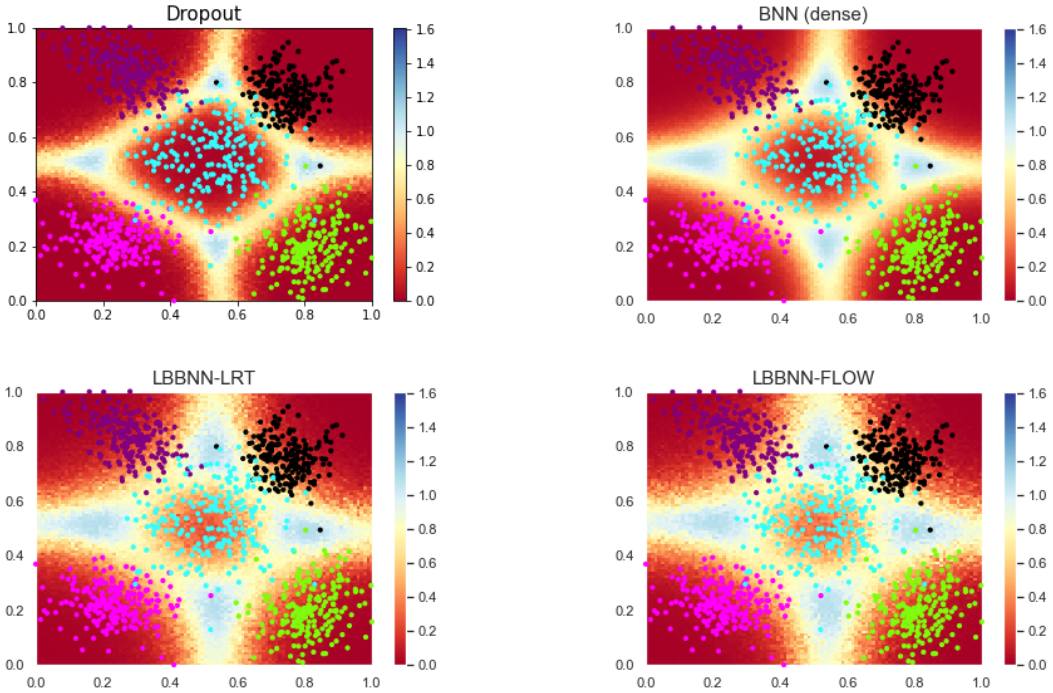

图8：每个簇取200个样本计算的熵

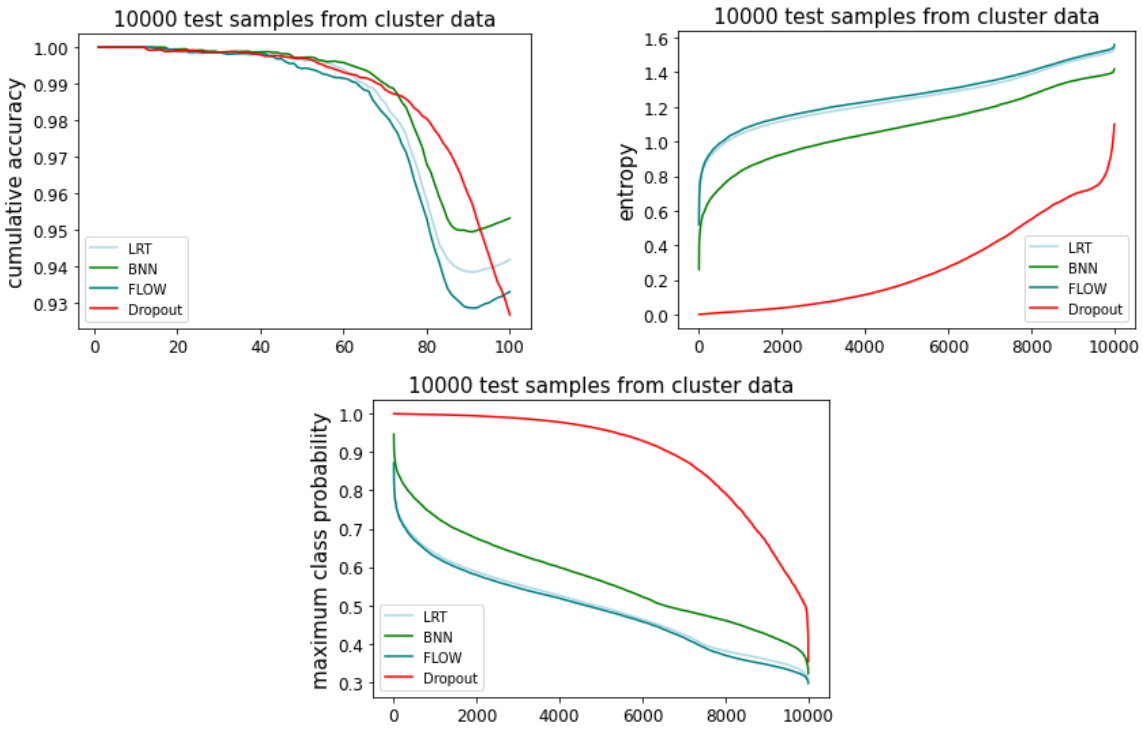

图 9：左上，累积准确率（每次取 100 个样本），其中每个点表示对应数据点的准确率。右上，熵按从低到高的顺序排列。下方，最大类别概率按从高到低的顺序排列。

置信度高的预测，依此类推，直到我们得到 100 个置信度最低的预测。结果见 图 9。使用 Dropout 时，最大类别概率通常非常高（也就是说，我们对样本属于哪个类别极有把握）。在最先的 5 000 个（排序后的）样本之后，最可能类别的输出概率约为 95%。而使用局部重参数化技巧和 FLOW 时，该概率则降至约 50%。这与我们先前看到的情况一致：Dropout 在大多数时候都具有很高的确定性。尽管如此，我们发现，在这个实验中，对于不确定性最高的情况，贝叶斯方法的预测准确率高于 Dropout。

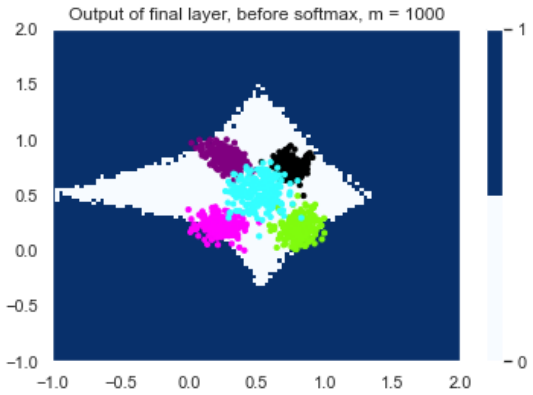

图 10：分布外熵，其中深蓝色对应贝叶斯神经网络检测到的分布外数据，白色为分布内数据。

作为最后一个示例，我们考虑这样一项实验：将最后一层的模型平均预激活输出（softmax 之前，即应用 softmax 函数之前）的最大值作为衡量指标，而不是使用熵。我们使用训练数据（m =1000）为所有类别的模型平均预激活输出生成经验置信区间。我们对上界使用单侧 95% 置信区间。测试期间，我们在一个网格上生成样本，该网格现在在两个维度上的范围均为 -1 到 2，并取模型平均预激活输出的最大值。然后，我们检查它是否落在经验置信区间内。结果如图 10所示。我们发现，在熵极低的区域中，可以检测到分布外数据。这表明，像 Louizos 和 Welling [2017] 所建议的那样，将最大熵用于分布外数据可能并非最优。不过，我们仍然看到了贝叶斯神经网络利用其输出的预激活值区分域内与域外不确定性的潜力。我们在此不再深入探讨，将这一主题留待未来研究。

6. 讨论

我们已经证明，与基线方法（采用平均场后验）相比，通过归一化流提高变分后验分布的灵活性，能够在获得更稀疏网络的同时提升预测能力，尽管其变分界比平均场方法更宽松。此外，流方法在 a 上表现最佳。

变量选择问题，在这种问题中，平均场方法难以处理高度相关的变量。更普遍地说，我们认为，贝叶斯神经网络（贝叶斯神经网络）比其频率学派对应方法更擅长获得符合实际的预测不确定性估计，因为在数据稀疏时，它们的不确定性更高。与我们的方法相比，我们没有观察到稠密贝叶斯神经网络所得的不确定性估计有很大差异。与稠密贝叶斯神经网络不同，我们的方法还有能够执行变量选择的优势。其缺点是，LBBNN 的每个权重都需要一个额外参数，因此计算效率低于稠密贝叶斯神经网络。使用归一化流会进一步增加计算负担，因为我们还必须优化所有额外的流参数。

<escape>在本文中，我们对所有权重和纳入指标使用相同的先验，尽管这并非必要。未来研究的一个可能方向是调整先验纳入概率，以形成不同的稀疏结构。目前，给定某种神经网络架构，我们考虑了权重和参数中的不确定性。未来，研究是否也能够将激活函数中的不确定性纳入模型，可能会很有意义。通过在各层之间建立连接，我们可以学习跳过所有非线性层，只要线性函数足以满足需求即可。一种可能的应用是使用我们的方法开展全基因组关联研究（GWAS）。将 LBBNN 与 GWAS 相结合的方案已由Demetci 等人 [2021], 提出，然而，该方案仅使用平均场后验分布。通过我们基于归一化流的方法，除了能够对不同 SNP 集合之间的依赖关系进行建模外，还可以轻松地对每个 SNP 集合内部的依赖关系进行建模。</escape>

补充材料

GitHub： 实验所用的代码可在以下地址找到
https://github.com/LarsELund/Sparsifying-BNNs-with-LRT-and-NF

参考文献

A. F. Agarap. 使用线性整流单元（ReLU）进行深度学习。*arXiv*预印本 *arXiv:1803.08375*，2018年。M. M. Barbieri 和 J. O. Berger。最优预测模型选择。《统计学年鉴》，32(3):870–897，2004年。C. M. Bishop。贝叶斯神经网络。《巴西计算机学会期刊》，4(1)，1997年。D. M. Blei、A. Kucukelbir 和 J. D. McAuliffe。变分推断：统计学家的综述。《美国统计学会杂志》，112(518):859–877，2017年。C. Blundell、J. Cornebise、K. Kavukcuoglu 和 D. Wierstra。神经网络中的权重不确定性。载于国际机器学习会议，第1613–1622页。PMLR，2015年。P. Carbonetto 和 M. Stephens。用于回归中贝叶斯变量选择的可扩展变分推断及其在遗传关联研究中的准确性。《贝叶斯分析》，7(1):73–108，2012年。

T. Clanuwat、M. Bober-Irizar、A. Kitamoto、A. Lamb、K. Yamamoto 和 D. Ha。用于日本古典文学的深度学习。*arXiv*预印本 *arXiv:1812.01718*，2018年。P. Demetci、W. Cheng、G. Darnell、X. Zhou、S. Ramachandran 和 L. Crawford。利用生物学注释神经网络对遗传性状架构进行多尺度推断。*PLoS Genetics*，17(8):e1009754，2021年。L. Deng。用于机器学习研究的手写数字图像MNIST数据库[网络精选]。*IEEE*信号处理杂志，29(6):141–142，2012年。U. Evci、T. Gale、J. Menick、P. S. Castro 和 E. Elsen。操纵彩票：让所有彩票券都成为赢家。载于国际机器学习会议，第2943–2952页。PMLR，2020年。J. Frankle 和 M. Carbin。彩票券假说：寻找稀疏可训练神经网络。*arXiv*预印本 *arXiv:1803.03635*，2018 年。Y. Gal。深度学习中的不确定性。博士论文，剑桥大学，2016年。Y. Gal 和 Z. Ghahramani。将Dropout作为贝叶斯近似：表示深度学习中的模型不确定性。载于国际机器学习会议，第1050–1059页。PMLR，2016年。T. Gale、E. Elsen 和 S. Hooker。深度神经网络稀疏度的现状。*arXiv*预印本 *arXiv:1902.09574*，2019年。S. Han、J. Pool、S. Narang、H. Mao、E. Gong、S. Tang、E. Elsen、P. Vajda、M. Paluri、J. Tran、B. Catanzaro 和 W. J. Dally。DSD：深度神经网络的稠密-稀疏-稠密训练，2017年。J. Hron、A. G. d. G. Matthews 和 Z. Ghahramani。变分高斯Dropout并非贝叶斯方法。*arXiv*预印本 *arXiv:1711.02989*，2017年。A. Hubin 和 G. Storvik。广义线性混合模型中贝叶斯变量选择的模式跳跃马尔可夫链蒙特卡罗。计算统计与数据分析，127:281–297，2018年11月。ISSN 0167-9473。doi: 10.1016/j.csda.2018.05.020。URLhttp://dx.doi.org/10.1016/j.csda.2018.05.020。A. Hubin 和 G. Storvik。贝叶斯神经网络中的模型与参数不确定性结合。*arXiv:1903.07594*，2019年。A. Hubin 和 G. Storvik。模型与参数不确定性下贝叶斯神经网络的变分推断。*arXiv:2305.00934*，2023年。A. Khan、A. Sohail、U. Zahoora 和 A. S. Qureshi。深度卷积神经网络近期架构综述。人工智能评论，53(8):5455–5516，2020年。D. P. Kingma 和 J. Ba。Adam：一种随机优化方法。*arXiv*预印本 *arXiv:1412.6980*，2014年。D. P. Kingma 和 M. Welling。自编码变分贝叶斯，2013年。

D. P. Kingma、T. Salimans 和 M. Welling。变分 dropout 与局部重参数化技巧。神经信息处理系统进展，28，2015年。D. P. Kingma、T. Salimans、R. Jozefowicz、X. Chen、I. Sutskever 和 M. Welling。使用逆自回归流改进变分推断。神经信息处理系统进展，29，2016年。Y. LeCun、L. Bottou、Y. Bengio 和 P. Haffner。应用于文档识别的基于梯度的学习。*IEEE* 会议录，86(11)：2278–2324，1998。Y. Li。深度强化学习：概述。*arXiv* 预印本 *arXiv:1701.07274*，2017年。C. Louizos 和 M. Welling。用于变分贝叶斯神经网络的乘性归一化流。载于 国际机器学习会议，第 2218–2227 页。PMLR，2017年。

D. J. MacKay. 贝叶斯神经网络和密度网络。 核仪器与方法：物理研究 *A* 辑：加速器、谱仪、探测器及相关设备，354(1):73–80，1995年。

W.S. McCulloch 和 W. Pitts。《神经活动中固有思想的逻辑演算》。数学生物物理学通报，5(4):115–133，1943年。

D. Molchanov、A. Ashukha 和 D. Vetrov。变分 dropout 使深度神经网络稀疏化。发表于国际机器学习会议，第 2498–2507 页。PMLR，2017年。

R. M. Neal。通过混合蒙特卡洛方法对反向传播网络进行贝叶斯训练。技术报告，Citeseer，1992年。

A. Nguyen、J. Yosinski 和 J. Clune。深度神经网络很容易受到欺骗：对无法识别的图像给出高置信度预测。载于*IEEE*计算机视觉与模式识别会议论文集，第427–436页，2015年。

A. Paszke、S. Gross、F. Massa、A. Lerer、J. Bradbury、G. Chanan、T. Killeen、Z. Lin、N. Gimelshein、L. Antiga、A. Desmaison、A. Kopf、E. Yang、Z. De- Vito、M. Raison、A. Tejani、S. Chilamkurthy、B. Steiner、L. Fang、J. Bai 和 S. Chintala。PyTorch：一种命令式风格的高性能深度学习库。载于神经信息处理系统进展第 *32* 卷，第 8024–8035 页。Curran Associates, Inc.，2019年。网址：
h
t
t
p://
papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf。

N. G. Polson 和 V. Roˇckov´a。《稀疏深度学习的后验集中》。神经信息处理系统进展，31，2018年。L. Prechelt。《提前停止——但何时停止？》。载于神经网络：技巧大全，第 55–69 页。施普林格，1998。

R. Ranganath、D. Tran 和 D. Blei。《层次变分模型》。载于国际机器学习会议，第 324–333 页。PMLR，2016年。D. Rezende 和 S. Mohamed。《基于归一化流的变分推断》。载于国际机器学习会议，第 1530–1538 页。PMLR，2015年。D. E. Rumelhart、G. E. Hinton 和 R. J. Williams。《通过反向传播误差学习表征》。自然，323(6088):533–536，1986年。C. Shorten 和 T. M. Khoshgoftaar。《深度学习图像数据增强综述》。大数据杂志，6(1):1–48，2019年。N. Srivastava、G. Hinton、A. Krizhevsky、I. Sutskever 和 R. Salakhutdinov。《Dropout：一种防止神经网络过拟合的简单方法》。机器学习研究杂志，15(1):1929–1958，2014年。C. Szegedy、W. Zaremba、I. Sutskever、J. Bruna、D. Erhan、I. Goodfellow 和 R. Fergus。《神经网络的有趣特性》。*arXiv* 预印本 *arXiv:1312.6199*，2013年。A. Voulodimos、N. Doulamis、A. Doulamis 和 E. Protopapadakis。《用于计算机视觉的深度学习：简要综述》。计算智能与神经科学，2018，2018年。H. Xiao、K. Rasul 和 R. Vollgraf。《Fashion-MNIST：用于机器学习算法基准测试的新型图像数据集》。*arXiv* 预印本 *arXiv:1708.07747*，2017年。T. Young、D. Hazarika、S. Poria 和 E. Cambria。《基于深度学习的自然语言处理最新趋势》。*IEEE* 计算智能杂志，13(3):55–75，2018年。

附录A. 卷积架构

对于卷积层，变分分布定义为：

$$
\begin{aligned}
q_{\boldsymbol{\theta}}(\mathbf{W}|\boldsymbol{\Gamma}, \boldsymbol{z}) &= \prod_{i=1}^{n_h}\prod_{j=1}^{n_w}\prod_{k=1}^{n_f}[\gamma_{ijk}\mathcal{N}(z_k\tilde{\mu}_{ijk}, \tilde{\sigma}^2_{ijk}) + (1-\gamma_{ijk})\delta(w_{ijk})] \\
q_{\tilde{\alpha}_{ijk}}(\gamma_{ijk}) &= \text{Bernoulli}(\tilde{\alpha}_{ijk}),
\end{aligned}
\tag{11}
$$

其中 n_h、 n_w 和 n_f 分别表示卷积核的高度、宽度和滤波器数量。

对于卷积层，逆归一化流采用以下形式：

$$
\begin{aligned}
\boldsymbol{\nu} &= ((\text{Mat}(\boldsymbol{W} \odot \boldsymbol{\Gamma})\mathbf{e}) \otimes \mathbf{d}_1)\,(\mathbf{1} \odot (n_{\text{h}} n_{\text{w}})^{-1}) \\
\log \boldsymbol{\tau^2} &= ((\text{Mat}(\boldsymbol{W} \odot \boldsymbol{\Gamma})\mathbf{e}) \otimes \mathbf{d}_2)\,(\mathbf{1} \odot (n_{\text{h}} n_{\text{w}})^{-1}).
\end{aligned}
\tag{12}
$$

<escape>这里，Mat(·) 表示矩阵化算子（定义见 [Louizos 和 Welling, 2017]），即将多维张量的形状转换为矩阵。</escape>

附录B。预测不确定性实验的数据

对于预测不确定性实验，我们从以下高斯分布生成数据：

$$
\begin{aligned}
G_1 &\sim \mathcal{N}\left(\begin{pmatrix}-8\\-8\end{pmatrix}, \begin{pmatrix}6 & -1\\-1 & 3.5\end{pmatrix}\right), \\
G_2 &\sim \mathcal{N}\left(\begin{pmatrix}6\\6\end{pmatrix}, \begin{pmatrix}0 & 3\\3 & 0\end{pmatrix}\right), \\
G_3 &\sim \mathcal{N}\left(\begin{pmatrix}-7\\8\end{pmatrix}, \begin{pmatrix}-3 & 4\\-5 & 1\end{pmatrix}\right), \\
G_4 &\sim \mathcal{N}\left(\begin{pmatrix}8\\-8\end{pmatrix}, \begin{pmatrix}0 & 5\\4 & 2\end{pmatrix}\right), \\
G_5 &\sim \mathcal{N}\left(\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0 & 9\\9 & 0\end{pmatrix}\right).
\end{aligned}
$$