Neural Network and Deep Learning-PJ 1

李明泽 22307140089

摘要

本报告介绍了使用 MLP/CNN 在 MNIST 数据集实现识别手写数字的过程。经过对比实验，MLP 的两个隐藏层应用 ReLU 函数，输出层应用 softmax 函数，优化器采用带动量的随机梯度下降算法，同时还应用了一些常用技术，如学习率下降、L2 正则化等。此外，我们还实现了一个简单的 CNN。在测试集上对两个模型进行评估时，可视化其结果来评估有效性。实验结果显示：MLP 最优模型的训练集准确率为 97.93%，测试集准确率为 97.46% 。简单 CNN 的的训练集准确率为 98.00%，测试集准确率为 97.70%。最后我们讨论了未来的改进方向。

1 Introduction

在本项目中，我们构建神经网络 (MLP、CNN) 来对图像进行识别。我们将在 MNIST 数据集上对网络进行训练，并根据分类的准确性对其性能进行评估。

项目代码：GitHub Repo MLP、CNN 模型权重：百度网盘

2 Dataset

MNIST 数据集是一个由美国国家标准与技术研究院整理的包含 70,000 张 28×28 像素灰度手写数字图像的数据集，其中 60,000 张用于训练，10,000 张用于测试，每张图像对应一个 0 到 9 的数字标签。它简单且多样，常被用来训练和测试分类算法，是学习和研究图像识别的经典基准数据集。

图 1: MNIST Dataset Example

3 Data transformation

在读入数据集后，对原始图像施加一系列微小的变换（包括平移、旋转、缩放），生成与原始图像具有一定差异但标签保持一致的新样本，从而提升网络的泛化能力。

4 Model

4.1 Model Architecture

经典的神经网络包含三个主要组成部分：输入层、隐藏层和输出层（图 2）。而卷积神经网络，在传统的多层神经网络基础上，全连接层前面加入了卷积层、和池化层（图 3），使得实际应用场景中能够构建更加深层、功能更强大的网络。

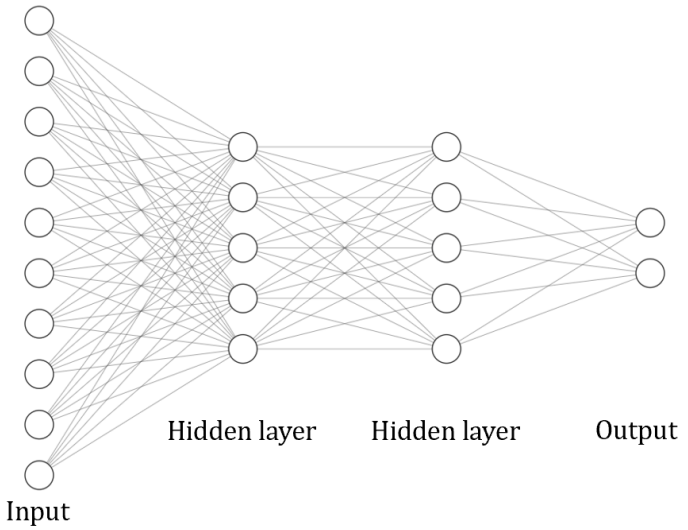


图 2: 神经网络架构图

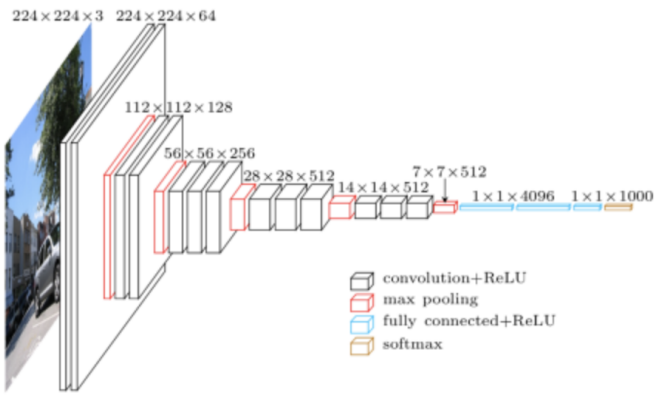


图 3: VGG-16 架构图

4.2 Activation Function

我们实现了几种常用的激活函数。

4.2.1 ReLU

Forward: $f(x) = \max(0, x)$ Backward: $\frac{\partial f}{\partial x} = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$

4.2.2 Sigmoid

Forward: $\sigma(x) = \frac{1}{1+e^{-x}}$ Backward: $\frac{\partial f}{\partial x} = \sigma(x) \cdot (1 - \sigma(x))$

4.2.3 Tanh

Forward: $\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$ Backward: $\frac{\partial f}{\partial x} = 1 - \tanh^2(x)$

4.3 Loss Function

损失函数帮助我们量化模型预测值与实际值之间的差异。在此，我们将实现交叉熵损失函数。多分类任务中，交叉熵损失函数通常与 softmax 函数结合使用。我们在网络的最后使用 softmax 函数，将模型的输出转换为概率分布，然后交叉熵损失函数计算该概率分布与真实分布之间的差异。

soft 函数定义如下：

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}$$

计算出 softmax 概率后，就可以使用真实标签和预测概率来计算交叉熵损失。交叉熵损失的定义如下：

$$\text{Cross-entropy loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{s=1}^{C} y_{i,s}\log(f(s)_i)$$

N：样本数量,C: 类别总量，$f(s)_i$：第 i 个样本属于类别 s 的预测概率（即 softmax 输出）,y_i, s：第 i 个样本在类别 s 的真实标签

4.4 Regularization Method

在原始的损失函数基础上，L2 正则化添加了一个正则化项，其数学表达式为：

$$L_{\text{regularized}} = L_{\text{original}} + \lambda\sum_{i=1}^{n} w_i^2$$

5 Training

5.1 Optimizer

5.1.1 Vanilla SGD

$$w^{t+1} = w^t - \alpha_t \nabla f(w^t)$$

其中 α_t 是 learning rate。

5.1.2 SGD with Momentum

$$w^{t+1} = w^t - \alpha_t \nabla f(w^t) + \beta_t(w^t - w^{t-1})$$

其中 α_t 是 learning rate,β_t 是 momentum, 常用值是 0.9，代码里也采用 0.9。

5.1.3 RMSProp and Adam

原理较为复杂，具体实现见实验代码。

5.2 Experiments

5.2.1 Experiments of different activation function

5.2.2 Experiments of different activation function

在 activation_experiment.py 中，我们想要观察不同激活函数下模型的性能，训练配置如表 1，实验结果如图 4、5。

表 1: 实验参数配置

网络结构	**训练配置**
隐藏层: [128,64] 激活函数: ReLU L2_reg: 0.001	epoch: 10 batch_size: 128 学习率: 0.005 update_rule:SGDMomentum deacy: 0.001

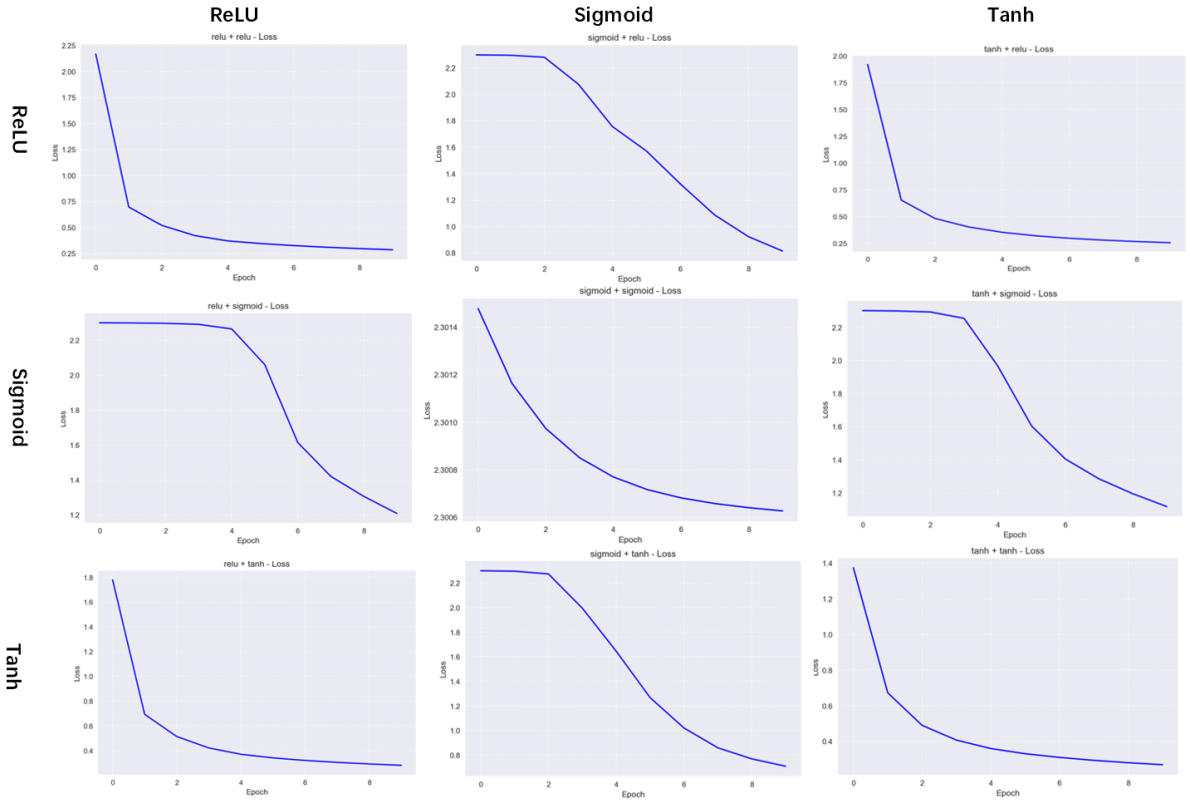


图 4: 不同激活函数下的 loss

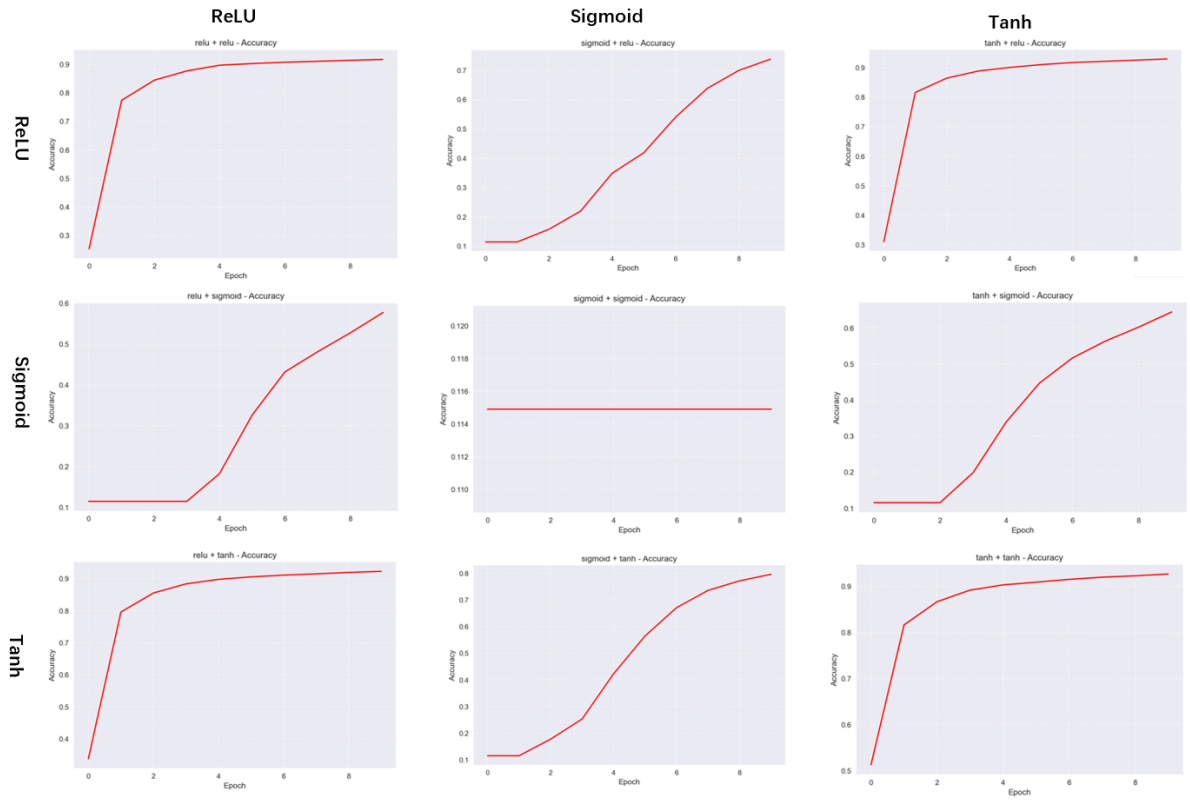


图 5: 不同激活函数下的 accuracy

表现比较好的四个组是 ['ReLU','ReLU'], ['ReLU','Tanh'],['Tanh','ReLU'],['Tanh','Tanh']。但大量文献证实 ReLU 函数在避免梯度消失、提升深层网络训练效率方面的优势显著，因此，我们决定在激活函数的选择上采用 “ReLU + ReLU” 组合。

5.2.3 Hyperparameters Tuning

我们采用网格搜索方法来寻找最佳超参数: 正则化参数（$l2\_reg\_weight$）、学习率（$learning\_rate$）和隐藏层神经元数量（$n\_neurons$）。

第一阶段，我们首先设置 epoch=5, 进行大范围的参数粗调，观察结果趋势。设置学习率的粗调范围为 $[0.0001, 0.001, 0.01]$, 设置隐藏层神经元数量范围为 $[[64, 64], [128, 64]]$, 设置正则化参数范围为 $[0.001, 0.01, 0.1]$。实验结果（表 2）表明，相较于学习率和正则化参数，隐藏层维度对模型准确度的影响较小。当学习率太小或者正则化参数太大时，模型的学习能力很差：训练集准确率仅为 11.17%，测试集准确率仅为 11.49%。

表 2: 不同超参数下训练集与测试集准确率（第一阶段粗测）

learning_rate	n_hidden	l2_reg_weight	Training_accuracy	Test_accuracy
0.0001	[64,64]	0.001	0.1117	0.1149
0.0001	[64,64]	0.01	0.1117	0.1149
0.0001	[64,64]	0.1	0.1117	0.1149
0.0001	[128,64]	0.001	0.1117	0.1149
0.0001	[128,64]	0.01	0.1117	0.1149
0.0001	[128,64]	0.1	0.1117	0.1149
0.001	[64,64]	0.001	0.6780	0.7630
0.001	[64,64]	0.01	0.3986	0.5822
0.001	[64,64]	0.1	0.1117	0.1149
0.001	[128,64]	0.001	0.7083	0.7891
0.001	[128,64]	0.01	0.6281	0.7428
0.001	[128,64]	0.1	0.1117	0.1149
0.01	[64,64]	0.001	0.9515	0.9640
0.01	[64,64]	0.01	0.8889	0.9333
0.01	[64,64]	0.1	0.1117	0.1149
0.01	[128,64]	0.001	0.9606	0.9718
0.01	[128,64]	0.01	0.9001	0.9414
0.01	[128,64]	0.1	0.1177	0.1149

在此基础上，进入第二阶段。我们选择在第一阶段表现较好的参数组合，适当增大 epoch 值，进行小范围的精细调参。根据第一阶段结果，缩小搜索范围，将 epoch 设置为 10，学习率 (*learning_rate*) 为 [0.01,0.05]，正则化系数 (*l2_reg_weight*) 为 [1e-4,1e-3]，隐藏层神经元数量固定为 [128,63]，继续进一步观察模型性能。实验结果见表 3, 可以看到，最好的测试集准确率达到了 97.73%。

表 3: 不同超参数下训练集与测试集准确率（第二阶段精调）

learning_rates	l2_weights	Training_accuracy	Test_accuracy
0.01	0.0001	0.9790	0.9746
0.01	0.001	0.9735	0.9764
0.05	0.0001	0.9841	0.9773
0.05	0.001	0.9650	0.9721

所以我们最好的参数配置是：

表 4: 最优超参数

learning_rate	l2_reg_weights	n_nhidden
0.05	0.0001	[128,64]

5.2.4 Experiments of different optimizer

在得到最优超参数后，我们想看看模型在不同 optimizer 下的表现（图 6、7）。与文献调研的结果一致，SGD+Momentum 的表现最好。

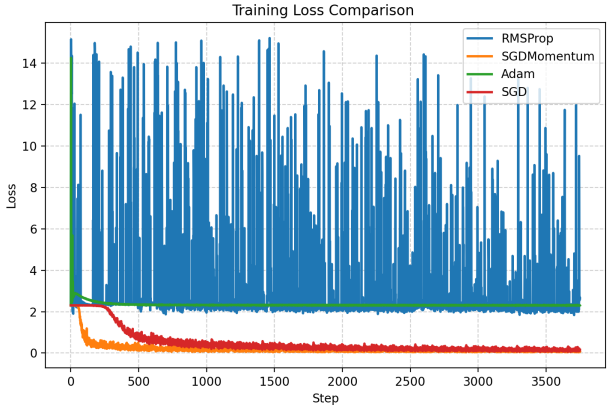


图 6: Training loss of activations

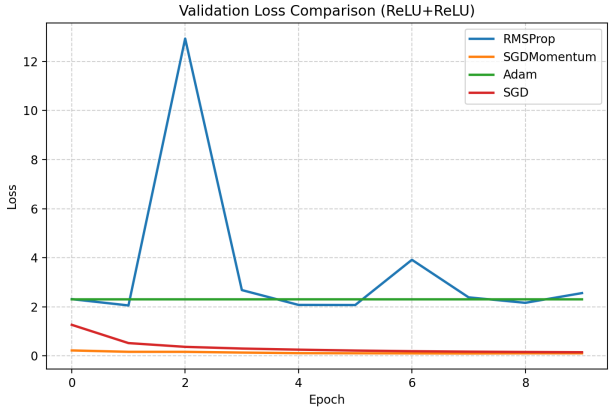
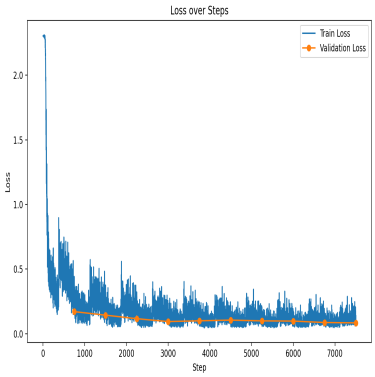


图 7: Validation loss of optimizers

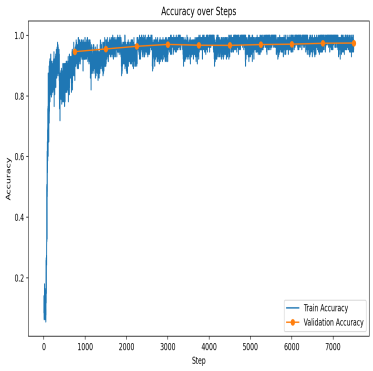
6 Results

6.1 MLP Results

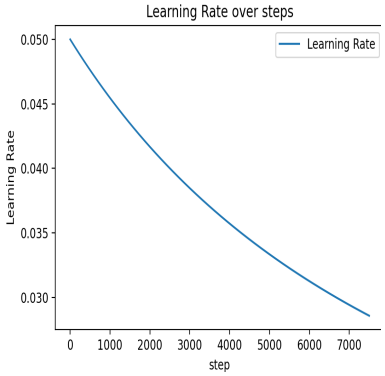
为了直观呈现最优模型的性能，我们设置 epoch=10，绘制了一些曲线，具体包括：loss 曲线、accuracy 曲线，learning rate 曲线以及权重分布图。



(a) 训练损失曲线



(b) 分类准确率曲线



(c) 学习率调度曲线

可以看到，模型在训练集上的 acc: 0.9793, loss: 0.1370，在测试集上的 acc: 0.9746, loss: 0.0839。

为了深入了解模型的学习过程，我们对隐藏层的权重进行了可视化，如图 9-12 所示。

我们观察到第一隐藏层的权重确实显示出一些模式，其中一部分符合手写数字的形状。这是因为第一隐藏层直接与输入层相连，能够捕捉输入数据的原始特征。

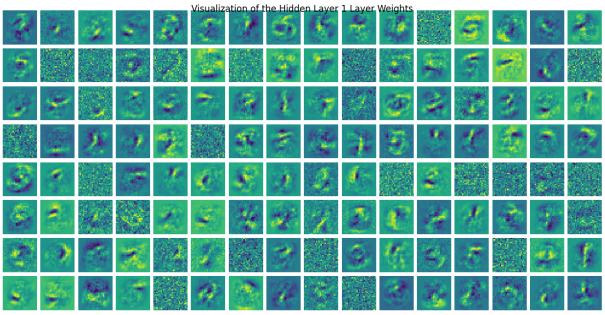


图 9: Visualization of the first layer weights

我们还绘制了第一隐藏层偏置的直方图，我们看到大多数偏置接近于零，这是符合预期的。

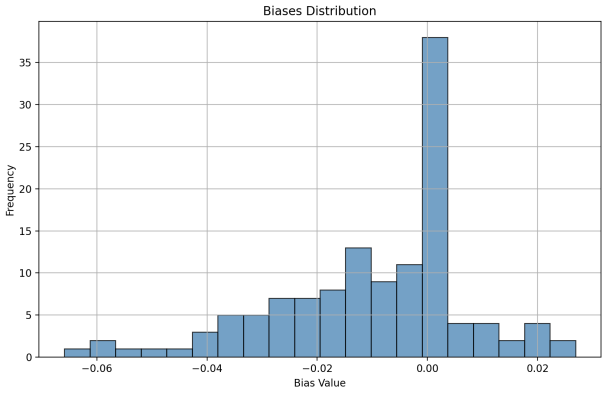


图 10: Visualization of the first layer bias

中间隐藏层的权重更为抽象，更难以解释，此时肉眼已经分辨不出图片代表什么数字。

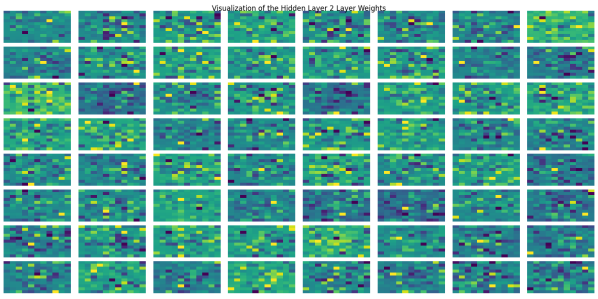


图 11: Visualization of the second layer weights

中间隐藏层的偏置也集中于零，但是偏置的分布范围变大了。

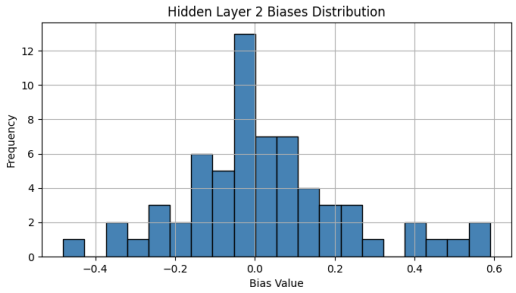


图 12: Visualization of the second layer bias

我们进一步绘制了子矩阵热力图和权重分布直方图（图 13、14）。热力图中出现了一部分色块和垂直/水平条纹，表明模型学习到了一部分特征。分布图中无死亡神经元，说明反向传播有效。第二层正负权重比例为 46.4%，说明网络在学习过程中没有完全偏向某一方向，而是较均匀地分配权重。

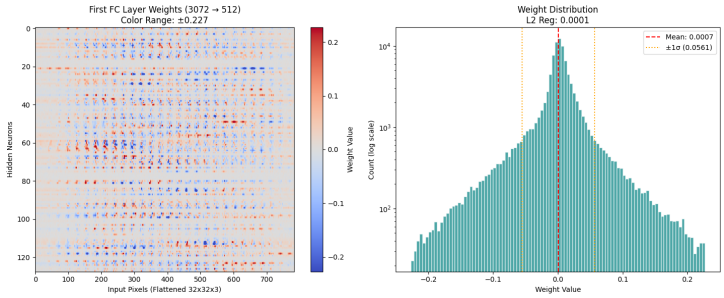


图 13: First layer distribution

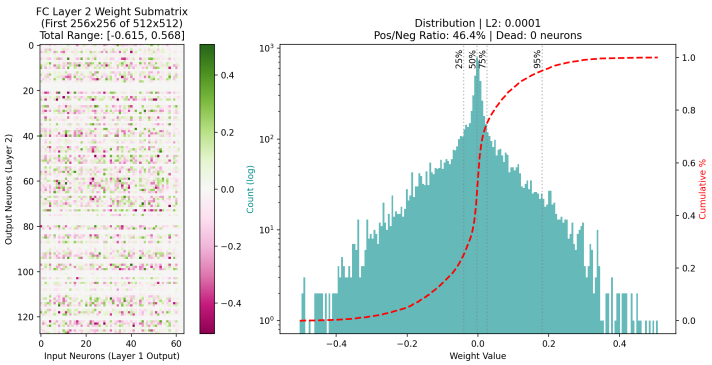


图 14: Second layer distribution

6.2 CNN Results

现在我们想看看更复杂的模型——卷积神经网络 (CNN) 的表现如何。我们实现了一个结构简单的 CNN: conv - relu - pool - affine - relu - affine - softmax。配置如下：卷积层包含 30 个卷积核，每个核的尺寸设定为 5×5，采用零填充（padding=0）和单位步

长（stride=1）；全连接层的隐藏单元数设为 100；学习率初始化为 0.001；总训练轮数设定为 3，并采用 Adam 优化器进行参数更新。根据实验结果，epoch=1 时，Train acc: 0.2140, Test acc: 0.1770。但是在第二个 epoch 时，模型的性能出现了显著提升，Train acc: 0.9640, Test acc: 0.9650。

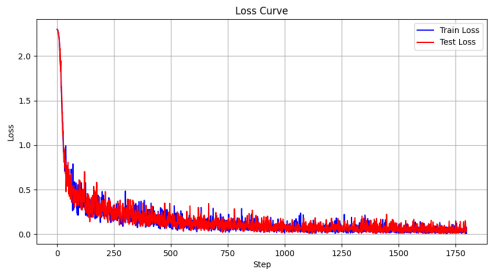


图 15: CNN 训练损失曲线

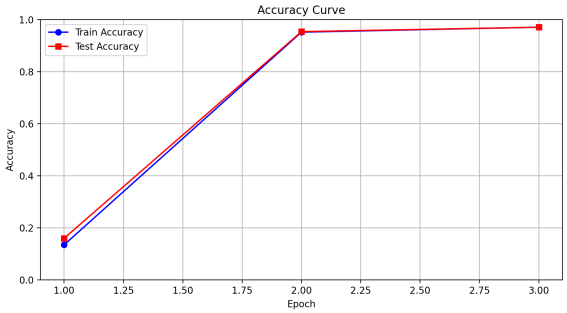


图 16: CNN 训练准确率曲线

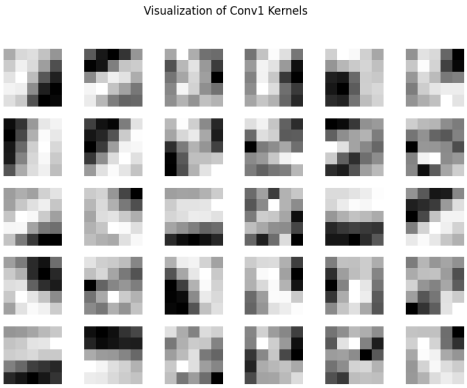


图 17: Visualization of Conv1 Kernels

我们可视化卷积层权重，可以看到卷积核提取的特征非常抽象。

7 Discussion

7.1 Model Architecture

由于 MNIST 数据集不是那么复杂，所以简单的网络就有很好的性能。但是当数据集变得复杂时，我们要考虑增加隐藏层数量，或者选择更加复杂的神经网络。

7.2 Hyperparameters Tuning

我们寻找超参数时仅仅是在一些常用的超参数范围下进行搜索，可能的改进是扩展超参数的搜索范围，也可以考虑对 batch_size,momentum 等参数也展开搜索。