机器学习数学导引 · 程序作业 · 上机报告

2100010891 吴卓轩　　2400921066 何冠聪　　2100010660 陈秋阳

摘要

本报告旨在研究二元及多元模加法运算的学习问题，并探讨不同因素对 Grokking 现象的影响。实验采用深度学习模型，特别是 Transformer、多层感知机（MLP）和长短期记忆网络（LSTM），结合多种优化器和正则化技术，对模加法运算进行学习。实验结果表明，在不同网络架构下，随着数据分数 α 的变化，模型大多表现出明显的 Grokking 现象，即在训练初期过拟合后，验证集的精度突然显著提升，最终实现完美泛化。此外，研究还发现，不同优化器和正则化技术对 Grokking 现象有显著影响，其中 AdamW 优化器结合适当的权重衰减和 Dropout 率表现出较强的泛化能力和鲁棒性。同时，对于多元模加法运算，随着求和项数量 K 的增加，问题的复杂度和难度逐渐上升，需要更多的训练轮次才能达到理想的性能水平。最后，本报告对 Grokking 现象的机制、影响因素以及未来研究方向进行了讨论，为理解深度学习的本质和开发更高效、更泛化的神经网络模型提供了新的视角和思路。

问题背景

考虑学习二元模加法运算：

$$(x, y) \mapsto x + y \mod p, \qquad \forall x, y \in \mathbb{Z}_p. \tag{1}$$

有两个因素会影响这个问题的可学习性：

1. 素数 p：更大的素数意味着更难的问题。
2. 数据分数：

$$\alpha = \frac{\#\text{训练数据}}{\#\text{全部数据}}, \tag{2}$$

其中 $\#\text{全部数据} = p^2$.

实验目的

本次上机实验旨在研究二元模加法运算的学习问题。为解决此问题，我们采用了深度学习模型，特别是 Transformer 模型，并深入探索了不同优化器和网络结构对学习效果的具体影响。

实验环境与工具

实验所需的编程环境及工具如下：

- 编程语言：Python
- 深度学习框架：PyTorch

实验方法与步骤

数据准备

实验数据集通过自定义函数 `collect_data_mod_p(p, K)` 生成，其中 p 代表模素数，K 代表加数的数量。该函数首先生成包含 K 个范围在 $[0, p-1]$ 内的列表，随后计算这些列表的笛卡尔积，得到所有可能的加数组合。每个组合被编码为 one-hot 向量，并计算其和（模 p）作为标签。使用 `split_data` 函数，根据 `alpha` 参数设定的比例，将数据集划分为训练集和验证集。

模型构建

根据命令行参数 `modelType` 的设定，选择相应的模型架构进行构建，包括多层感知机（MLP）、Transformer 和长短期记忆网络（LSTM）。在 `model.py` 文件中，定义了这三种模型的类，均继承自 `torch.nn.Module`，并实现了 `forward` 方法以定义前向传播过程。在本实验中，我们主要使用 Transformer 模型，其结构包括嵌入层、若干编码器块（未使用解码器块）以及输出层。嵌入层的维度、编码器块的数量、注意力头的数量等参数均可根据实验需求进行配置。

模型训练

模型训练过程在 `train.py` 文件的 `train` 函数中实现。该函数在每个 epoch 中遍历训练集，计算损失和准确性，并更新模型参数。在训练过程中，我们采用了 `torch.optim.AdamW` 优化器和 `torch.optim.lr_scheduler.LinearLR` 学习率调度器，以优化训练效果。此外，我们还探索了 SGD、Adam、RMSprop 等多种优化器，以及带有 Nesterov 动量的 SGD、带有权重衰减的 AdamW 等变体，并研究了不同学习率和权重衰减率对学习效果的具体影响。

模型评估

模型评估过程在 `train.py` 文件的 `evaluate` 函数中实现，该函数在验证集上计算并返回模型的准确性。在模型训练完成后，我们使用验证集对模型性能进行评估，评估指标包括准确性和损失等。同时，我们利用 `util.py` 文件中的 `plot` 函数，绘制了训练和验证准确性随 epoch 变化的曲线图，以可视化模型在训练过程中的表现。

1 Grokking 现象重现及数据分数 α 对其之影响

1.1 任务要求

使用 AdamW 优化器为 transformer 模型重现 Grokking 现象，并检查数据分数 α 如何影响 Grokking 现象。

1.2 参数配置

见附录 (6.1)、(6.4)。

1.3 实验结果

图 (1) 是取 $p = 97$ 时，数据分数 α 分别取 $0.3, 0.4, 0.5$ 时的结果。

1.4 结果分析

数据分数 $\alpha = 0.3$ 的情况 当数据分数 α 设置为 0.3 时，训练精度在 Epoch=100 时已基本达到 100%。然而，验证精度的显著提升则发生在 Epoch=1300 之后，并且在 Epoch=2000 之后，验证精度也基本达到了 100%。

数据分数 $\alpha = 0.4$ 的情况 在数据分数 α 为 0.4 的条件下，训练精度同样在 Epoch=100 时迅速达到 100%。但验证精度的明显提升则延迟到了 Epoch=400 之后。值得注意的是，在 Epoch=600 左右，验证精度出现了一个短暂的小幅下降，随后在 Epoch=800 之后，验证精度接近 100%，但表现出一定的不稳定性，偶尔会有下降的趋势。

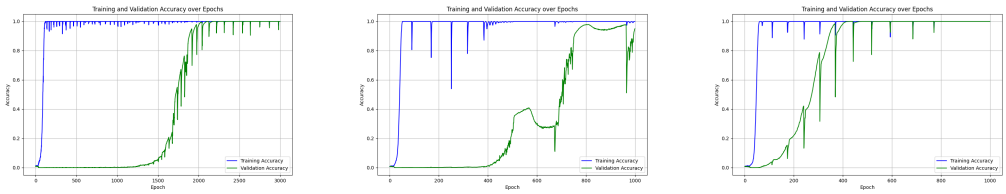

Figure 1: 数据分数 $\alpha = 0.3$（左），0.4（中），0.5（右）时，在 Transformer 架构下，训练与验证精确度变化情况

数据分数 $\alpha = 0.5$ 的情况 当数据分数 α 提升至 0.5 时，训练精度在 Epoch=70 时就已达到了 100%。与此同时，验证精度也在 Epoch=400 之后迅速达到了 100%。

Grokking 现象的重现 以上结果充分展示了 Grokking 现象，即在模型过拟合后的很长一段时间内，验证集的精度突然出现了显著的上升。

数据分数 α 对训练难度的影响 随着数据分数 α 的降低，我们发现训练难度显著增加。这体现在训练过程的不稳定性增加，以及达到较高精度的训练周期数也随之增加。特别是当数据分数 α 为 0.3 时，我们在实际训练过程中遇到了模型不收敛的情况，经过多次调整后才获得了最终的结果。这也进一步证实了数据分数 α 的降低会导致训练难度的上升。

训练过程中的精度波动 观察精度图像，我们发现每间隔一定的训练周期，训练精度会出现极其短暂的突降。这可能与具体的训练方法以及机器学习问题的非凸性有关。这种波动可能反映了模型在训练过程中的探索过程，以及在不同局部最优解之间的跳跃。

2 MLP 和 LSTM 网络架构中的 Grokking 现象

2.1 任务要求

研究其他网络结构（如 MLP 和 LSTM）的 Grokking 现象。

2.2 参数配置

见附录 (6.2)、(6.3)、(6.4)。

2.3 实验结果

图 (2) 是取 $p = 97$ 时，在 MLP 架构下，数据分数 α 分别取 $0.4, 0.5, 0.6$ 时的结果。

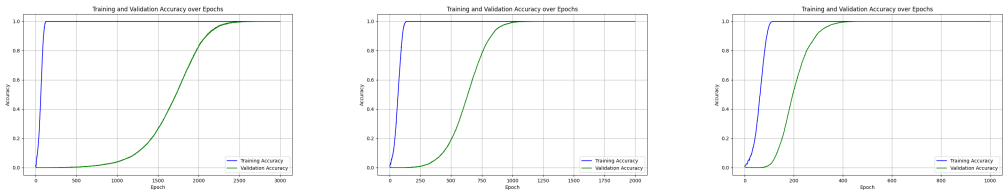

Figure 2: 数据分数 $\alpha = 0.4$（左），0.5（中），0.6（右）时，MLP 架构的精度变化

图 (3) 是取 $p = 97$ 时，在 LSTM 架构下，数据分数 α 分别取 $0.3, 0.4, 0.5$ 时的结果。

2.4 结果分析

MLP 架构

- $\alpha = 0.4$：100 周期后训练精度近 100%，2500 周期后测试精度近 100%。

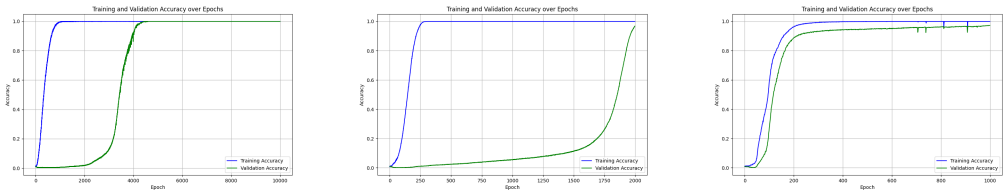

Figure 3: 数据分数 $\alpha = 0.3$（左），0.4（中），0.5（右）时，LSTM 架构的精度变化

- $\alpha = 0.5$：125 周期后训练精度近 100%，1000 周期后测试精度近 100%。
- $\alpha = 0.6$：110 周期后训练精度近 100%，400 周期后测试精度近 100%。

LSTM 架构

- $\alpha = 0.3$：1000 周期后训练精度近 100%，4500 周期后测试精度近 100%。
- $\alpha = 0.4$：300 周期后训练精度近 100%，2000 周期后测试精度近 100%。
- $\alpha = 0.5$：200 周期后训练精度近 100%，但测试精度未出现明显 Grokking 现象。

值得注意的是，与 Transformer 架构相比，MLP 和 LSTM 架构的训练与测试精度曲线更为平滑。这可能是由于不同网络架构的内在特性所导致的。此外，除了 $\alpha = 0.5$ 时的 LSTM 架构外，其他所有架构在不同 α 取值下均表现出了明显的 Grokking 现象。

3 不同优化器和正则化技术对 Grokking 现象的影响

3.1 任务要求

分析不同优化器（例如具有不同学习率和批大小的 SGD、RMSprop、基于 momentum 的优化器等）和正则化技术（例如，dropout、权重衰减）对 Grokking 现象的影响。

3.2 实验结果

在实验中，我们采用了如附录 (6.5) 的八种不同的训练配置，并系统地探究了取 $p = 97$，数据分数 α 分别取 $0.1, 0.2, \ldots, 0.9$ 时，各配置下的算法表现。

实验结果如图 (4) 所示，图 (5) 和图 (6) 分别展示了不同优化器和正则化技术配置下的算法收敛速度和预测精确度。

3.3 结果分析

1. 对于**采用 Nesterov 动量的 SGD 训练，无 Dropout** 的配置，在任何选取的数据分数下均未观察到收敛现象。这表明该配置下的模型难以从给定数据中学习到有效的特征。
2. 对于 **RMSprop 训练，无 Dropout**、**Adam 训练，无 Dropout** 和**小批量 Adam 训练，无 Dropout** 的配置，在数据分数 $\alpha \geq 0.7$ 时才有较好的收敛表现。其中，**Adam 训练，无 Dropout** 和**小批量 Adam 训练，无 Dropout** 的迭代次数相对较少，表明这些配置下的模型能够更快地找到最优解。
3. 对于 **Adam 训练，Dropout 率为 0.1**、**AdamW 训练，权重衰减为 0.1，无 Dropout**、**AdamW 训练，权重衰减为 0.1，Dropout 率为 0.1** 和 **AdamW 训练，权重衰减为 0.5，无 Dropout** 的配置，即使对于较小的 α 值也能获得较好的收敛结果。特别是 **AdamW 训练，权重衰减为 0.5，无 Dropout** 的配置，甚至可以使 $\alpha = 0.3$ 的情形收敛。这表明这些配置下的模型具有更强的泛化能力和鲁棒性。
4. 观察收敛速度比较图 (5)，在达到 99% 的收敛要求时所需的迭代周期数，关于数据分数大致呈现“U”型曲线。即过低的数据分数由于训练数据量较少导致收敛缓慢，而过高的数据分数可能由于训练数据量过多导致训练缓慢。然而，该现象在 **AdamW 训练，权重衰减为 0.1，无 Dropout** 的配置下不明显，即使取较高的数据分数 $\alpha = 0.9$，也未见迭代周期数显著上升。

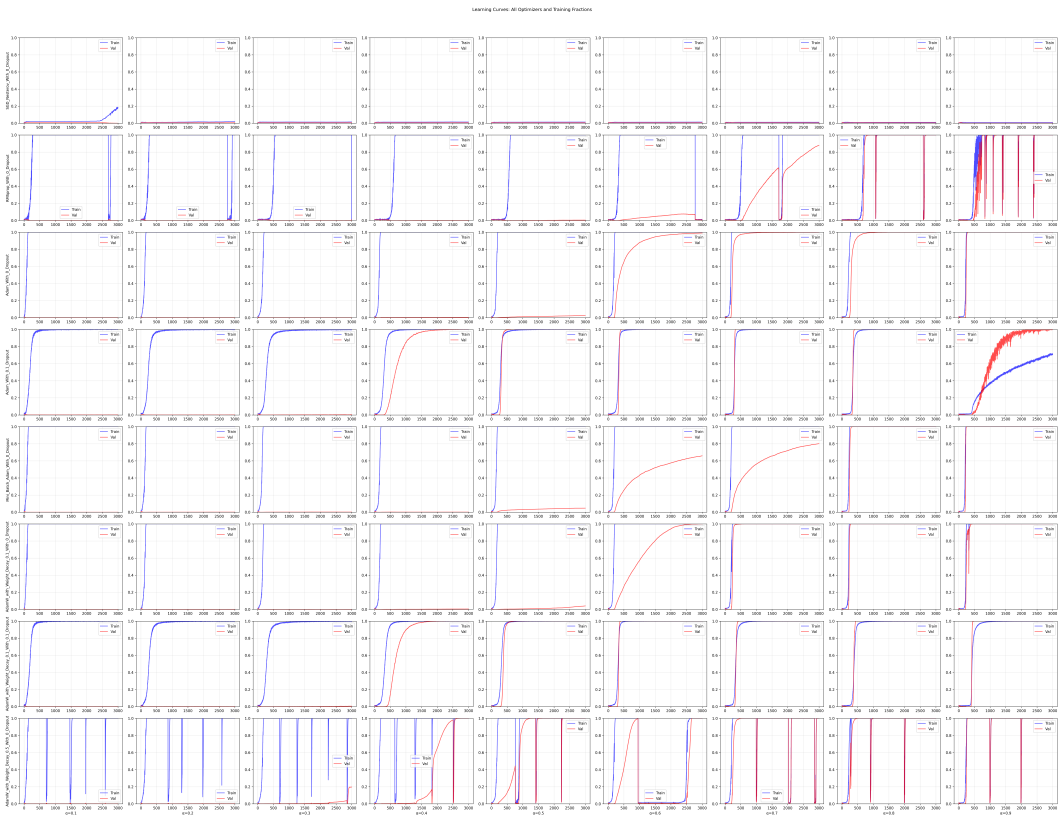

Figure 4: 不同数据分数、优化器和正则化技术下的学习曲线

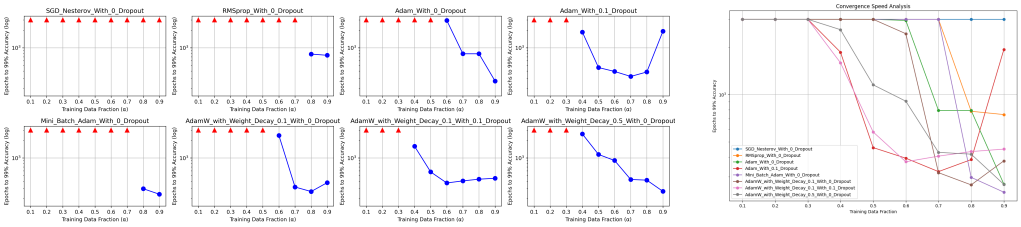

Figure 5: 不同优化器和正则化技术下的收敛速度比较

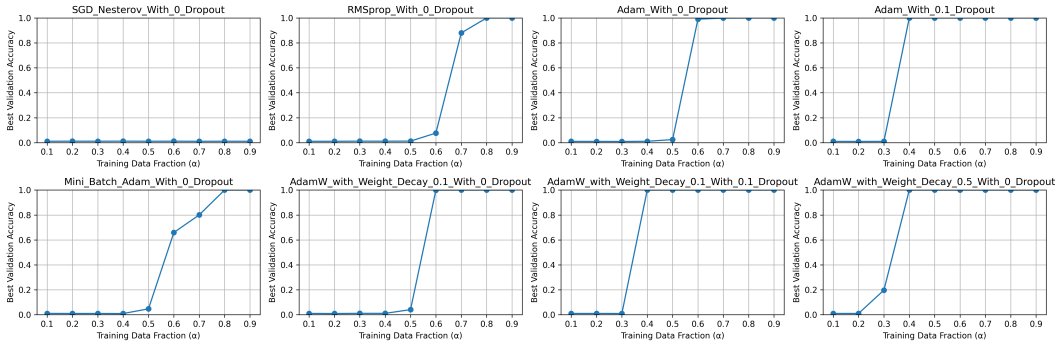

Figure 6: 不同优化器和正则化技术下的预测精确度比较

5. 观察预测精确度比较图 (6)，无论选取何种配置，在一定的最大迭代周期数下，最好的预测精度随数据分数 α 均呈现指数上升到 100% 的趋势。这可能是由于当数据分数 α 上升时，训练集中已知的信息越多，问题的难度在某种意义上呈指数级下降。

4 多元模加法项数对 Grokking 现象的影响

4.1 任务要求

考虑下面更复杂的问题

$$(x_1, x_2, \cdots, x_K) \mapsto \left(\sum_{k=1}^{K} x_k\right) \mod p, \tag{3}$$

其中 $x_k \in \{0, 1, \cdots, p-1\}$.

研究求和项数量 K 对 Grokking 现象的影响。

4.2 理论分析

在实际实现中，当 $K \geq 2$ 时，输入数据组合数达 p^K，随 K 增大而指数级增加，增加了问题复杂性和计算成本。为平衡实验有效性和时间成本，我们设定 $K = 2, 3, 4$ 并选较小质数 $p = 19$，在合理时间内完成实验，同时保持问题复杂性，有效研究 K 对 Grokking 现象的影响。

4.3 实验结果

在 MLP 模型和 AdamW 优化器下（详见附录 (6.6)），我们用以下参数进行了实验：

1. 当 $p = 19$，$K = 2$ 时，数据分数 $\alpha = 0.9$，批处理大小设置为 2。
2. 当 $p = 19$，$K = 3$ 时，数据分数 $\alpha = 0.4$，批处理大小设置为 8。
3. 当 $p = 19$，$K = 4$ 时，数据分数 $\alpha = 0.1$，批处理大小设置为 32。

如此选择参数是为了更好的收敛性考虑。

实验结果如图 (7) 所示。

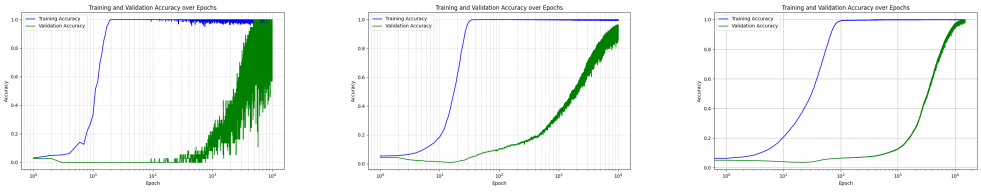

Figure 7: $K = 2$（左），$K = 3$（中），$K = 4$（右）时的学习曲线

4.4 结果分析

1. 在设置了横轴为对数坐标后，对于 $K = 2, 3, 4$ 的情况，我们都可以观察到明显的 Grokking 现象。值得注意的是，与之前未设置对数坐标时的分析相比，此时观察到的 Grokking 现象实际上表现得更为微弱。
2. 随着 K 的增大，训练集和验证集达到较高精确度的时间均有所推迟。这表明问题的复杂度和难度在逐渐上升，需要更多的训练轮次来使模型达到理想的性能水平。这与理论分析相一致。
3. 在训练中期，当训练集精度迅速上升时，验证集精度却会下降到接近 0 的水平。这一现象与文献 [2] 中提到的验证集损失函数值先增后减的趋势相吻合。

5 Grokking 现象的解释

5.1 Grokking 现象概述

Grokking 现象是指在训练神经网络处理由小型算法生成的数据集时观察到的一种独特行为，最初由论文 [5] 提出。该现象表现为，尽管网络在训练初期达到过拟合，其验证准确率会在某时突然提升，最终实现完美泛化。这与传统现象截然不同。

5.2 Grokking 现象的机制

数据集规模和特性 Grokking 现象在小型算法生成的数据集上尤为明显，这些数据集通常包含一系列离散的符号和运算规则。神经网络需要学习这些符号之间的关系和运算规则以准确预测新的运算结果。当数据集规模较小时，Grokking 现象尤为显著，因为此时网络需要更多的优化步骤来找到泛化解 [5]。

懒惰训练动态与特征学习阶段 研究人员提出，Grokking 现象可能源于神经网络从懒惰训练动态向丰富特征学习阶段的过渡 [3]。在懒惰训练阶段，神经网络主要由神经正切核（Neural Tangent Kernel, NTK）主导，能够迅速降低训练损失，但测试损失仍然较高。随着训练的进行，网络逐渐过渡到特征学习阶段，开始提取有用的特征，并表现出更强的泛化能力。

LU 机制与表示学习 Liu 等人通过分析神经网络的损失景观，提出了 LU 机制来解释 Grokking 现象 [4]。在训练过程中，训练损失随模型权重范数的增加而减少（“L” 型曲线）；而测试损失则随权重范数的增加先增加后减少（“U” 型曲线）。当训练损失已经很小而测试损失仍然很大时，如果模型权重范数能够继续增加并穿越某个临界点，测试损失将急剧下降，从而实现泛化。

Slingshot 机制与自适应优化器 在文献 [6] 中，作者发现了 Slingshot 机制。模型在训练过程中，最后一层的权重迅速增长，随后训练损失出现尖峰，并最终进入一个范数平稳期。这一过程循环发生，每一次循环都伴随着模型在验证集上泛化能力的显著提升。

5.3 影响 Grokking 现象的因素

正则化技术 正则化技术（如权重衰减）可以显著提高网络的数据效率，这表明正则化有助于网络更快地找到泛化解，并减少过拟合的风险 [5]。

模型大小与优化器 随着模型复杂度的增加，模型会经历从过度拟合到泛化的转变。此外，不同类型的优化器也会影响 Grokking 现象的发生。例如，自适应优化器（如 Adam）更容易观察到 Slingshot 效应和 Grokking 现象 [6]。

内在维度与特征表示 研究人员发现 Grokking 发生时，模型的最后一层激活的内在维度（Intrinsic Dimension, ID）会突然下降。这一发现表明 Grokking 可能与模型内部表示的简化有关，即模型在 Grokking 过程中找到了更加简洁且有效的特征表示方式 [1]。

5.4 未来研究方向

未来的研究可以进一步探索 Grokking 现象的机制、影响因素以及如何在实践中应用这一现象来提高模型的泛化能力。Davies 等人在文献 [2] 中指出，Grokking 和另一种称为双下降的现象共享相同的潜在学习动力，这也为未来的研究提供了新的视角。

5.5 结论

Grokking 现象揭示了神经网络在泛化过程中的复杂性和不确定性，并为我们理解深度学习的本质提供了新的视角。通过深入分析其机制和因素，我们有望为开发更高效、更泛化的神经网络模型提供新的思路和方法。

6 附录

6.1 Transformer 架构参数配置

Transformer 模型的配置如下：

- **模型参数**:
  - input_dim $= p$：输入维度为 p。
  - hidden_dim $= 128$：隐藏层维度为 128。
  - num_heads $= 4$：多头注意力机制中头的数量为 4。
  - num_layers $= 2$：Transformer 编码器的层数为 2。
  - output_dim $= p$：输出维度为 p。
  - seq_len $= K$：序列长度为 K。
- **优化器**: 使用 `torch.optim.AdamW` 优化器，配置如下：
  - 学习率设置为 1×10^{-3}。
  - betas $= (0.9, 0.98)$：用于计算一阶和二阶矩估计的指数衰减率。
  - weight_decay $= 0.1$：权重衰减率为 0.1。

6.2 MLP 架构参数配置

MLP 模型的配置如下：

- **模型参数**
  - `input_dim` $= K \times p$：输入维度为 $K \times p$。
  - `hidden_dim1` $= 256$：第一个隐藏层的维度设置为 256。
  - `hidden_dim2` $= 128$：第二个隐藏层的维度设置为 128。
  - `output_dim` $= p$：输出维度与输入数据的某一维度 p 相同。
- **优化器配置**
  - 使用 `torch.optim.AdamW` 优化器。
  - 学习率设置为 1×10^{-3}。
  - `betas` $= (0.9, 0.98)$：AdamW 优化器的 β 参数配置。
  - `weight_decay` $= 0.2$：权重衰减率设置为 0.2，用于防止模型过拟合。

6.3 LSTM 架构参数配置

LSTM 模型的配置如下：

- **模型参数**
  - `input_dim` $= p$：输入维度为 p。
  - `hidden_dim` $= 128$：隐藏层的维度设置为 128。
  - `num_layers` $= 2$：LSTM 网络的层数设置为 2。
  - `output_dim` $= p$：输出维度与输入维度 p 相同。
  - `seq_len` $= K$：序列长度设置为 K。
- **优化器配置**
  - 使用 `torch.optim.AdamW` 优化器。
  - 学习率设置为 1×10^{-3}。
  - `betas` $= (0.9, 0.99)$：AdamW 优化器的 β 参数配置。
  - `weight_decay` $= 0.6$：权重衰减率设置为 0.6，以进一步控制模型的过拟合风险。

6.4 学习率调度器参数配置

使用 `torch.optim.lr_scheduler.LinearLR` 作为学习率调度器，配置如下：

- start_factor = 0.1：学习率起始因子为 0.1，意味着初始学习率是基础学习率（1×10^{-3}）的 0.1 倍。
- end_factor = 1.0：学习率结束因子为 1.0，意味着最终学习率会恢复到基础学习率。
- total_iters = 10：总迭代次数为 10 次，学习率会在这 10 次迭代中线性地从起始因子增加到结束因子。

6.5 八种优化器与正则化技术的参数配置

1. **采用 Nesterov 动量的 SGD 训练，无 Dropout**：我们使用了带有 Nesterov 动量的随机梯度下降（SGD）算法进行训练，未应用 Dropout 正则化技术。
2. **RMSprop 训练，无 Dropout**：我们选择了 RMSprop 优化算法进行训练，未使用 Dropout 正则化。
3. **Adam 训练，无 Dropout**：我们采用了 Adam 优化算法进行训练，未引入 Dropout 正则化。
4. **Adam 训练，Dropout 率为 0.1**：我们使用了 Adam 优化算法，并设置了 0.1 的 Dropout 率进行正则化处理。
5. **小批量 Adam 训练，无 Dropout**：我们采用了小批量数据，并使用 Adam 优化算法进行训练，未应用 Dropout 正则化。
6. **AdamW 训练，权重衰减为 0.1，无 Dropout**：我们选择了带有权重衰减（系数为 0.1）的 AdamW 优化算法进行训练，未使用 Dropout 正则化。
7. **AdamW 训练，权重衰减为 0.1，Dropout 率为 0.1**：我们使用了带有权重衰减（系数为 0.1）的 AdamW 优化算法，并设置了 0.1 的 Dropout 率，以增强模型的泛化能力。
8. **AdamW 训练，权重衰减为 0.5，无 Dropout**：我们选择了带有更高权重衰减（系数为 0.5）的 AdamW 优化算法进行训练，未使用 Dropout 正则化。

6.6 多项模加法参数配置

在实验设置中，我们采用了多层感知机（MLP）作为模型架构。该 MLP 模型具有以下配置：

1. **输入维度**：由于每个输入变量 x_k 可以取 p 个值（即 $0, 1, \ldots, p-1$），且共有 K 个这样的变量，因此输入层的大小为 $K \times p$。
2. **隐藏层**：模型包含两个隐藏层，第一层具有 256 个神经元，第二层具有 128 个神经元。
3. **输出维度**：输出层的大小为 p，对应于模 p 后的可能求和结果。

优化方面，我们使用了 AdamW 优化器，其配置如下：

1. **学习率**：设置为 1×10^{-3}。
2. **β 参数**：分别设置为 0.9 和 0.99，用于控制一阶矩估计和二阶矩估计的指数衰减率。
3. **权重衰减**：设置为 0.005，用于正则化模型参数，防止过拟合。

References

[1] Bradley CA Brown, Jordan Juravsky, Anthony L. Caterini, and Gabriel Loaiza-Ganem. Relating regularization and generalization through the intrinsic dimension of activations. In *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*, 2022.

[2] Xander Davies, Lauro Langosco, and David Krueger. Unifying grokking and double descent. In *NeurIPS ML Safety Workshop*, 2022.

[3] Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *The Twelfth International Conference on Learning Representations*, 2024.

[4] Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *The Eleventh International Conference on Learning Representations*, 2023.

[5] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

[6] Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua M. Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022.