Who is Adam?
Adam finds the NFL theory of optimizer

Zilu Zhang *
BUPT
zhangzilu@bupt.edu.cn

ChatGPT †
OpenAI
chatgpt@openai.com

Codex ‡
OpenAI
codex@openai.com

Abstract

长期以来，我们普遍认为 Adam 的泛化性质不如 SGD，然而，这一观点缺乏充分的实验支撑。本文为 Adam 与 SGD 的泛化性质实验进行了补充，指出 Adam 并不存在普遍的泛化劣势，是特定的损失曲面性质导致了泛化性差异。总而言之，本文有以下贡献：(1) 补充若干实验，支撑 Adam 与 SGD 泛化对比的实证研究；(2) 指出 Adam 并不存在泛化劣势；(3) 从 NFL 定理出发，阐述优化器选择的实质是归纳偏置，不存在最好的优化器；
Code available at `https://github.com/ZZL-2005/Who-is-adam`

1 Introduction

Adam 优化器通过统计梯度平方信息的滑动平均，实现了对近期梯度大小的信息感知，从而实现了自适应的对每个参数的学习率调整。[2] 长期以来，我们普遍认为 Adam 的泛化性质不如 SGD，然而，这一观点缺乏充分的实验支撑。

机器学习领域，在模型选择上存在 No Free Lunch 定理：不存在一个模型能在所有任务上都优于其他模型。[5] 因此，我们认为在缺乏广泛实验支撑的情况下，断言 Adam 的泛化性质不如 SGD 是不严谨的。在优化器意义上，我们相信也存在一种类似的 No Free Lunch 定理：不存在一个优化器能在所有任务上都优于其他优化器。

模型的选择暗含了我们对数据的先验假设（尽管我们往往未能明确表达出来），同样，优化器的选择也暗含了我们对损失曲面的先验假设。比如，在不考虑随机梯度噪声的情况下，Zilu 等人指出，从局部二次展开来看，MSGD 会比 SGD 更加偏好平坦极小值。[8]

总之，我们做出来如下贡献：(1) 补充大量实验，支撑 Adam 与 SGD 泛化对比的实证研究；(2) 指出 Adam 泛化性差异的根源在于损失曲面的性质，而非优化器本身，Adam 并不存在稳定的泛化劣势；(3) 尝试构建优化器选择意义上的 No Free Lunch 定理。

2 Review of Adam

Adam（Adaptive Moment Estimation）是一类带有动量与自适应步长的随机优化方法。第 t 次迭代在样本（或小批量）上得到随机梯度 g_t。Adam 同时对梯度的一阶矩（均值）与二阶原始矩（未中心化方差）做指数加权滑动平均：

$$m_t \;=\; \beta_1\, m_{t-1} \;+\; (1-\beta_1)\, g_t,$$
$$v_t \;=\; \beta_2\, v_{t-1} \;+\; (1-\beta_2)\, g_t^{\odot 2},$$

其中 $m_t, v_t \in \mathbb{R}^d$ 与 $\theta \in \mathbb{R}^d$ 等维；“$\odot$”表示按元素（逐坐标）运算，$g_t^{\odot 2}$ 表示逐元素平方；$\beta_1, \beta_2 \in [0,1)$ 为动量衰减系数（常用默认值 $\beta_1 = 0.9,\ \beta_2 = 0.999$）。

*这是唯一的人
†高价聘请的科研顾问
‡代码主体的完成者

1st Nndl homework , Beijing, China.

由于指数平均在初期存在零偏（靠近 0），Adam 采用标准的偏差校正（bias correction）：

$$\hat{m}_t \;=\; \frac{m_t}{1-\beta_1^t}, \qquad \hat{v}_t \;=\; \frac{v_t}{1-\beta_2^t}.$$

据此得到逐坐标自适应步长的更新：

$$\theta_t \;=\; \theta_{t-1} \;-\; \eta_t \,\frac{\hat{m}_t}{\sqrt{\hat{v}_t} \;+\; \varepsilon},$$

其中 $\eta_t > 0$ 为外层学习率（可常数或按计划衰减），$\varepsilon > 0$ 为数值稳定常数（常用 10^{-8}）。分母按元素取平方根与加法。

由于二阶矩是逐维度估计的，因此，Adam 可以理解为为每个参数都分配了一个自适应的学习率，从而实现了对不同参数的不同更新幅度的调整。

3 Related Work

Ashia 等人给出了部分在语言任务和视觉任务上的实验结果，并给出了一个构造性的二分类问题，使得 Adam 在此问题上远远弱于 SGD。[4] 然而，尽管他们做了大量的实验，但所有的实验结果都是基于交叉熵损失的，缺乏对其他损失函数的实验验证。此外，构造性的二分类问题，只能说明特定的损失曲面上，Adam 效果不佳，尽管他们的构造非常惊艳，但是我们认为这个结果依然是平凡的。某种意义上讲，这可以理解为优化器意义上的 No Free Lunch 定理：不存在一个优化器能在所有问题上都优于其他优化器。[5]

Nitish 等人延续了前人的工作，尝试了一种新的训练策略，即在训练的前期使用 Adam 优化器，在训练的后期切换到 SGD 优化器，这样的策略在实验意义上实现了对泛化性能的提升。[1]

还有工作从理论角度给出了优化器性质的描述，指出优化器主导了鞍点逃逸以及极小值偏好，最终影响了泛化性能。[6]

尽管有如此多的工作研究 Adam 的泛化性质，但由于 Adam 的鼎鼎大名，我隐约认为也许优化器没有太多的优劣之分，先前的研究也许只是在特定的损失曲面上，观察到了某种现象，而非普遍现象。因此，我们决定补充更多的实验，尝试验证 Adam 与 SGD 的泛化性质差异是否普遍存在。

4 Experiments

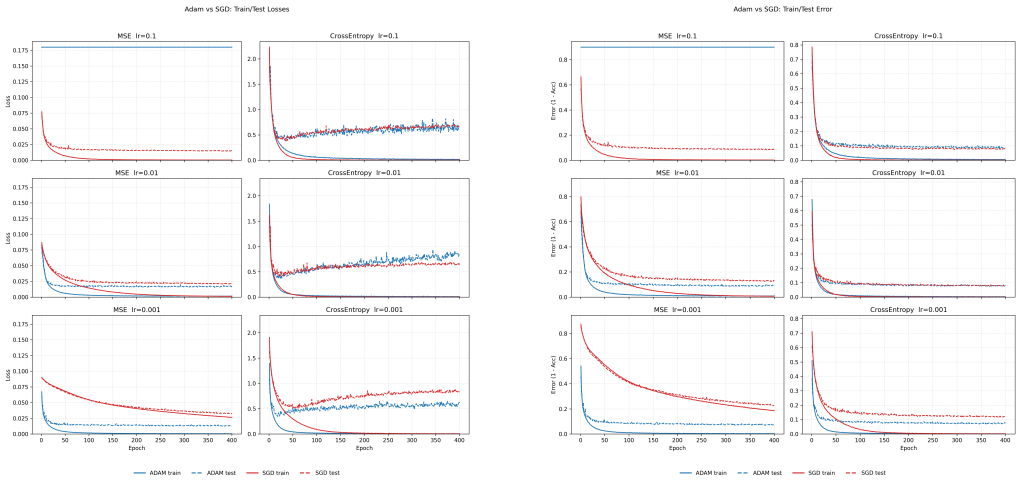
4.1 E1：Cifar-10,ResNet-18,Adam 与 SGD,MSE 与 CE, 固定学习率

本实验我们在 CIFAR-10 数据集上，使用 ResNet-18 模型，比较 Adam 与 SGD 在不同损失函数下的训练效果。我们参考了 Zilu 在上次作业 HW2 中的猜想：“也许是 loss 形式影响了损失曲面，进而影响了优化器的泛化性质”。[8] 实验设置如表1所示。

Table 1: **Experimental Setup for Adam vs. SGD Comparison on CIFAR-10 (ResNet-18)**

Component	Configuration
Dataset	CIFAR-10
Model	ResNet-18
Optimizers	Adam, SGD
Loss Functions	Mean Squared Error (MSE), Cross Entropy (CE)
Learning Rates	Fixed at $\{0.1, 0.01, 0.001\}$
Batch Size	128
Epochs	400
Weight Decay	0
Train/Test Split	50,000 / 10,000
Random Seed	42
Framework	PyTorch
Objective	作为一个初步的探索性质实验，探究 Adam 泛化性质

对于图1，我们发现 MSE lr=0.1 的子图中，Adam 的学习出现异常情况，这一现象在 Zilu 的实验中也曾出现 [7]，我们猜测这是由于 Adam 在大学习率下，第一轮更新容易引发梯度问题。比如第一个 batch 算出的梯度是 g，那么 $\hat{m_1} = g$，$\hat{v_1} = g^2$，那么更新量就是 $\frac{\eta g}{\sqrt{g^2+\epsilon}}$，数值大小当 g 远大于 ϵ 时，近似为 η，如果 η 过大，就会引发不稳定。而 SGD 的更新量是 ηg。对此我们给出的猜想解释是，g 的许多维度，其对应的梯度分量应该是比较小的，但是 Adam 放大了这些本不应该被大幅度更新的维度，因此在第一轮的“不合理更新”后，模型产生了退化，无法继续训练。不过对此现象，我们还没有进一步的实验验证与直观的可视化分析。



(a) Loss (b) Error

Figure 1: Cifar-10,ResNet-18,Adam 与 SGD,MSE 与 CE

此外，我们在实验中似乎并没有发现任何稳定的泛化差异现象。尽管模型不同，我们的损失曲线与论文 [1] 的结果也不太一样，他们的损失曲线存在明显的“跳崖”现象。我们发现，他们的实验设计中加入了学习率衰减，而我们并没有使用学习率衰减。我们猜测这可能是一个关键的因素。此外，他们的实验也并没有涉及到 ResNet-18 模型。所以，在接下来的实验中，我们打算探索学习率衰减这一因素的影响。并在模型上，尝试 ResNet-32，DenseNet-121，与论文 [1] 的设置保持一致。

另外，我们观察到 SGD 上，小学习率存在更严重的过拟合倾向。对此我的理解是，训练集所确定的损失曲面与真实分布带来的损失曲面之间是存在差异的。也许由于这种差异，会给损失曲面带来许多的高曲率极小值。而根据 [8] 中导出的 SGD 在极小值附近的收敛条件(符号约束略去)：

$$\eta\lambda_{max} \leq 2$$

当学习率较小的时候，SGD 可以“容忍”更多的高曲率极小值，也就是有更多的收敛选择，我认为正确的选择一定是少的，你能容忍的更多，就意味着你会收敛到更多不太好的极小值。

不过，以上的分析都只是猜想，缺乏进一步的实验验证。理论也并不完善，只是停留在一种自圆其说的程度。其正确性依然是未知的，需要进一步的实验与理论分析来验证。

4.2 E2：Cifar-10,ResNet-32,Adam 与 SGD,MSE 与 CE, 固定学习率

我们首先在固定学习率设置上，使用更大的 ResNet-32 模型进行实验，尝试确认是否是学习率衰减的影响，实验设置如表2所示。不过此处我们没有启用学习率衰减，一方面是因为我直到做完了这个实验，发现现象与论文 [1] 中描述的并不一致，才意识到他们可能使用了学习率衰减，并去论文中确认。第二个方面是，其实不启用学习率衰减，可以让我们探究是不是模型大小的影响。进而间接地验证学习率衰减的影响。

Table 2: **Experimental Configuration for Adam vs. SGD under Fixed Learning Rate (CIFAR-10, ResNet-18)**

Component	Configuration
Dataset	CIFAR-10
Model	ResNet-32
Optimizers	Adam, SGD
Loss Functions	Cross Entropy (CE), Mean Squared Error (MSE)
Learning Rate	Adam Fixed at 0.001, SGD Fixed at 0.1,momentum=0.9
Batch Size	128
Epochs	300
Weight Decay	0
Metrics Tracked	Train/Test Loss, Train/Test Error
Framework	PyTorch
Objective	参考上个实验的超参数设计，使用更大的模型 ResNet-32 进行实验

这个实验的结果如图2所示，我们依然没有观察到 Adam 泛化弱于 SGD 的现象。尽管如此，这里面也有一些有趣的现象值得我们去关注。总体的 acc 排序是，Adam-CE > SGD-CE > SGD-MSE > Adam-MSE。结合前面的实验结果，这更加告诉了我们，优化器的泛化性质是一个复杂的问题，我们很难简单地给出判据“MSE 下 Adam 就更好”，由此基本可以否定 Zilu 在 [8] 中提出的猜想。尽管我们没能找到某个法则，把这一切都解释清楚，但是值得肯定的是，通过实验我们获得了全新的视角，深刻地意识到了这是个极其复杂的问题，值得我们去深入研究。

同时，我们还观察到了 testLoss 的强烈振荡，这与论文中的前半段 acc 情况基本符合 [1]，不过这在我们的 ResNet-18 的实验上并没有观察到。modelsize 的变化，居然就引发了这么奇怪的现象，因为我们几乎可以认为这不是学习率或者梯度的问题，loss 曲线在 train 上是十分平滑地下降的。不过对于这个现象，我目前无法给出合理的解释。

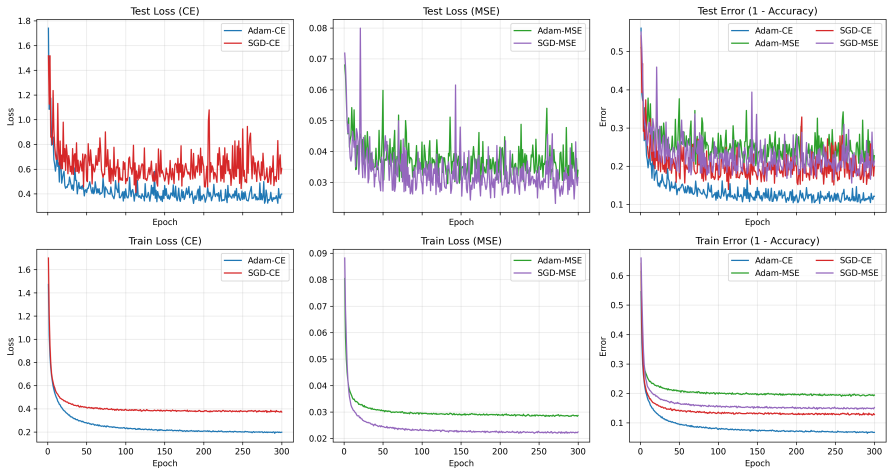


Figure 2: Cifar-10,ResNet-32,Adam 与 SGD,MSE 与 CE

4.3 E3：Cifar-10,DenseNet-121,Adam 与 SGD,MSE 与 CE, 学习率衰减

我们的实验设置与论文 [1] 中基本一致，不过他们的学习率衰减策略是机械设定的衰减模式，而我们使用的是自适应的学习率衰减策略，即当验证集指标在一段时间内没有提升时，自动降低学习率。我们选择这样的变种是有一定的考虑的，第一是避免去重复他们的实验，减轻实验成本。第二是，我们在论文 [4] 中看到，他们使用的学习率衰减策略是基于验证集指标的自适应衰减，且在他们的实验设置上，出现了 Adam 泛化弱于 SGD 的现象。因此，我们的实验设置相当于是二者的结合。实验设置如表3所示。

Table 3: **Experimental Setup for Adam vs. SGD with LR Decay on CIFAR-10 (DenseNet-121)**

Component	**Configuration**
Dataset	CIFAR-10
Model	DenseNet-121
Optimizers	Adam, SGD (momentum = 0.9)
Loss Functions	Cross Entropy (CE), Mean Squared Error (MSE)
Learning Rate	Initial: Adam = 0.001, SGD = 0.1
Scheduler	`(mode="max", factor=0.9, patience=10)`
Batch Size	128
Epochs	300
Weight Decay	0
Train/Val/Test Split	45,000 / 5,000 / 10,000
Checkpoint Interval	Every 30 epochs
Framework	PyTorch
Objective	探究学习率衰减对优化器性能的影响

然而，从实验结果上看，我们依然没有观察到 Adam 泛化弱于 SGD 的现象，如图3所示。应当强调的是，我没有对训练的任何超参数进行调优，基本直接沿用了论文的超参数设置。

如果 Adam 真的存在结构化的泛化劣势，那么在如此多的实验中，我们应该能够观察到一些蛛丝马迹。然而事实上并非如此。

这组实验中，我们观察到了 Adam 的 test loss 稳定小于 SGD。如果 Adam 真的存在普遍泛化劣势的话，那我们的这组实验，只不过是更换了一个新的学习率调度机制，就完全颠覆了 Adam 与 SGD 的泛化对比关系，这似乎也不太合理。只剩下两种可能性：第一，Adam 并不存在普遍的泛化劣势，确实就是有限的实验出现的偶然现象；第二，可能我们遗漏了某

些关键的实验设置。考虑到我暂时没有发现 2 的情况，因此我倾向于 1 的可能性。更何况，如果 Adam 真的存在普遍的泛化劣势，那我们总得从上面的实验中看出些端倪，然而事实并非如此。

此外，图中出现了一个 test 尖刺，这肯定对应了损失曲面中一个极其特殊的位置，对那个局部展开分析，也许有助于我们更好地理解损失曲面的结构。不过遗憾的是，我们没有存储对应位置的检查点。

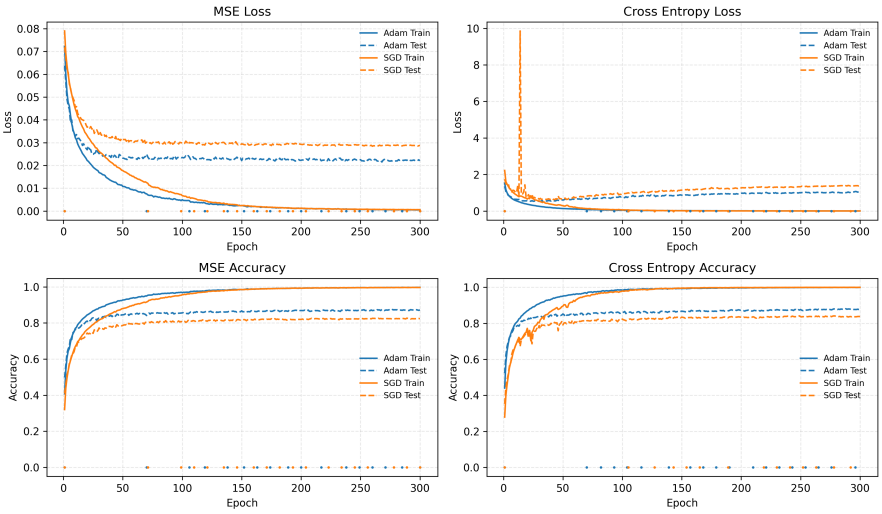


Figure 3: Cifar-10,DenseNet-121,Adam 与 SGD,MSE 与 CE

5 Analysis

5.1 NFL：没有免费午饭定理

按照计划，这里我应该深入了解 NFL 定理的相关理论，不过写到这里的时候已经 8 点了，没时间详细理解了，只能从结果上谈谈简单的认识。但我知道这是远远不够的。

首先，NFL 定理的论文 [5] 中，给出了 NFL 定理的明确定义范围。我们的研究对象是黑箱优化算法，即，我们只能访问到一个函数的输入和输出，算法 a 企图以这种有限的信息获取方式，找到函数的最值。NFL 定理指出，对于所有可能的函数 f，任意两个优化算法 a_1, a_2，在 f 上表现的平均性能是相同的。

$$\sum_f P(d_m^y \mid f, a_1, m) = \sum_f P(d_m^y \mid f, a_2, m)$$

其中 $P(d_m^y \mid f, a_1, m)$ 理解为：在函数 f 上，使用算法 a_1，经过 m 次查询后，得到的输出数据集为 d_m^y 的概率。所以这个定理形式就告诉了我们，对于一串查询结果序列，当我们把视角放在所有可能的任务上的时候，一串查询结果的“可能性”是一样的。

这里，用序列“可能性”（其实也不是，因为没归一化）并不直接作为算法表现的量化，所以上述公式并没有直接指出那句我们耳熟能详的话：“任何算法在任何任务上都不能保证优于其他算法”。我们还要再过一个性能函数。假设你定义任何一个**性能函数** $u(d_m^y)$：它衡量算法跑完后结果的“好坏”（比如最低值、平均值、最后一步结果……）。

那么算法 a 的平均表现就是：

$$\mathbb{E}_f[u(d_m^y)] = \sum_f \sum_{d_m^y} u(d_m^y)\, P(d_m^y \mid f, a, m)$$

NFL 定理告诉你：

$$\sum_f P(d_m^y \mid f, a_1, m) = \sum_f P(d_m^y \mid f, a_2, m)$$

于是乘上同样的 $u(d_m^y)$ 并求和，立刻得到：

$$\sum_f \sum_{d_m^y} u(d_m^y)\, P(d_m^y \mid f, a_1, m) = \sum_f \sum_{d_m^y} u(d_m^y)\, P(d_m^y \mid f, a_2, m)$$

即：

$$\mathbb{E}_f[u(d_m^y)]_{a_1} = \mathbb{E}_f[u(d_m^y)]_{a_2}$$

不过要注意，此处的性能函数，也就是你的评估指标，必须是与 f 无关而只与 d 有关。尽管我们没能完整严格的推导一边，但单纯从外层的梳理上看，NFL 的条件是苛刻的，我们对研究的问题，算法的定义，性能的定义，都有着清晰明确的约束。如果想要构建类似意义的优化器层面的 NFL 定理，我们也必须给出类似的严格定义与约束。

5.2 优化器选择中的 NFL

按照 NFL 原论文的观点，现在的神经网络优化器，是不能理解为定理中的黑箱优化算法的。因为神经网络优化器，能够访问到损失函数的梯度信息。

但是，我感觉 NFL 也许在这个问题上也是一样的。当我们考虑到一切可能的损失曲面的时候，任意两个优化器的平均表现也是一样的。当然了，上面这句话是显然错误的，比如你优化器 a 用梯度上升法，优化器 b 用梯度下降法，那么 a 在所有任务上都不如 b。不过我感觉这里面 NFL 的思想是好的，即不存在一个优化器能在所有任务上都优于其他优化器。我相信在构建了严格定义，若干约束性质后，某种意义上优化器层面的 NFL 定理应该也是存在的。不过现在已经九点了，我应该没有办法在三小时内思考出来优化器意义上的 NFL 是什么形式。不过我顺便查到了一篇论文，好像也是 NFL 延伸的工作，后续可以参考。[3]

5.3 特殊问题构造

论文 [4] 给出了一个十分精妙的构造，使得 Adam 在该问题上表现不佳。

首先我们依据论文 [4] 给出的设置，完成了一个实验。比较奇怪的是，他们自己给出了理论推导，但却没有展示这个构造结果的实验情况，不过没关系，我们做了。

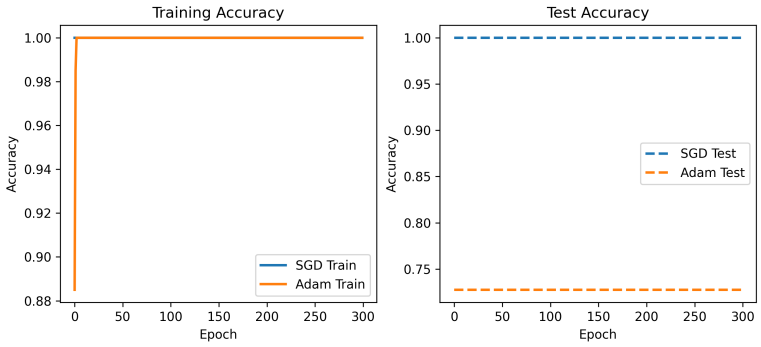


Figure 4: 构造性二分类问题上，Adam 与 SGD 的对比实验

尽管我没有深入研究他们构造的细节，但是实现过程中，我发现他们的构造是十分苛刻的。我尝试使用他们类似的思想，引入噪声维度，构造了二分类问题，但是，只要你不采取他们的实验设置，就无法出现类似的效果。

事实上，他们的构造恰恰证实了我们的观点：“你以为好的优化器，总能找到损失曲面，使其没那么好”。以子之矛，攻子之盾的行为是小聪明，这不可取，要印证我们的观点，我们就要自己给出一个新的构造，使得 SGD 在该问题上完全失效，而 Adam 依然优秀。考虑到我们在先前的实验中，发现了 Adam 第一轮更新异常的现象，我们决定构造一个初始化点在盆地，盆地四周是高坡，高坡之外是“瀑布”的损失曲面。这样，由于 Adam 第一轮更新过大，直接跳出盆地，落入瀑布，从而获得优秀的泛化性能。而 SGD 由于更新较小，始终困在盆地中，无法逃脱，从而泛化性能极差。我们的构造直接指向了损失曲面，而他们的构造则是间接地通过数据分布来影响损失曲面。他们的构造是更加困难且高明的。

不过，我个人的观点是，花时间在这种精妙的构造上意义不大。我们应该广泛地研究真实损失曲面中的性质，找到自然界数据带给我们的对损失曲面结构特征的先验知识，并以此为基础，设计出更好的优化器。

6 Conclusion

通过这次实验，我们否定了前一次作业中提出的猜想。更进一步地认识到了损失曲面性质是一个深刻而复杂的问题。尽管我们未能实现预期的目标，提出一套合适的理论框架。但是我们的工作并非无意义的，首先，我们补充了关于 Adam 与 SGD 泛化性质的实证研究，其次，我们提出了优化器选择上应该也存在类似于 NFL 的机制。

遗憾的事情是，我们很大程度上只是完成了若干实验内容，并没有将自己的任何思想转化为落地的理论。尽管如此，我认为这些观察，实验过程中的感受，质疑的产生，实施，都是十分珍贵的经验。

References

[1] Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from adam to sgd, 2017. URL `https://arxiv.org/abs/1712.07628`.

[2] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL `https://arxiv.org/abs/1412.6980`.

[3] Tom F. Sterkenburg and Peter D. Grünwald. The no-free-lunch theorems of supervised learning. *Synthese*, 199(3–4):9979–10015, June 2021. ISSN 1573-0964. doi: 10.1007/s11229-021-03233-1. URL `http://dx.doi.org/10.1007/s11229-021-03233-1`.

[4] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning, 2018. URL `https://arxiv.org/abs/1705.08292`.

[5] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

[6] Zeke Xie, Xinrui Wang, Huishuai Zhang, Issei Sato, and Masashi Sugiyama. Adaptive inertia: Disentangling the effects of adaptive learning rate and momentum, 2022. URL `https://arxiv.org/abs/2006.15815`.

[7] Zilu Zhang. Mytorch: A minimal numpy-based autograd and neural network framework, 2025. URL `https://github.com/ZZL-2005/Mytorch/blob/main/report/main.pdf`.

[8] Zilu Zhang. Gradient propagation and loss geometry (hw2), 2025. URL `https://github.com/ZZL-2005/Question-Answers/blob/main/main.pdf`. Course Assignment A2, Deep Learning and Optimization, 2025.