

華中科技大學

本科生毕业设计

基于生成对抗网络的闪存仿真器设计与实现

院 系	计算机科学与技术
专业班级	计算机 1604
姓 名	孟嵩淼
学 号	U201614613
指导教师	吴非

2020 年 06 月 10 日

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包括任何其他个人或集体已经发表或撰写的成果作品。本人完全意识到本声明的法律后果由本人承担。

作者签名： 年 月 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保障、使用学位论文的规定，同意学校保留并向有关学位论文管理部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权省级优秀学士论文评选机构将本学位论文的全部或部分内容编入有关数据进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于 1、保密口，在 年解密后适用本授权书

2、不保密口 。

（请在以上相应方框内打“√”）

作者签名： 年 月 日

导师签名： 年 月 日

摘要

近年来，闪存因其高性能、低功耗等特点，逐渐替代磁记录成为主流存储器件，但其可靠性却受到严重挑战。存储在闪存中的数据受到复杂的干扰条件而产生比特错误，深入理解其错误特性是闪存存储可靠性优化的基础。因此需要对闪存全方位测试，结合大数据技术进行特性分析、建模。而错误特性的采集往往基于真实的闪存测试平台，覆盖测试所有干扰条件组合，需要极高时间和空间开销。随着闪存技术的持续发展、新型闪存架构、工艺的不断涌现，面对品类浩繁的闪存型号及日益增长的测试需求，传统闪存测试与特性分析技术受到严重挑战。亟需建立一套闪存仿真器，以满足快速测试与特性分析的需求。

生成对抗网络通过生成器与判别器的对抗学习，具有强大的生成能力。基于生成对抗网络的闪存仿真器通过对部分干扰条件组合快速获取的数据集进行统计，分析闪存数据在不同干扰条件下的错误分布，并利用真实数据对条件生成对抗网络进行训练，所得生成器可生成未覆盖测试条件下闪存块内各页的相对错误分布。并通过块间错误分布控制单个闪存块的错误总数，与块内各页相对错误分布合并，最终获得可与真实数据相媲美的闪存错误分布，突破闪存快速测试与特性分析带来的挑战。

关键词：条件生成对抗网络；闪存；错误特性；可靠性；数据生成；闪存测试；

Abstract

In recent years, due to its high performance and low power consumption, flash memory has gradually replaced magnetic recording as the mainstream storage device, but its reliability has been seriously challenged. The data stored in the flash memory is subject to complex interference conditions and generates bit errors. A deep understanding of the error characteristics is the basis for flash memory storage reliability optimization. Therefore, comprehensive testing of flash memory is required, combined with big data technology for feature analysis and modeling. The collection of error characteristics is often based on a real flash memory test platform, covering all combinations of interference conditions, requiring extremely high time and space overhead. With the continuous development of flash memory technology, the continuous emergence of new flash memory architectures and processes, the traditional flash memory testing and characterization techniques are severely challenged in the face of the vast variety of flash memory models and the growing testing needs. There is an urgent need to establish a flash memory emulator to meet the needs of rapid testing and characterization.

Generative Adversarial Network have powerful generating capabilities through adversarial learning between generators and discriminators. The flash memory simulator based on Generative Adversarial Network analyzes the error distribution of the flash memory data under different interference conditions by statistically acquiring data sets obtained by combining some interference conditions, and uses real data to train the Conditional Generative Adversarial Network. Generate a relative error distribution of pages within the flash block under uncovered test conditions. And through the error distribution between blocks to control the total number of errors in a single flash block, merge with the relative error distribution of each page in the block, and finally obtain a flash error distribution comparable to real data, breaking through the challenges brought by flash memory rapid testing and feature analysis.

Keywords: Conditional Generative Adversarial Network(CGAN), Flash Memory, Error characteristics, Reliability, Generate data, Flash test

目 录

摘 要.....	I
Abstract.....	II
1 绪 论.....	1
1.1 课题背景.....	1
1.2 国内外研究现状.....	2
1.3 生成对抗网络.....	4
1.4 研究目的和主要内容.....	6
1.5 论文结构.....	6
1.6 课题来源.....	7
2 闪存仿真器开发方案论证.....	8
2.1 闪存仿真器需求分析.....	8
2.2 开发工具分析及选择.....	10
2.3 基本方案制定.....	10
2.4 本章小结.....	11
3 闪存仿真器总体设计.....	12
3.1 各模块功能需求分析.....	12
3.2 闪存仿真器总体结构图.....	12
3.3 文本解析模块设计.....	13
3.4 数据统计与分析模块设计.....	14
3.5 生成对抗网络模型设计.....	14
3.6 比特错误总数生成.....	17
3.7 设计中考虑的制约因素.....	20
3.8 成本估算.....	21
3.9 本章小结.....	22
4 闪存仿真器的实现.....	23
4.1 文本解析模块.....	23
4.2 数据库连接模块.....	25
4.3 数据统计与分析模块.....	25

4.4 比特错误相对分布生成模块.....	30
4.5 比特错误总数生成模块.....	34
4.6 结果整合	38
4.7 本章小结.....	38
5 测试与分析.....	39
5.1 测试环境.....	39
5.2 闪存仿真器数据生成测试.....	39
5.3 本章小结.....	47
6 总结与展望.....	48
致 谢.....	49
参考文献	50

1 绪 论

1.1 课题背景

1.1.1 研究背景和趋势

随着信息化时代的到来，人类的生活方式发生了巨大的变革。我国信息产业发展势头良好，产业体系不断完善，产业链掌控能力显著提高，正日益成为我国创新发展的先导力量、驱动经济持续增长的新引擎、引领产业转型和融合创新的新动力、提升政府治理和公共服务能力的新手段^[1]。

全球数据圈^[2]为每年被创建、采集或是复制的数据集合，IDC 预测 2018 至 2025 年全球数据圈将增 5 倍以上^[2]，从 2018 年的 33ZB 增至 2025 年的 175ZB^[2]，同时还预测 2025 年中国将拥有全球最大的数据圈^[2]。根据 IDC 的统计数据显示，中国不同介质类型的存储容量之中，存储在闪存之中的数据容量正在逐年增大^[2]。

基于闪存的固态硬盘在数据存储市场广泛地受到欢迎，但底层的存储介质闪存却变得越来越不可靠。存储在闪存中的数据受到复杂的干扰条件而产生比特错误，深入理解其错误特性是闪存存储可靠性优化的基础。因此需要对闪存全方位测试，结合大数据技术进行特性分析、建模。

错误特性的采集往往基于真实的闪存测试平台，覆盖测试所有干扰条件组合，需要极高时间和空间开销。随着闪存技术的持续发展、新型闪存架构、工艺的不断涌现，面对品类浩繁的闪存型号及日益增长的测试需求，传统闪存测试与特性分析技术受到严重挑战。亟需建立一套闪存仿真器，以满足快速测试与特性分析的需求。

1.1.2 闪存测试面临的问题和挑战

闪存存在运行过程中所产生的比特错误受到多种干扰条件的影响，如编程与擦除操作(Program / Erase, P/E)，保存时间，读写干扰等。以前两种测试条件为例，P/E 周期测试需要模拟真实用户的数据磨损进度（如一年磨损 1000 次），设置合理的间隔时间，导致测试时间成本很高；数据保存时间的测试本身也很耗时，通

常需要高温加速；若 P/E 周期测试范围取[0, 20000]之间的整数（实际上在 P/E 次数接近 20000 时，闪存块将会发生损坏而无法得到完整的数据），那么针对 P/E 周期这个测试条件的条件数目就是 20001 个。同时若准备了 N 个不同大小的时间间隔，作为闪存在不同保存时间的条件下进行测试，再针对这两种测试条件进行组合，将会产生 20001×N 个测试条件组合，想要覆盖测试每种条件组合并在该条件组合下收集大量的块数据，将会产生大量的时间开销和空间开销。若是除了 P/E 周期和保存时间之外，考虑更多干扰条件（如读干扰）及其相互组合时，测试将面临条件组合数暴增，给测试和收集工作带来困难。

1.2 国内外研究现状

研究显示闪存错误的三个主要来源为磨损，数据保持错误和读取干扰^[3]。

磨损。对页面进行编程后，不能够对其覆盖编程，需要擦除后方可再次编程^[4]。重复的编程与擦除操作（P/E 循环）会使得存储数据的闪存单元发生磨损，并对其产生不可逆转的损坏^[3]。闪存制造商因此指定了耐久性极限，闪存块可以承受的 P/E 周期数，并且对于每个新一代闪存，该极限一直在稳步降低^[3]。

数据保持错误。随着时间的推移，存储在闪存单元中的电子会逐渐泄漏，从而使得难以正确读取存储的数据，并且由于存储单元的磨损，由保持错误引起的误差也会增加^[3]。尽管许多研究表明保持错误是错误的主要来源，但电荷泄露是暂时的：只需擦除块，该块便恢复正常^[3]。

读取干扰。读取块中的一条字线会微弱地影响该块中的其他字线，从而无意中更多的电子插入其存储单元^[3]。干扰和数据保持错误是相反的错误机制，但不一定相互抵消：读取干扰主要影响电子较少的存储单元，而电荷泄漏则影响具有较多电子的存储单元^[3]。与保持错误相似，由读取干扰引起的错误会随着单元磨损而增加，同时会随着块的擦除而恢复正常^[3]。

对这三种错误来源，使用公式对闪存原始误码率^[3] (Raw Bit Error Rate, RBER)进行建模：

$$\begin{aligned} RBER(cycles, time, reads) & \quad (1-1) \\ &= \varepsilon + \alpha \cdot cycles^k \quad (\text{磨损}) \\ &+ \beta \cdot cycles^m \cdot time^n \quad (\text{保持错误}) \end{aligned}$$

$$+\gamma \cdot \text{cycles}^p \cdot \text{reads}^q \quad (\text{干扰})$$

其中 ε , α , β 和 γ 是系数, 而 k , m , n , p 和 q 是闪存特有的指数。这九个参数定义了闪存芯片的 RBER。块可以被认为是独立的, 读取, 写入或擦除一个块不会影响到其他块, 原因在于将块与 SGS 和 SGD 器件以及在擦除过程中通过高效的浮动字线方案施加到其他块的电压隔离。尽管 RBER 是一个有用的参数, 但它却像温度一样, 仅在特定时间和特定位置精确^[5]。RBER 在一个块的不同页之间是不同的, 在不同芯片之间也是不同的^[5]; 页面的位错误数在写入页面的时间点与写入块的其余位置时间点之间也有所不同^[5]; 如果以不同的温度来读写设备, 则 RBER 也会有所变化^[5]。

Y. Cai 设计了一个考虑 P/E 周期效应的阈值电压分布模型^[6], 可以为未来的闪存设计更有效的容错机制^[6]。将闪存单元编程为某个值时, 编程操作不仅会影响该单元的阈值电压, 还会影响其周围的其他单元的阈值电压^[7]。这种干扰可能导致周围的单元移动到与其原始状态不同的逻辑状态 (即阈值电压范围), 从而导致在读取该单元时出现错误^[7]。Y. Cai 开发了一个新模型, 该模型可根据阈值电压值和相邻单元中的变化来预测编程干扰的数量^[7]。Y. Cai 使用 2y-nm MLC NAND 闪存芯片来表征闪存的阈值电压分布如何随着不同的保存时间 (自对闪存单元编程以来的时间长度) 而变化^[8]。从特征结果中观察到: (1) 闪存存在最佳读取参考电压, 并随着其保存时间而变化, 使用该电压可以以最低的原始误码率 (RBER) 读取数据^[8]; (2) 不同闪存区域可能具有不同的保存时间, 因此有不同的最佳读取参考电压^[8]。Y. Cai 首次实验性地描述了最先进的 2y-nm MLC NAND 闪存芯片的读取干扰错误^[9]。发现 (1) 阈值电压偏移的大小与读取操作次数相关联^[9]; (2) P/E 周期计数和保存时间是如何影响读取干扰引起的错误^[9]; (3) 确定降低直通电压电平可减少读取干扰的影响并延长闪存寿命^[9]。Y. Cai 基于最先进的 MLC 和 TLC 的 NAND 闪存测试数据, 描述了几种错误缓解和恢复技术, 包括: (1) 减轻小区间干扰^[10]; (2) 最佳的多级单元感测^[10]; (3) 使用最新算法进行纠错^[10]; (4) 纠错失败时的数据恢复^[10]。

B. Schroeder 概述了在生产中对闪存可靠性的了解, 包括 (1) 闪存驱动遇到的各种类型的错误及其频率^[11]; (2) 磨损, 使用年限和工作负载对 RBER 的影响

^[11]; (3) 不同类型的硬件故障的现场特性, 包括块故障, 芯片故障以及驱动器的维修和更换率^[11]; (4) 闪存因故障停止工作的症状和对其进行预测^[11]; (5) 比较不同闪存技术 (MLC, eMLC, SLC 驱动器) 的可靠性^[11]; (6) 闪存驱动器和 HDD 可靠性的比较^[11]。

Y. Luo 开发了一个新模型, 该模型可以预测保存时间, 磨损, 自我恢复和温度如何影响原始误码率和单元阈值电压^[12]。以此提出了 HeatWatch^[12], 这是一种提高 3D NAND 闪存可靠性的新机制。

1.3 生成对抗网络

1.3.1 生成对抗网络介绍

生成对抗网络(Generative Adversarial Network, GAN)^[13]是一种非监督式学习的方法^[14], 由一个生成器和一个判别器构成, 生成器以一组随机取样值作为输入, 其输出的数据需要尽可能的接近训练集中的真实数据。如图 1-1 所示, 判别器以真实数据或者生成数据作为输入, 并需要区分真实数据与生成的假数据并输出输入数据的标签值。通过对生成器和判别器不断地进行训练, 两者在动态博弈中达到平衡^[15], 最终生成器可生成足以媲美真实数据的假数据, 使得判别器无法给出确切的判断。GAN 网络通常用于生成以假乱真的图片数据, 视频数据或者 3D 模型。

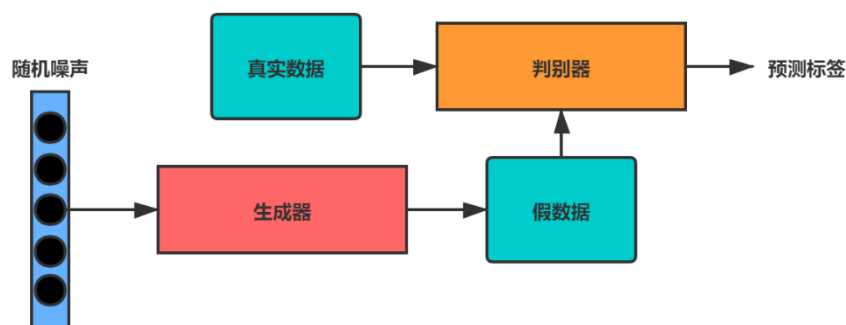


图 1-1 生成对抗网络(GAN)

生成对抗网络自提出以来便得到了广泛的关注, 研究者们对原始的 GAN 网络进行了改进并产生了许多不同类型的变种^{[16][17]}, 以适应不同问题情形下的需要。

Alec Radford 等人提出了深度卷积生成对抗网络(Deep Convolutional Generative Adversarial Networks, DCGAN)^[18], 为 GAN 网络的一个变种。在判别器的内部使用了卷积神经网络^[19]来代替全连接神经网络, 在生成器内部则使用了转置卷积进行上采样来生成假数据, 并对网络内部的架构进行了精心的设计来解决原始 GAN 网络的问题, 卷积网络的引入则提高了模型对于图像数据的学习能力。

Martin Arjovsky 提出了 Wasserstein GAN^[20]从数学原理层面对 GAN 网络进行改进, 它去掉了判别器的 Sigmoid 层和损失函数的 log 层^[21], 每次更新判别器的参数后将其截断到一个区间 $[-c, c]$ (c 为一固定常数)^[21], 并且不再使用基于动量的优化算法^[21], 最终解决了 GAN 网络训练不稳定的问题^[21], 增加了生成样本的多样性^[21]。

Jun-Yan Zhu 提出了循环生成对抗网络(Cycle-Consistent Adversarial Networks, CycleGAN)^[22], 模型存在两个判别器分别针对数据域 A 和数据域 B 进行鉴别, 同样存在两个生成器, 分别负责将数据从 A 映射到 B 和从 B 映射到 A。CycleGAN 根据其特性常用于图像翻译转换工作, 例如将普通马匹的图像转换为斑马的图像。

1.3.2 条件生成对抗网络模型简介

Mehdi Mirza 等人提出了条件生成对抗网络(Conditional Generative Adversarial Network, CGAN)^[23], 为 GAN 网络的一个变种。如图 1-2 所示, 对于生成器, 除了一组随机取样值 z 外, 还需添加一标签值 y 用于控制生成结果的类型; 同样对于判别器除了以待判别数据作为输入外, 也需要额外添加指定数据类型的标签值 y 。CGAN 网络训练至收敛后, 可以通过输入生成器的标签值来控制产生数据的类型, 有别于 GAN 网络数据生成的随机性。

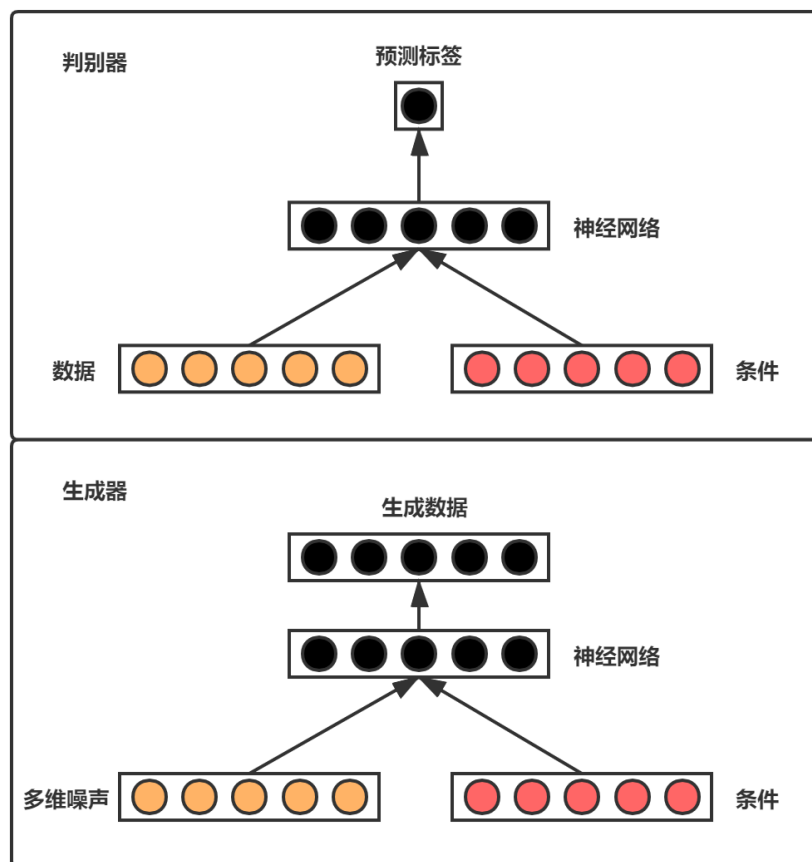


图 1-2 条件生成对抗网络(CGAN)

1.4 研究目的和主要内容

本课题将基于生成对抗网络设计并实现一个闪存仿真器,用以仿真闪存的性能、可靠性等行为,支持不同干扰特性定量组合的闪存数据生成。

研究该课题的直接意义在于,它可以对不同测试条件进行定量组合,生成可媲美测试数据的结果,大大降低 3D 闪存测试的时间成本和空间成本。

1.5 论文结构

本文的主要内容如下:

第一章介绍了本课题的来源,以及研究的目的和意义;介绍了相关背景和研究趋势,论述了国内外研究现状,包括闪存错误特性研究以及生成对抗网络的相

关研究。

第二章则进行了方案论证，介绍了闪存的基本知识，对块数据进行分析，设计了多种方案进行论证，最终选择一种方案来实现，并根据需求选择要使用的开发工具。

第三章对系统的整体结构进行设计，说明了系统各个模块的功能需求，以及各个模块的设计过程。

第四章则是详细介绍了系统各个模块的具体实现方案，以及各个模块之间的作用方式。

第五章对整个系统进行了测试，包括各个功能模块的测试，以及对系统最终的生成数据的效果进行了测试与分析。

第六章对本论文所做的工作进行了总结，并对未能实现的一些方案进行了展望。

论文的最后是致谢与参考文献。

1.6 课题来源

该课题来自国家自然科学基金。

2 闪存仿真器开发方案论证

上一章对于论文的研究背景与趋势进行了分析，并阐明了本课题所要解决的问题。本章将对课题内容进行具体分析，并设计一种方案进行实现。

2.1 闪存仿真器需求分析

闪存是一种非易失性存储器，断电之后数据也不会丢失^[24]。其基本存储单元（Cell）是一种类 NMOS 的双层浮栅 MOS 管^[24]。一个存储单元存储 1bit 数据的闪存，被称作 SLC（Single Level Cell），存储 2bit 数据的闪存为 MLC（Multiple Level Cell），存储 3bit 数据的闪存为 TLC（Triple Level Cell）^[24]。

一个闪存内部的存储组织结构如图 2-1 所示，一个闪存芯片有若干个 DIE（又被叫做 LUN），每个 DIE 有若干个 Plane，每个 Plane 有若干个 Block，每个 Block 有若干个 Page，每个 Page 对应一个 Wordline，Wordline 由成千上万个存储单元构成^[24]。

一个 Wordline 对应着一个或若干个 Page，具体是多少取决于是 SLC、MLC 或者 TLC^[24]。对 SLC 来说，一个 Wordline 对应一个 Page^[24]；MLC 则对应 2 个 Page，这两个 Page 是一对^[24]；TLC 对应 3 个 Page^[24]。

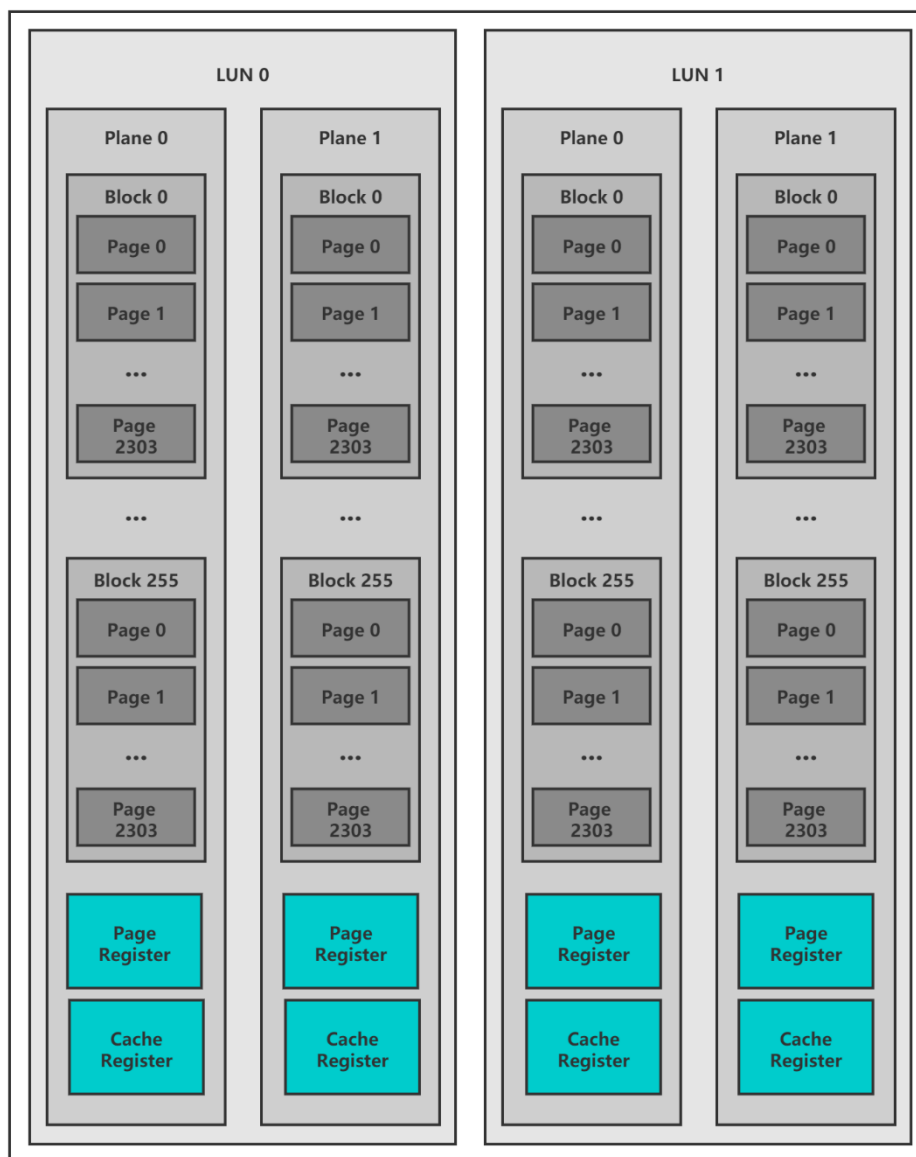


图 2-1 闪存内部组织架构^[24]

实验中用到的闪存，其每个块由 2304 个页（Page）组成，每个页又被划分为 16 个 Frame。块在特定条件下进行测试，这 2304×16 个位置均会产生一定量的比特错误。现对闪存在不同的 P/E 周期数条件下进行测试，得到了一系列记录文件，其中详细记录了块在闪存中的位置，测试条件，以及块中每个位置的错误数等信息。从这些记录文件中提取得到块在不同 P/E 条件下的比特错误矩阵并组织成真实数据集。要求通过 GAN 网络使用真实数据集进行训练，最终得到的生成器可以产生足以媲美真实数据的生成数据。

如图 2-2 为块数据的形式，记录了在特定 P/E 条件下进行测试，一个闪存块每个位置的错误数，最终目标生成数据也为这种形式，即尺寸为 2304×16 的整形矩阵。

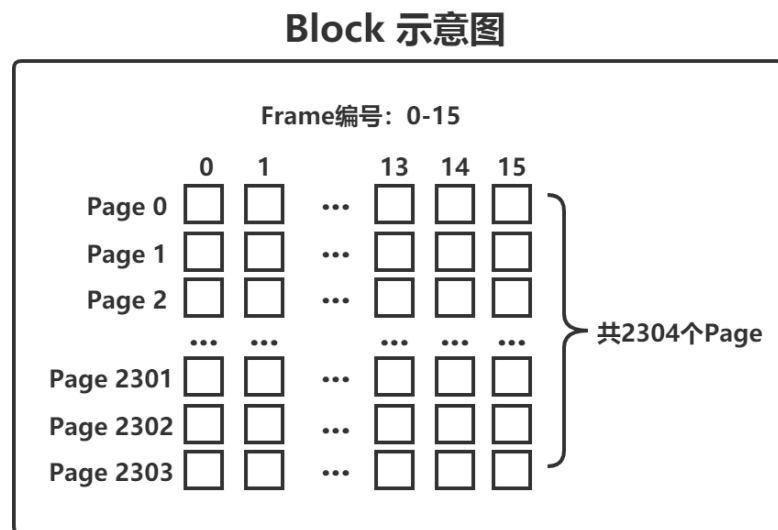


图 2-2 数据形式

2.2 开发工具分析及选择

目前较流行的机器学习的开源框架有 Tensorflow^[25]，Pytorch^{[26][27]}，LightGBM，Keras 等等。考虑到 Pytorch 接口定义简洁明晰，容易学习和掌握，且多用于学校的学术研究之中，因此选择 Pytorch 来构建整个系统，选择编程语言 Python^{[28][29]}与 IDE PyCharm。由于 Anaconda3 集成了大量的用于科学计算的工具，因此选择 Anaconda3 作为使用 Python 的一个基本环境。

2.3 基本方案制定

根据系统需求，需要在不同的测试条件下对闪存错误数据进行生成，因此应选择条件生成对抗网络（CGAN）作为系统构建的模型。

方案 1：块错误矩阵维度为 2304×16 ，判别器与生成器内部使用全连接神经网络，考虑直接将块错误矩阵平铺为一维数据进行输入，因此生成器的输出层与判别器的错误数据输入部分的尺寸为 36,864 (2304×16)。以这种最直接的方式

处理数据进行训练，训练效果不佳，原因在于输入层尺寸过大，使用全连接神经网络会产生大量的训练参数使得计算压力过大，且生成器与判别器训练不均衡。

方案 2：将原始数据处理为尺寸 $48 \times 48 \times 16$ 的多通道数据作为输入，它相当于分辨率为 48×48 的 16 通道图像，块错误矩阵中的列 $f_0 - f_{15}$ 分别作为一个通道，而在列内的 2304 个值组织成尺寸为 48×48 的二维数据。因此在 CGAN 的内部使用卷积神经网络，来处理这种具有图像特点的数据。

方案 3：对每个块错误数据进行处理，统计出块中每页的错误总数 ($P_i, i = 0, 1, \dots, 2303$)，将所有的页错误数除以该块中的最大页错误数 (P_i/P_{max})，得到该块所有页的相对错误分布比率（数据尺寸为 2304×1 ）。整个方案分为三部分：

(1) 使用块的相对错误分布数据集对 CGAN 进行训练，使得最终的生成模型可根据条件 (P/E 次数) 产生对应的相对错误分布。

(2) 针对每个 P/E 值都建立一个正态分布来对该条件下的块错误总数分布进行拟合，最终使用该正态分布生成块的错误总数。

(3) 整合 (1) (2) 得到块的各项错误总数，针对每一页错误总数，生成和为页错误总数的 16 个均匀分布的随机数，最终得到块每个位置的错误数据。

最终选择使用方案 3，该方案的合理性将在后文论证。

2.4 本章小结

本章分析了整个系统的需求，并从需求出发选择了要使用的开发工具与技术，并制订了多个方案进行对比，最终确定将整个生成过程分为三部分，首先对块各项错误总数的相对分布进行生成，之后生成块的错误总数，由这两步可得到各项的错误数量，之后针对每一页，利用均匀随机分布生成页内的错误数量，最终完成数据的生成。

3 闪存仿真器总体设计

前面分析了系统的需求和方案，本章则根据方案的功能需求，对系统进行整体设计。

3.1 各模块功能需求分析

(1) 设计文本解析模块，能够对闪存的记录文本进行解析，从中提取闪存块的各项属性，测试条件，以及块错误矩阵，并以页为单位导入数据库或者以二进制格式保存至本地。

(2) 设计数据库连接模块，能够连接数据库服务器，能够从数据库中读取块错误数据等信息。

(3) 设计数据统计模块，可以对原始数据进行统计与分析，根据统计分析结果辅助方案的设计。

(4) 设计块相对错误分布生成模块，应用生成对抗网络，可以对真实数据集进行学习，训练后的模型能够生成媲美真实错误分布的假数据；

(5) 设计块错误总数生成模块，针对不同的 P/E 值，能够生成符合真实数据分布的块错误总数。

(6) 能够对相对分布和错误总数进行整合，得到完整的块错误数据。

3.2 闪存仿真器总体结构图

系统整体设计如图 3-1 所示，原始的闪存块错误信息以文本文件的格式存储，首先通过文本解析模块将块错误信息从文本中提取出来，或导入数据库，或以二进制文件格式保存在本地。可以对这些处理得到的数据集进行统计与分析，得到数据的一些特征来辅助系统设计。对原始数据集进行处理来训练 CGAN 网络，可得到错误相对分布生成模型，同时错误总数生成模块可以生成块的错误总数，最终整合这些生成数据，得到完整的块错误数据。

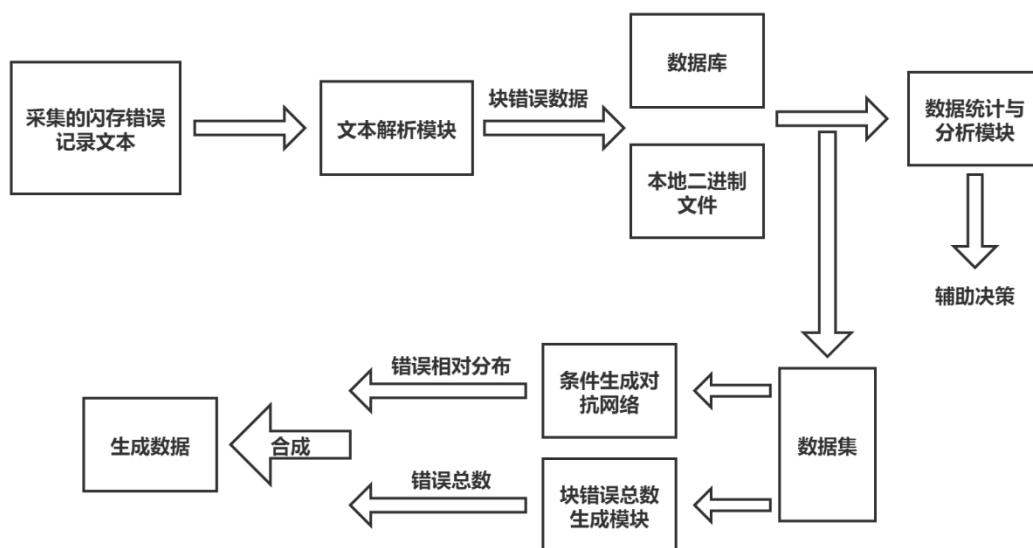


图 3-1 系统总体设计

3.3 文本解析模块设计

闪存错误记录以单个文本文件为单位，存储了芯片当中一个 chip 的闪存块在运行过程中的错误信息，以若干个 chip 的错误记录文件组成一个文件夹代表一组记录，并以测试日期对文件夹进行命名。

在记录文件之中，以块为单位进行错误信息采集，并以多行文本进行存储，其中每行记录块中一页的错误情况，包含该页所在的位置、页错误总数和各个 Frame 的详细错误情况。

因此，对于文本解析模块来说，最核心的部分便是对单个记录文件的处理，并从中提取所要的信息。考虑其存储形式，信息提取以逐行处理的方式进行，并将提取到的信息暂时保存至内存当中。

最终的目的在于从多个文件夹内的记录文件中提取信息，并进行聚合，最终保存至单个二进制文件当中，以方便之后模型的训练。

但由于每个文本文件的尺寸较大，处理时间较长，因此设计以文件夹为单位进行信息提取，将提取到的信息保存至二进制文件中。最终再将多个二进制文件整合在一起，构成数据集。

3.4 数据统计与分析模块设计

数据统计与分析模块的目的在于辅助决策，为后续生成模块的设计提供支持。因此在设计之时较为灵活多变，随时可根据需要对代码进行修改。

对闪存块错误数据的分析主要从以下角度进行：（1）在不同 P/E 条件下，闪存块错误总数的变化趋势；（2）在固定 P/E 条件下，闪存块错误总数的分布情况；（3）在不同 P/E 条件下，闪存块各位置相对错误分布的变化趋势；（4）在固定 P/E 条件下，闪存块错误次数在页和 Frame 两个维度的分布情况。

通过对数据分布的观察与分析，针对其特点对生成模块进行合理的分解，将生成工作分为几个部分，最终再对各个部分的工作结果进行整合，得到最终的结果。

为方便对分析结果进行观察，因此多采用图像的方式对结果进行呈现。例如（1）以曲线对块错误总数随 P/E 变化趋势进行拟合；（2）以散点图对不同 P/E 值下的块错误总数分布进行直观地显示，再以频数分布直方图对其分布进行数学意义上的统计；（3）为直观显示块的相对错误分布，可利用灰度图对每个位置的错误次数进行映射，以灰度图不同位置的明暗来对应块中各位置错误次数的多少；

3.5 生成对抗网络模型设计

比特错误相对分布是指在比特错误总数确定的情况之下，块中哪些位置错误数较多，哪些位置错误数较少。主要利用条件生成对抗网络来实现比特错误相对分布的生成。

（1）网络输入中的条件部分

a. 如图 3-2, 对 P/E 值进行归一化和正则化后直接输入网络来控制数据生成；

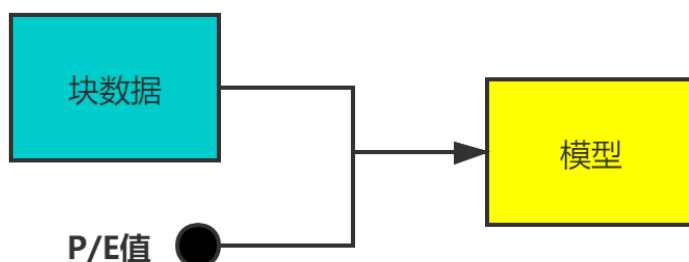


图 3-2 直接输入模型

b. 虽然 P/E 值在整数范围内是一个连续的取值，但也可以以分类的形式来进行输入，如图 3-3 所示；以一定的间隔值，如 1000，来获得 P/E 取值集合{1, 1000, 2000, 3000, ... 16000, 17000}，而在集合当中的每个值都被当作是一种生成类别来进行处理，并将其编码为对应的“one-hot”向量，如 P/E 值等于 1 可编码为(1, 0, 0, ..., 0, 0)，P/E 值等于 1000 可编码为(0, 1, 0, ..., 0, 0)，P/E 值等于 17000 可编码为(0, 0, 0, ..., 0, 1)，该向量为 18 维，分别对应 P/E 值集合中的 18 个值。对于不处在集合当中的 P/E 值，采用其所在 P/E 区间两端的生成数据再进行加权平均的方式获得，例如 P/E 值为 6700，并不属于给定的 P/E 值集合，便将其两端的 P/E 值 6000 和 7000 输入模型获得生成数据 $data_{6000}$ 和 $data_{7000}$ ，那么按距离进行加权平均的方式，则 $data_{6700} = \frac{300}{300+700} \times data_{6000} + \frac{700}{300+700} \times data_{7000}$ 。

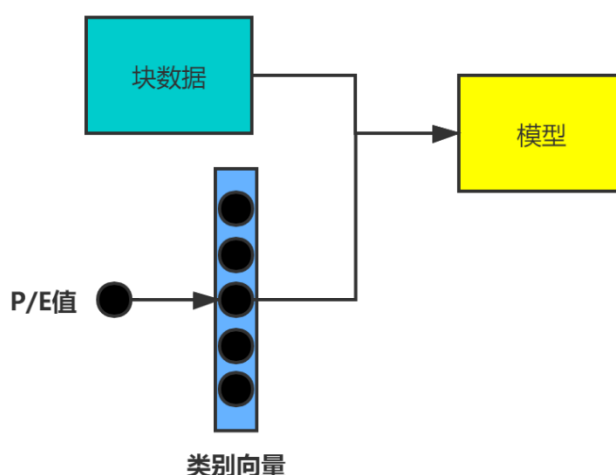


图 3-3 编码为向量输入

方案 a 中，输入条件为连续值，其优势在于对于训练好的生成器，可直接将 P/E 周期输入模型来生成数据；方案 b 中，输入条件为离散值，模型能够更好地区分不同 P/E 值之间带来的差异，但其使用的类别向量意味着训练好的生成器只能生成几种固定 P/E 值的数据。

目前所考虑采用的方案是方案 a，若是不同 P/E 值之间的生成效果差异不够明显，可以考虑使用方案 b。

(2) 网络输入中的块数据部分

a. 如图 3-4 所示，将 2304×16 的块数据直接平铺展开输入全连接神经网络，此种方式最为简单，但考虑到块数据的尺寸过于庞大，因此预计模型参数多，训练效果差。

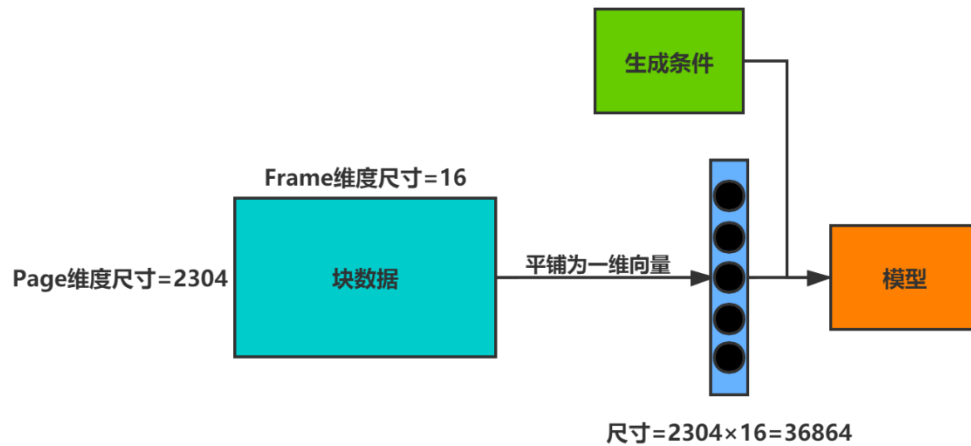


图 3-4 平铺输入

b. 如图 3-5 所示，将 2304×16 的块数据当作一张二维图片输入到卷积神经网络当中，通过精心设计的网络结构进行训练，但是由于图片的长宽过于悬殊(即 Page / Frame 过大)因此卷积核对于图片局部特征的学习效果较差，无法得到想要的结果。

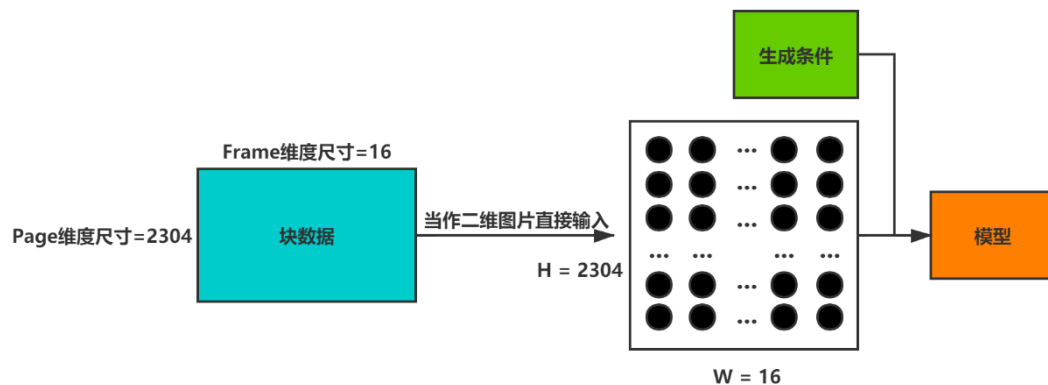


图 3-5 直接输入卷积神经网络

c. 如图 3-6 所示，将 2304×16 的块数据作为有 16 个通道的 $48 \times 48 \times 16$ 的图片输入到卷积神经网络当中，结合数据分析模块的统计结果，块在 Page 维度和 Frame 维度的特征均能得到妥善的学习，预计此种方式将会表现良好。

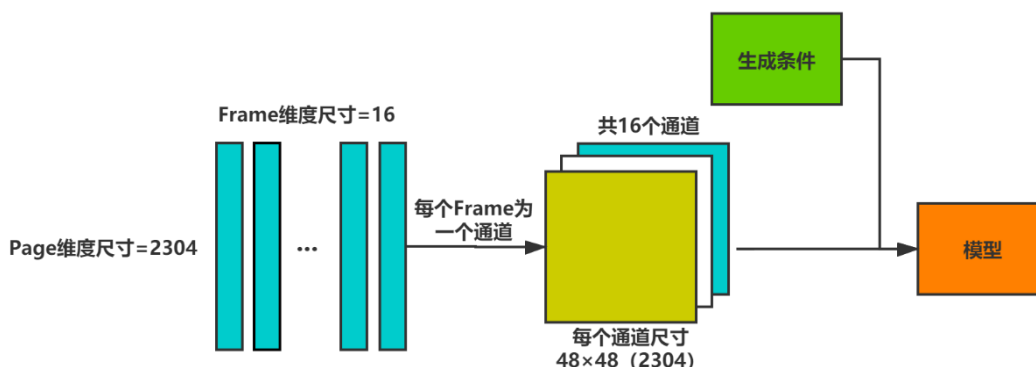


图 3-6 多通道输入卷积神经网络

d. 根据数据分析模块的统计结果，对于比特错误在一个 Page 的 16 个 Frame 的分布情况，可以考虑使用均匀分布模型进行生成，如图 3-7 所示，那么 CGAN 网络便只需对每个 Page 的错误总数分布进行学习，此时数据的尺寸被直接减小至 2304×1 ，因此使用最简单的全连接神经网络也可以得到很好的训练效果，同时模型内部需要进行学习的参数也大大减少，减轻了对于硬件设备的负担。

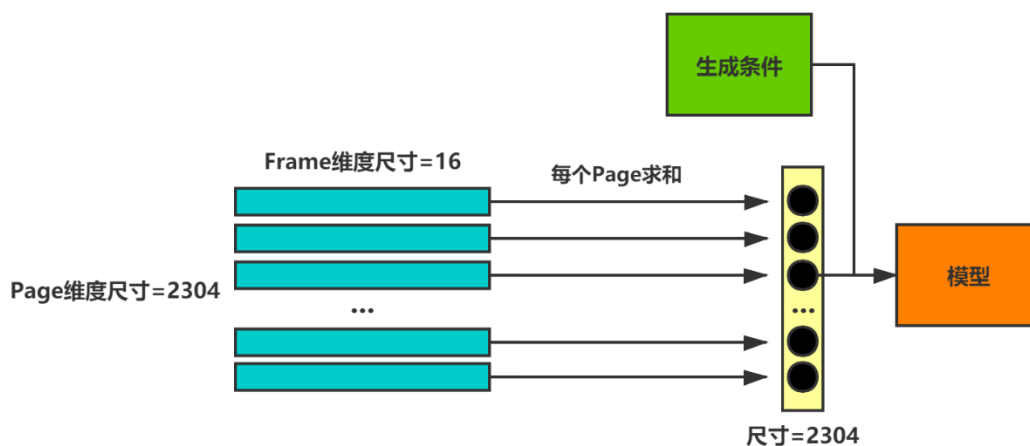


图 3-7 每页求和输入

3.6 比特错误总数生成

根据数据分析模块的统计，块错误总数平均值随 P/E 值的变化基本呈现一个线性增长的趋势；但在 P/E 值确定的情况下，两个块的错误总数差别可能非常巨大，因此可以明确曾设想的块错误总数是围绕其均值有一个较小的偏差是错误的，在 P/E 值确定时，块错误总数也存在一个分布。

通过对所拥有的块数据进行统计,在 P/E 值确定时,绘制其频数分布直方图,发现块错误总数的分布比较近似正态分布,因此考虑对不同的 P/E 值下的块数据,统计其错误总数均值和标准差,并以此为根据设计正态分布对真实分布进行拟合。

但实际上,这种方式的问题在于,当 P/E 值确定时,块错误总数的分布恰好较符合正态分布。如果之后的工作当中,测试条件除了 P/E 值外,添加了数量更多的输入条件,那么这种生成错误总数的方式将会变得困难。因此对块错误总数的生成方式进行了如下构想:

a. 如图 3-8 所示,将块错误总数分布所处的区间分为 N 个子区间,以相同测试条件下若干个(例如 3000 块)块的错误总数为一组,统计其落在这 N 个区间的频率,以这 N 个频率数据构成训练数据集中的一条数据。最终将构造的数据集输入到条件生成对抗网络当中进行训练,以此方式学习得到块在不同测试条件下,错误总数的分布情况。

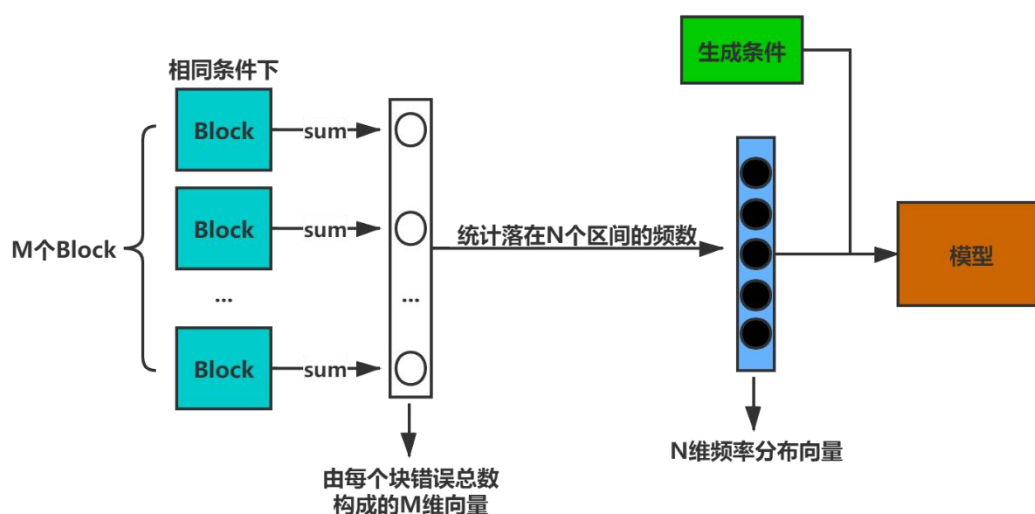


图 3-8 块错误总数生成

不过目前之所以不采用这种方式的原因在于,它需要测试更多的块数据,而目前所拥有的块数据远远不够。其中的问题在于,对错误相对分布进行学习时,一个块数据的信息便构成了数据集中的一条数据;但使用这种方法生成块错误总数时,需要多个块数据(如 3000 块)才能得到一条具有统计意义的频率分布情况,如图 3-8,对 M 个 Block 进行统计之后,仅获得了一个各区间频率分布向量。

考虑的解决方法，(1) 增加所测试的块的数量；(2) 对已有的块数据进行复用，例如以 3000 个块数据为一组，倘若目前拥有 6000 个块数据，如果以不重复的方式形成数据集，6000 个块可被分为两组，对这两组数据以图 3-8 所示的方式进行统计，那么只能得到两个频率分布向量，即数据集中的两条数据；但是以可重复的方式，从这 6000 个块数据中随机抽取 3000 个块构成一组进行统计，将会得到 C_{6000}^{3000} ($C_n^m = \frac{n!}{m!(n-m)!}$) 个频率分布向量，即数据集中的 C_{6000}^{3000} 条数据。只是此种方式的训练效果未知，也许会因为数据的复用而产生过拟合现象。

b. 直接采用数学统计的方式（与方案 a 类似，只是不再分组统计，不再输入模型训练，直接统计所有数据的分布情况），将块错误总数分布所处的区间分为 N 个子区间，对已有的数据集，统计不同条件下，块错误总数在 N 个子区间的频率分布情况，获得 N 维频率分布向量，在生成时，以统计的频率代替概率随机抽取一个子区间，并在子区间内部使用均匀分布模型来随机生成块错误总数。若要对数据集中不存在的条件进行生成，可以参考 K 最近邻分类算法^[4]的思想，在条件空间中选择 K 个与当前条件最接近的已知条件，获得这些已知条件的块错误总数在各子区间的频率分布向量，最后按在条件空间中的距离进行加权平均的方式得到未知条件的频率分布向量。它要求测试和统计的数据要足够多，使得拥有的已知条件能够较密集且均匀地分布在整个条件空间当中。

以 P/E 周期和保存时间这两个测试条件为例，如图 3-9 所示，为这两个条件进行组合之后所构成的条件空间。图中的每一个点都代表着一种条件组合，其中，每个灰色点意味着已对该条件组合进行过测试，并统计出其频率分布向量，为已知点。而白色点则是未被测试和统计的条件组合，为未知点。若要生成未知点的块错误总数，便选取 K 个（如图 3-9 所示 K=4）与它最近邻的已知点，按在条件空间中的距离远近进行加权平均，合成未知点的频率分布向量，之后再以频率代替概率进行随机抽样来生成块错误总数。

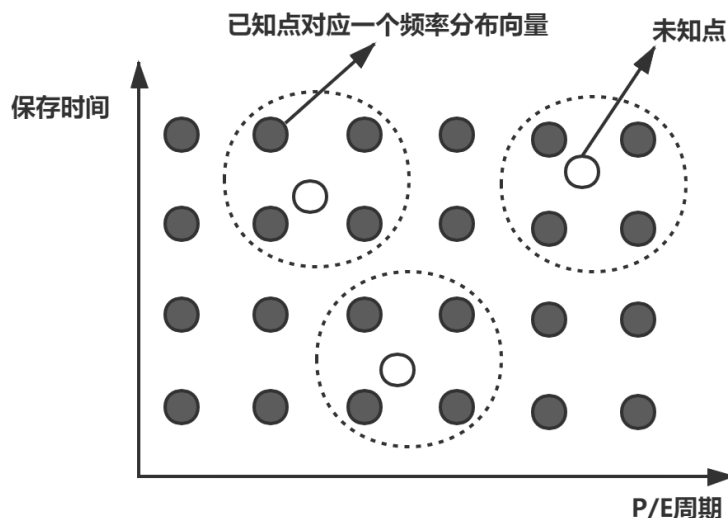


图 3-9 条件空间

它存在的问题是，按在条件空间中的距离进行加权平均的方式未必能真实反应条件组合之间的差异对块错误总数的影响。不过优点在于简单，直接，无需用到复杂的网络模型进行训练。

3.7 设计中考虑的制约因素

3.7.1 数据收集

本论文研究工作的主要内容是通过控制 P/E 值来生成不同条件下闪存块的错误次数矩阵。P/E 值的取值范围处于 0 到 20000 之间的任意整数，但在构建数据集之时很难覆盖测试每一个 P/E 周期，因为它会导致数据集异常庞大。其次 P/E 值的在一个小的范围内变化之时，对于块错误分布的影响是极其微小的。

因此在针对闪存芯片进行测试采集，生成错误记录文件之时，选取的 P/E 值间隔为 100；而在从这些文本文件中提取块数据构造训练数据集之时，所选用的 P/E 值间隔会更大，比如说 500，1000。

3.7.2 模型构造

在进行模型设计，需要考虑到硬件的制约因素。由于 GPU 设备的运算能力限制，在保证训练效果的前提下，模型的结构应该尽可能的简单，所包含的训练参数尽可能的少。

3.7.3 模型训练

使用 GAN 网络生成闪存块错误数据与生成图像数据在判断训练效果方面存在差异。例如，使用 GAN 网络对人脸图像数据集进行训练，那么在使用生成器产生一些假数据之后是很容易通过观察判断模型的训练效果。但生成的闪存块错误数据是很难通过观察来对模型的训练效果进行直观地判断，因此在设计之时通过将生成数据图形化的方式来比对模型的训练效果，包括错误相对分布灰度图和错误总数频数分布直方图两个维度。

3.8 成本估算

选用基本 COCOMO 模型(Constructive Cost Model)对整个工程的开发成本进行估算。基本 COCOMO 模型用到以下变量：

KLOC：千源代码行数。

E：开发工作量，单位“人月”。

D：累计开发时间，单位为“月”。

P：需要的人数

在该模型中考虑开发环境，软件开发项目的类型分为三种：有机项目、中度分离项目、嵌入式项目。

由于本项目相对较小、较简单，程序规模不是很大，因此属于有机项目。其估算工作量，开发时间和开发人数的公式如下：

$$E = 2.4 \times KLOC^{1.05} \quad (3-1)$$

$$D = 2.5 \times E^{0.38} \quad (3-2)$$

$$P = E/D \quad (3-3)$$

KLOC 取值 2，即源代码 2000 行。代入公式得 $E=4.97$, $D=4.61$, $P=1.08$ 。

因此估计开发时间为 4.6 个月，开发人数为 1 人。该模型主要用于开发成本和时间的粗略估算，忽略了开发者水平，工具技术等因素的影响。由于项目开发使用了成熟的开源框架以及编码工作相对简单的 Python 语言，因此预计开发时间要少很多。

3.9 本章小结

本章给出了系统的功能需求分析，说明了系统分为哪些模块以及各个模块的功能，并将这些模块联系在一起，阐明了系统的总体设计思路，并给出了系统总体设计图。接着对各个模块的设计进行了详细的论证分析，并给出了设计中考虑的制约因素以及整个工程的开发成本估算。

4 闪存仿真器的实现

上一章给出了系统的总体设计，本章将给出系统中各个模块的具体实现，包括文本解析模块，数据库连接模块，数据统计与分析模块，块相对错误分布生成模块，块错误总数生成模块。

4.1 文本解析模块

对闪存进行测试可获得一系列错误记录文件，以文本的形式保存着闪存块的测试条件，各页错误信息，块属性等信息。文本解析模块的目的在于通过对这些文本的处理来提取系统所需要的信息，训练模型需要测试时块的 P/E 值和记录块各位置错误情况的错误数矩阵，若要导入数据库则还需额外提取块的一些属性信息，如闪存块的位置信息，页的类型。

每个记录文件保存了闪存中一个 chip 中的若干个块的信息，采用逐行处理的方式来提取信息。在对闪存块进行测试并采集数据时，P/E 测试点的间隔为 100，实际当 P/E 值改变时，闪存块的错误情况变化极不明显，因此在提取信息时考虑将 P/E 次数的间隔值改为 1000 或者其他较大的值，来使得不同 P/E 值下，块错误情况存在较明显的差异。

块的每一页信息在记录文件中保存为一行，因此如图 4-1 所示，对文本进行逐行处理，剔除不满足要求的块，因为当 P/E 值较大时，有些块的页信息不足 2304 页，因此需要剔除；同样需要根据读取到的第一页的信息判断该块的 P/E 值等条件是否满足提取条件。

全部的记录文件保存为若干个文件夹，每个文件夹以测试日期命名，每个文件夹内包含若干个记录文件，以其所测试的 chip 号命名。由于对这些上百 GB 的文本文件处理需要耗费较多的时间，因此以文件夹为单位进行处理，每处理完一个文件夹就写入存档文件，这样即使文件解析过程因异常情况而中断，再次启动时可根据存档文件继续处理。得到的块数据或被导入数据库，或以二进制格式保存至本地（`numpy` 格式文件）；这些块信息包括完整的块错误矩阵，仅记录各页错误总数的块错误矩阵和块的测试条件；

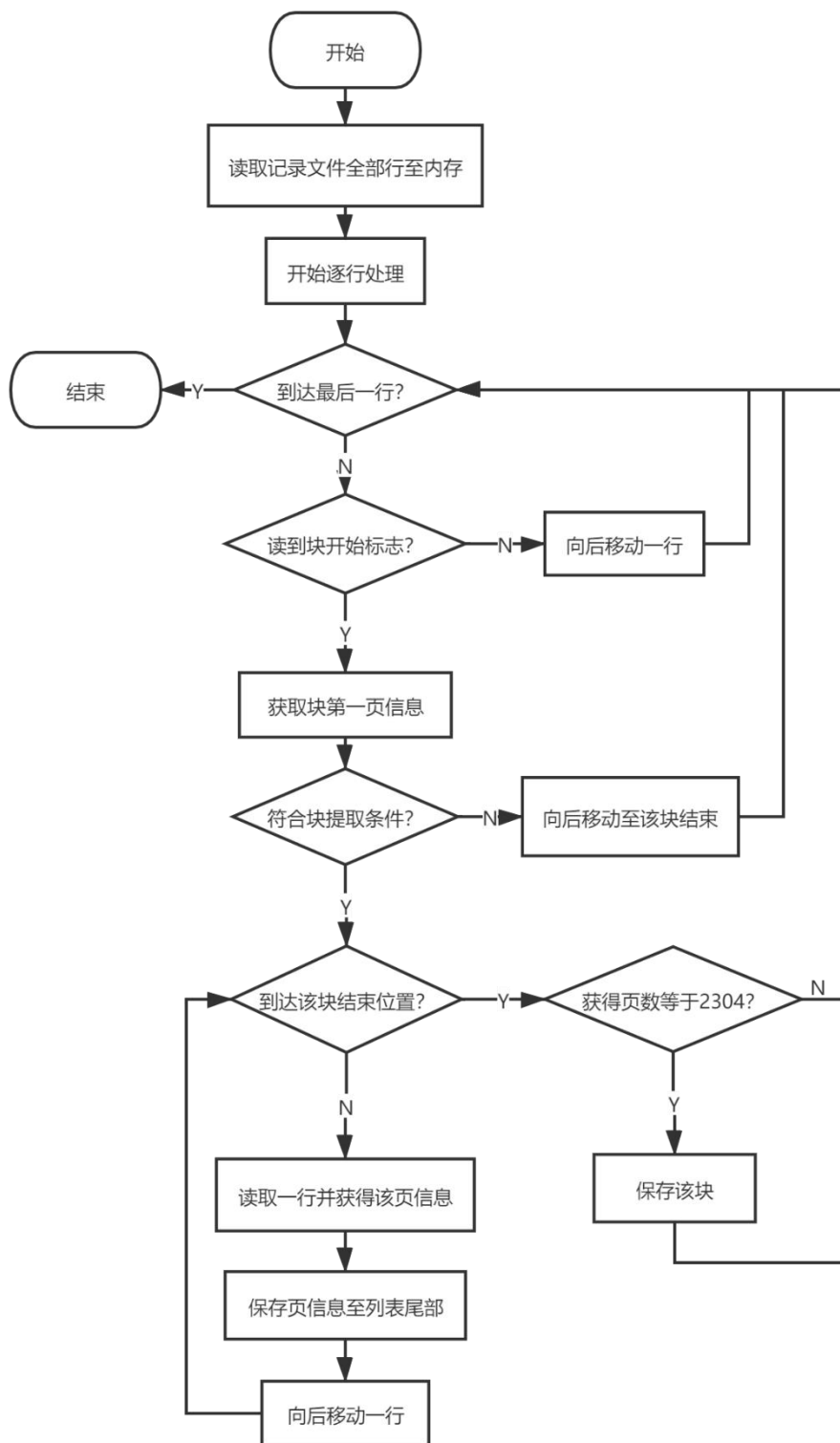


图 4-1 文本解析模块流程图

4.2 数据库连接模块

数据库连接模块需要利用 Python 的 pymysql 库，连接远程的 MySQL^[30]数据库服务器，构造 sql 语句对数据库中的数据进行操作。包括读取数据集配置信息，读取完整的块错误矩阵，读取页信息，写入页信息等读写操作。不过由于从本地读取 npy 格式数据的速度要远快于从数据库中读取数据的速度，因此对于模型的训练直接从本地保存的 npy 格式数据构造数据集。

4.3 数据统计与分析模块

4.3.1 块错误相对分布灰度图

一个闪存块共被划分为 2304×16 个位置，每个位置的错误数或多或少，可将其所有位置错误数的相对分布转换为灰度图直观地显示出来；某个位置错误次数越多，那么该位置的颜色就越趋于黑色，某个位置的错误次数越少，则趋于白色；转换过程如图 4-2 所示，具体转换方式如下：

针对给定 P/E 值，将该 P/E 值对应的块集合中的所有块错误矩阵累加，得到一尺寸为 2304×16 的矩阵 T，矩阵中的每个值记录了所有块在该位置的错误次数总和。现要将该矩阵转换为一尺寸为 2304×16 的灰度图，矩阵中每个位置都对应图片中的一个像素点，转换公式为：

$$Pixel_{i,j} = \left(1 - \frac{T_{i,j}}{T_{max}}\right) \times 255 \quad (4-1)$$

如图 4-3 展示了 P/E 值分别为 1, 8000, 17000 时的灰度图，为了方便显示在文档中，它在纵向上是被严重压缩过的；为方便对比观察，可将图片每行的像素点重新排列组合，将图片尺寸调整为 192×192 ；之后使用生成对抗网络生成的假数据可以用同样的方式转换为灰度图，与真实数据对应的灰度图进行对比，来判断模型的训练效果。

通过对不同 P/E 值对应灰度图的观察可以看出，闪存块的错误分布在页维度上是存在一定规律的，根据不同的 P/E 值而改变；页内的错误分布则接近均匀分布。

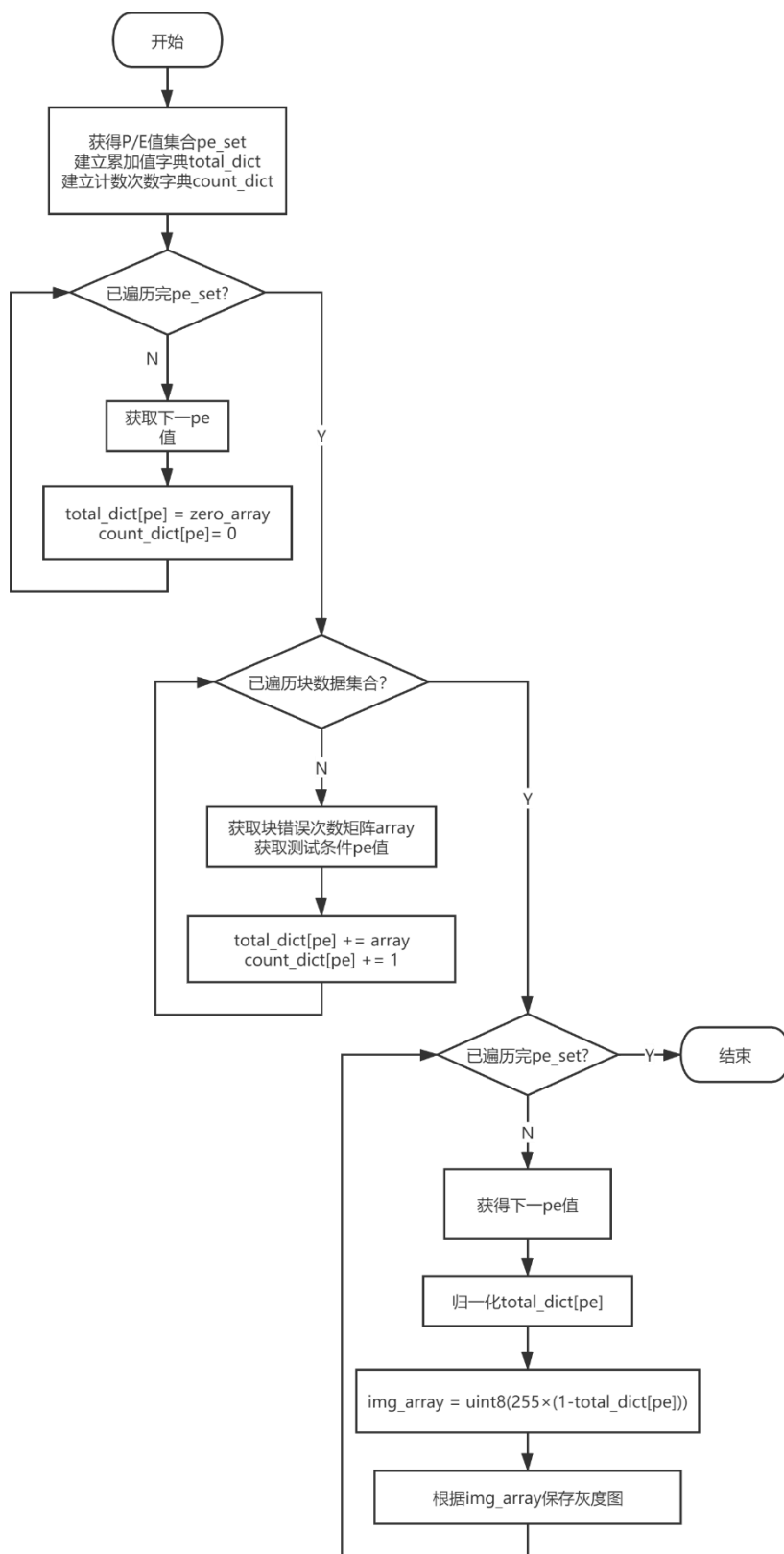


图 4-2 错误矩阵转灰度图流程

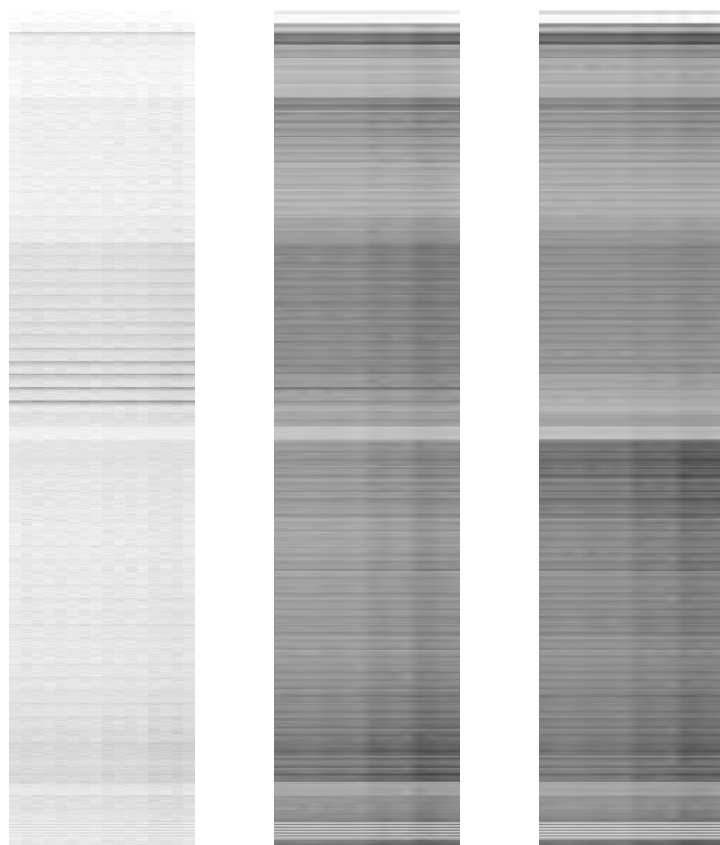


图 4-3 闪存块错误相对分布灰度图
(从左至右 P/E 值依次对应 1,8000,17000)

4.3.2 块错误总数统计与分析

a. 块错误总数整体分析

通过解析闪存错误记录文本文件，在 P/E 值集合： $\{1, 1000, 2000, \dots, 16000, 17000\}$ 的所有测试点收集块错误分布矩阵，共获得块错误分布矩阵 62182 个，每个块错误分布矩阵尺寸为 2304×16 ，每个 P/E 值对应约 3500 个块数据。统计不同 P/E 值下，块错误总数的最小值，平均值以及标准差；如图 4-4 所示，在无法获得所有 P/E 值对应的块错误数据的情况下，可以考虑使用插值法对已有的若干散点进行拟合获得插值函数，用函数值来代替未被测试收集到的 P/E 点。

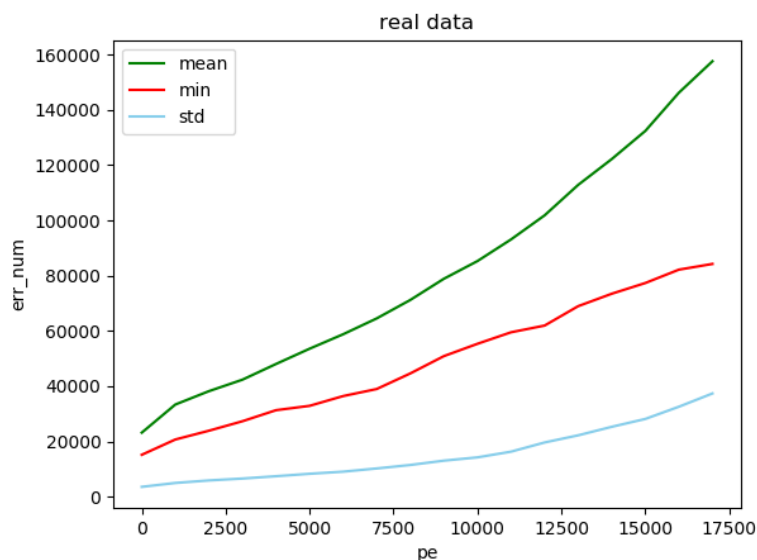


图 4-4 块错误总数-P/E（三条曲线自上到下依次为平均值，最小值和标准差）

b. 特定 P/E 值下块错误总数分布

之前一部分针对了块错误总数的整体情况进行了分析，接下来，则考虑在给定 P/E 值下所有块的块错误总数的分布情况。使用直方图来直观地表示这种分布情况；根据对块错误总数的统计，其值基本分布在区间[15000, 320000]之间，因此如图 4-5 所示，在该区间内以 3000 为间隔将区间分为若干段，针对某一 P/E 值，统计该 P/E 值下所有块的块错误总数落在各个区间内的频数，生成频数分布直方图；

同时使用正态分布 $X \sim N(\text{mean}, (\theta \times \text{std})^2)$ 随机生成与块数量相同的若干随机值，生成其频数分布直方图并绘制，其中 mean 为该 P/E 值下块错误总数均值，std 为该 P/E 值下块错误总数标准差。通过不断调整 θ 值来观察正态分布直方图与真实分布直方图的关系，来确定一个较为合适的 θ 值对真实分布进行拟合。

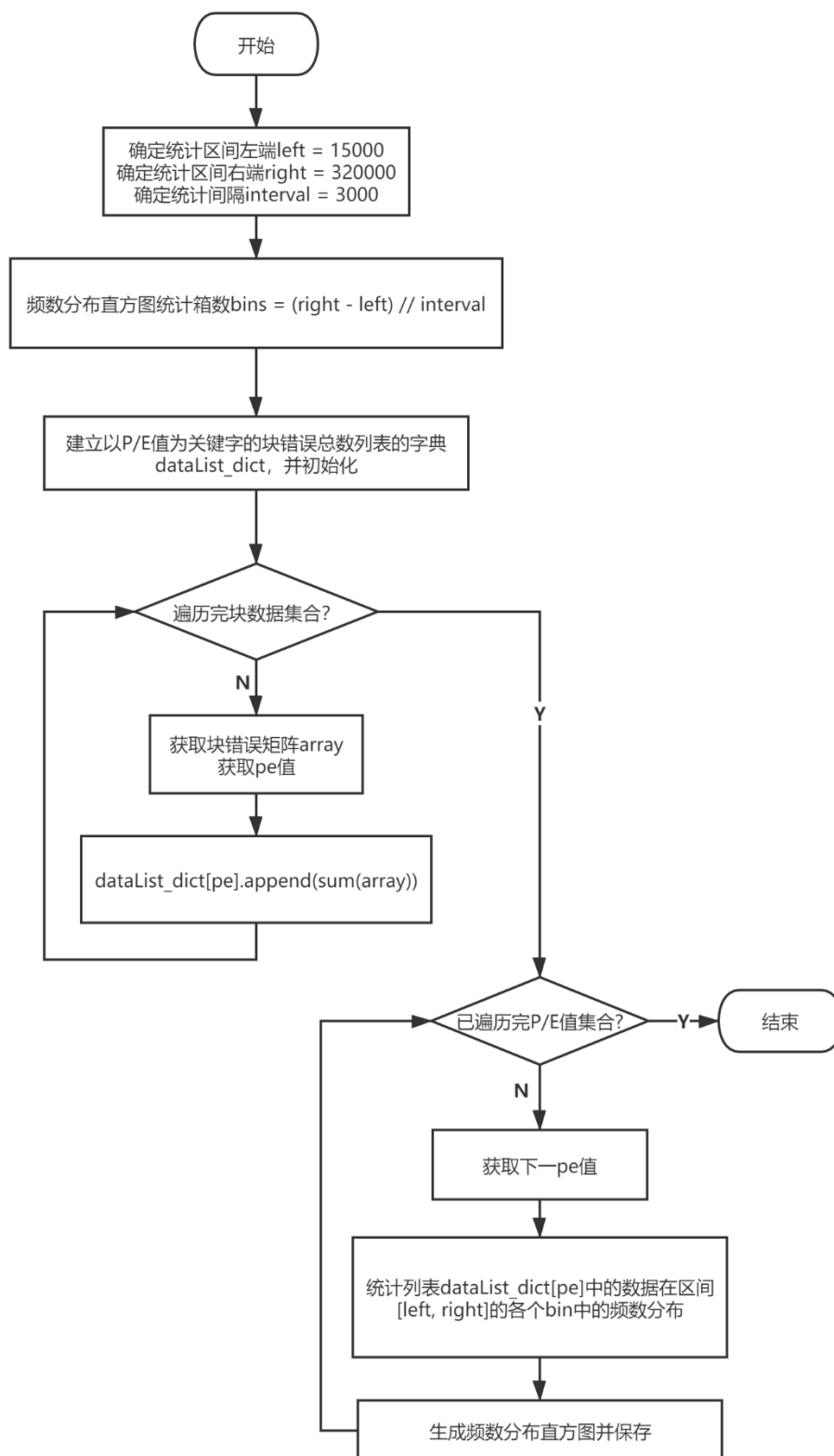


图 4-5 块错误总数频数分布直方图生成流程

4.4 比特错误相对分布生成模块

4.4.1 数据集定义

由于块相对错误分布生成模块的目的为生成块内部各页的相对错误分布，因此该模块所用到的数据为块的各页错误总数向量和块对应的 P/E 周期数。图 4-6 展示了输入数据的格式，每条数据尺寸为(2304,)，即包含了每页的错误总数，共 2304 页。

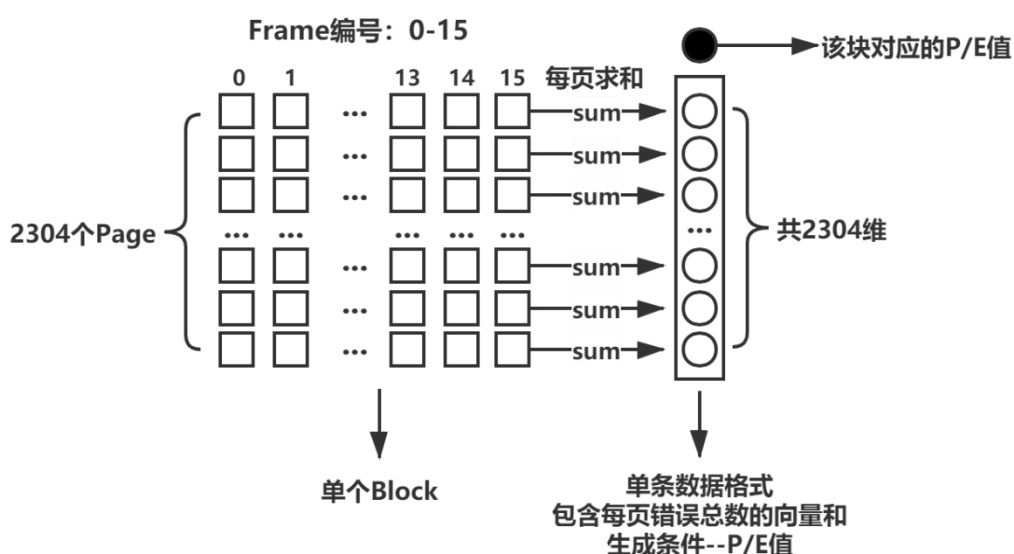


图 4-6 单条数据形式

为使模型学习到错误次数的相对分布，需要对数据集进行一定的处理。数据集中的每条数据均为尺寸(2304,)的矩阵，记录了每页的错误总数，将矩阵中的每个值除以该矩阵中的最大值，意味着将错误次数矩阵转换为了错误相对分布矩阵。

$$martix' = \frac{martix}{martix.max 0} \quad (4-2)$$

矩阵中的每个浮点值表示了该页的错误总数相对于页错误总数最大值的比率，称其为相对错误比值，取值范围为[0.0, 1.0]，当某页对应的值为 1.0 时，表明该页的错误次数为其所在块内的最大值。

其次为方便模型进行训练，需对数据集进行正则化，处理公式为：

$$martix' = \frac{(martix - mean)}{std} \quad mean = 0.5, std = 0.5 \quad (4-3)$$

最终矩阵内元素的取值范围为[-1, 1]。

4.4.2 模型定义

选用条件生成对抗网络(CGAN)来学习块错误相对分布。CGAN 模型分为两部分，生成器与判别器，如图 4-7 所示，为模型的整体结构示意图。

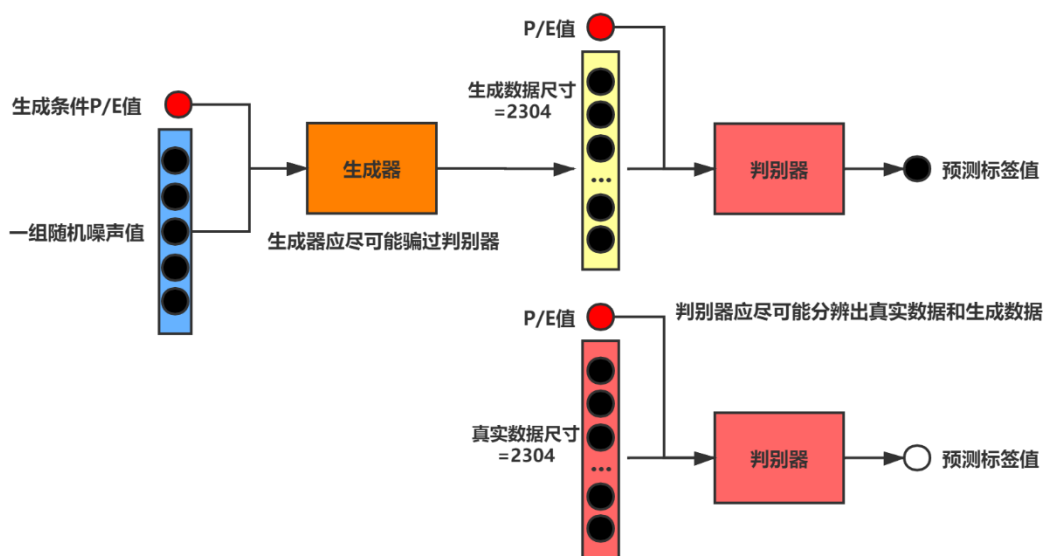


图 4-7 模型总体架构图

a. 生成器定义

生成器的输入端接受一组随机噪音值 z 和该块的生成条件 P/E 值（P/E 值需要经过归一化和正则化）。输出一尺寸为 2304×1 的浮点型矩阵。其内部的具体结构如图 4-8 所示，为全连接神经网络，选择 LeakyReLU 作为激活函数，系数设置为 0.2；在线性层与激活层之间添加批归一化层用于加快模型收敛速度，且在一定程度上缓解了梯度弥散的问题，使得模型的训练过程更加稳定。最终输出层选择 Tanh 函数将生成结果取值限制在[-1, 1]。

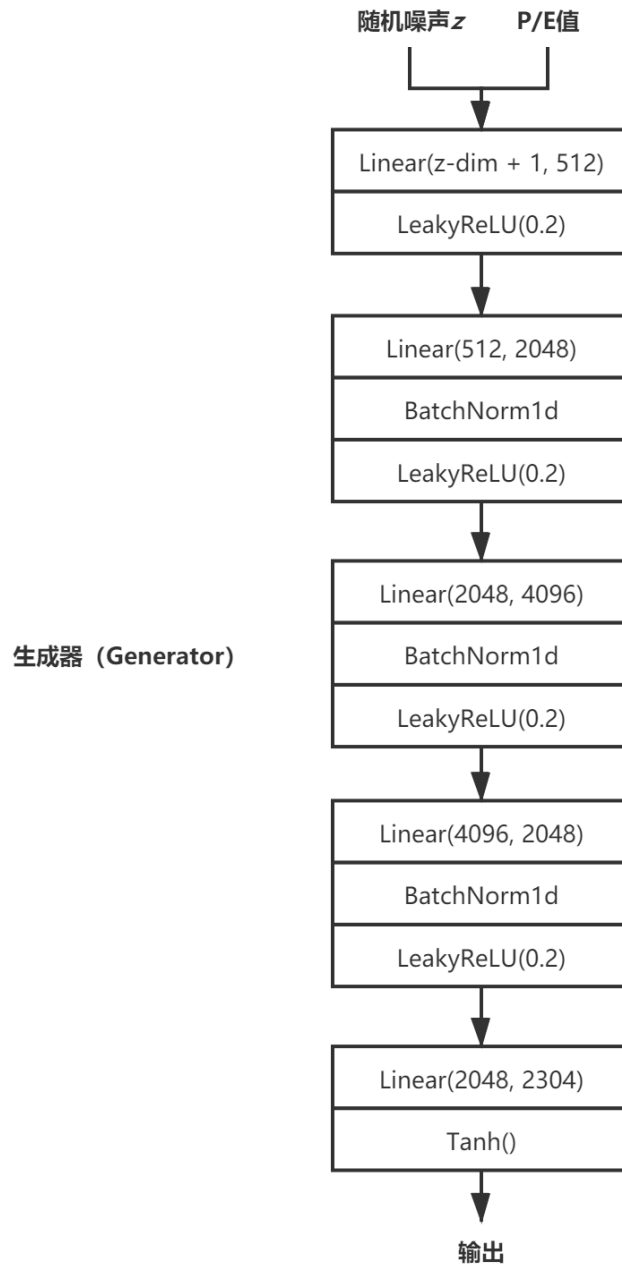


图 4-8 生成器

b. 判别器定义

判别器的具体结构如图 4-9 所示，其输入端接受块数据与 P/E 值，输出判别结果为单个值，“1”代表为真实数据，“0”代表为假数据。参考 Wasserstein GAN 网络，去掉判别器最后一层的 Sigmoid 函数；在线性层与激活函数之间添加 Dropout 层来防止过拟合；由于在 GAN 网络训练过程中易出现训练不均衡的现象，

通常是判别器的训练程度过高所导致，因此在设计时，判别器的层数与神经元数量要少于生成器，通过简化判别器来均衡训练过程。

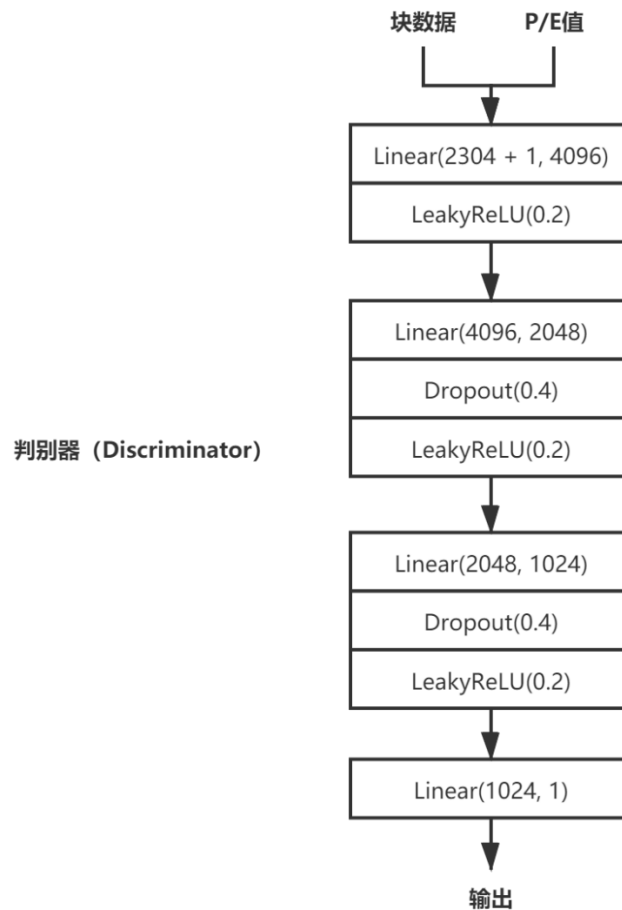


图 4-9 判别器

c. 训练过程

模型训练流程如图 4-10 所示，整体思路为利用生成器生成的假数据与来自真实数据集的数据训练判别器，再从判别器对生成数据的输出结果反向传播，训练生成器，两者交替训练，直到完成训练轮次。生成器与判别器的优化算法均选用 Adam 算法，损失函数则使用均方损失函数 (MSELoss)。

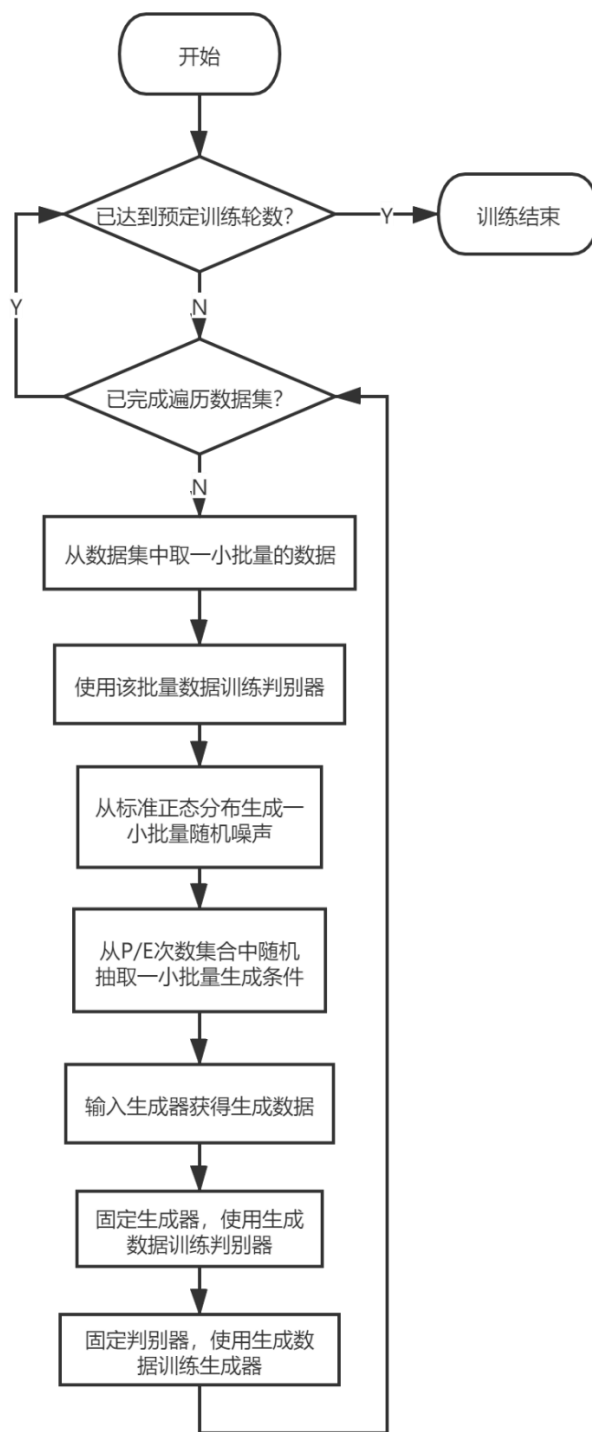


图 4-10 训练流程

4.5 比特错误总数生成模块

根据数据统计与分析的结果，使用正态分布来拟合块的错误总数在不同 P/E

条件下的真实分布。

a. 统计与记录

对于拥有的全部块错误数据，如图 4-11 所示，按 P/E 值进行分类，分为若干个集合，统计每个集合内块错误总数的平均值与标准差，并与其对应的 P/E 值一同保存下来。

b. 生成块错误总数

由于数据集所收集的块记录对应的 P/E 集合为 $\{1, 1000, 2000, \dots, 16000, 17000\}$ ，或者是以其他间隔值来确定 P/E 集合，总之所统计的均值与标准差为有限个 P/E 值对应的结果，例如若要针对 P/E 等于 1200 的条件进行生成，就无法从统计结果中获得确切的值。因此考虑使用三次样条插值，如图 4-12 所示，以统计所得的有限个点获得拟合函数，即以点集 $\{(P/E_1, mean_1), (P/E_2, mean_2), \dots, (P/E_n, mean_n)\}$ 进行插值，获得 $P/E \rightarrow mean$ 的拟合函数，标准差则同理；最终若要生成的 P/E 值不在统计范围内，则用拟合函数对应的值代替。

设某 P/E 值对应的均值为 $mean$ ，标准差为 std ，根据之前的分析结果，该 P/E 条件下使用正态分布 $X \sim N(mean, (0.8 \times std)^2)$ 来拟合块错误总数的真实分布。使用 Python 中的标准正态分布 $X \sim N(0, 1)$ 来生成一组值 y ，则 $(y \times \sigma + \mu)$ 即为正态分布 $X \sim N(\mu, \sigma^2)$ 的对应值。

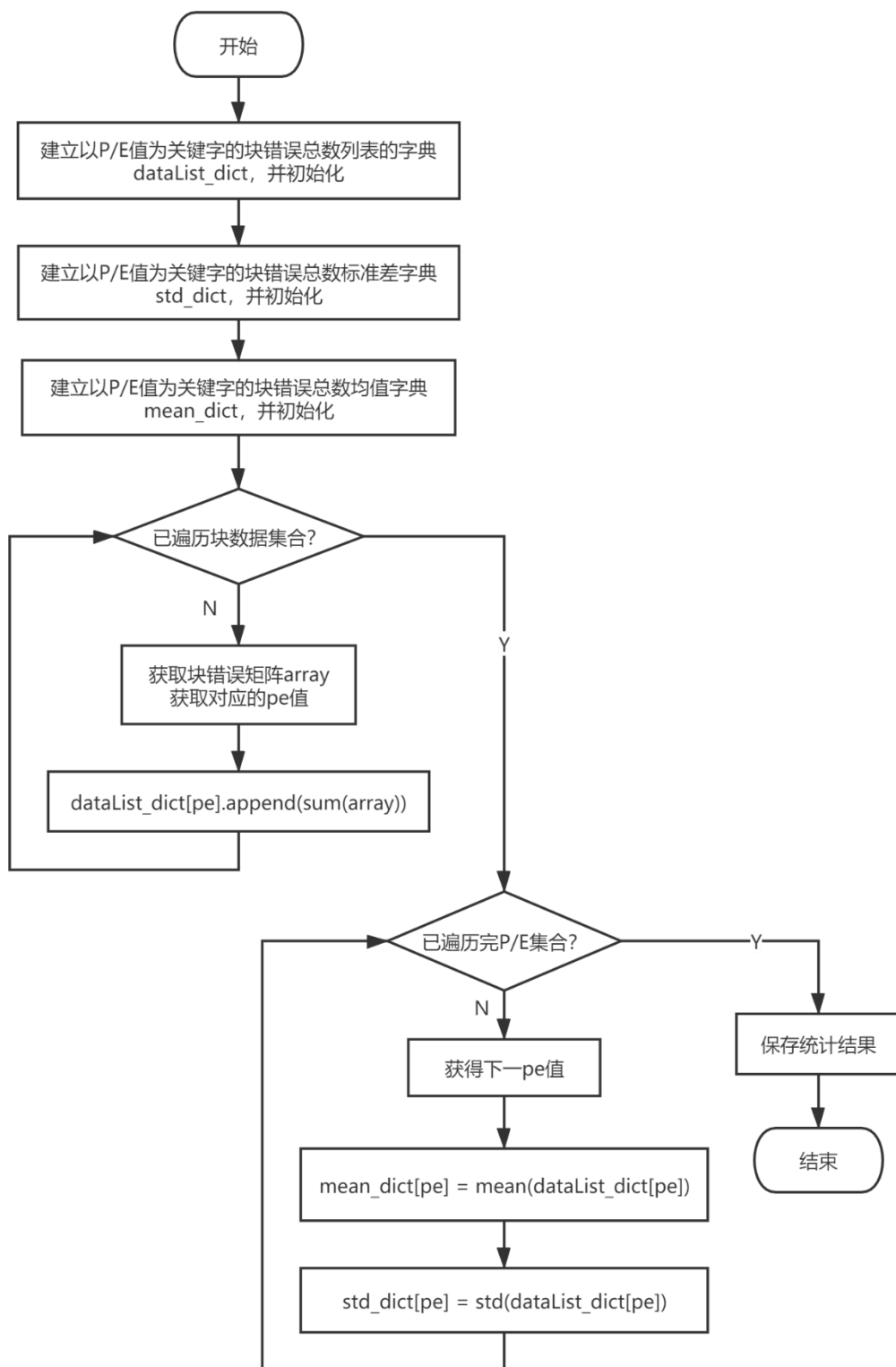


图 4-11 统计过程

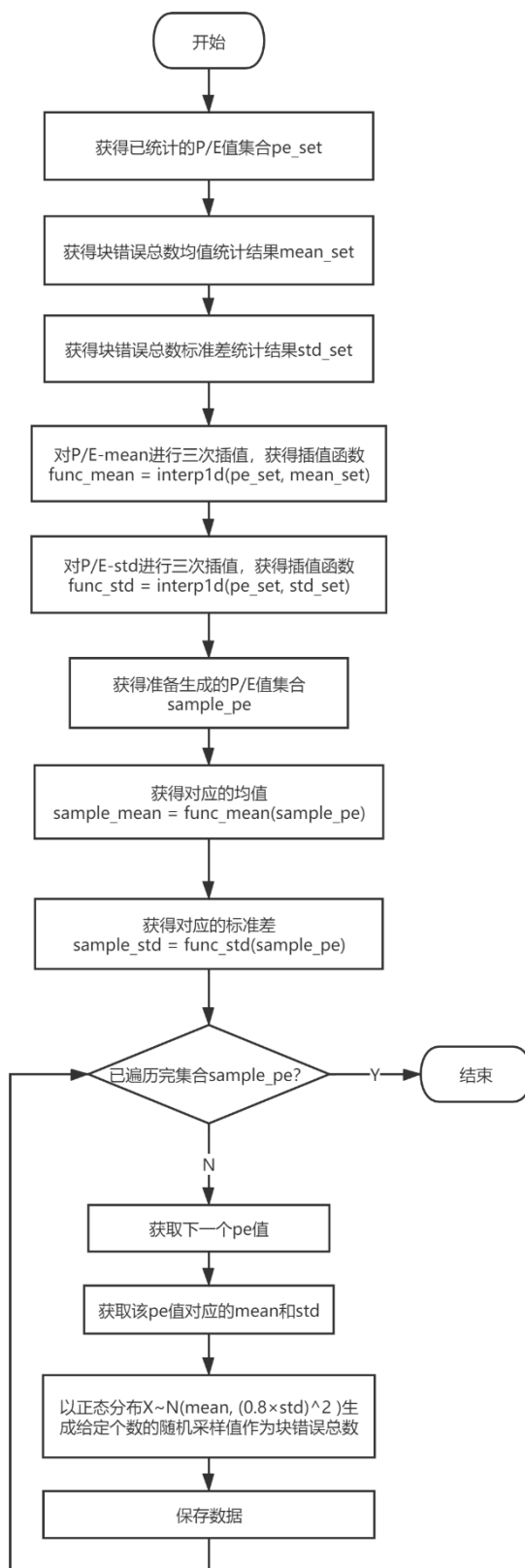


图 4-12 块错误总数生成过程

4.6 结果整合

设通过 CGAN 模型的生成器在给定的 P/E 条件下生成了一个块的各页相对分布矩阵 $\{r_0, r_1, \dots, r_{2302}, r_{2303}\}$, 第 i 页的相对错误比值即为 r_i ; 将 P/E 值代入插值函数中, 获得对应的均值和标准差, 从而获得拟合块错误总数分布的正态分布, 根据正态分布获得块错误总数 E , 因此可知第 i 页的错误总数为:

$$Page_Error_i = \frac{r_i}{\sum_{k=0}^n r_k} \times E, n = 2303 \quad (4-4)$$

根据之前的统计分析结果, 页内的 16 个区域的错误分布情况符合均匀分布的特点, 因此根据公式(4-4)可获得每页的错误总数 $Page_Error_i$, 再生成和为 $Page_Error_i$ 的 16 个均匀分布的随机数作为页内 $f_0 - f_{15}$ 每个 Frame 的错误次数, 具体操作如下:

- a. 在区间 $[0, Page_Error_i]$ 生成 15 个均匀分布的随机整数;
- b. 对之前生成的 15 个随机数升序排列, 得到分割点集 $\{s_1, s_2, \dots, s_{15}\}$;
- c. 将 0 和 $Page_Error_i$ 添加入分割点集, 得到 $\{s_0, s_1, s_2, \dots, s_{15}, s_{16}\}$, 其中 $s_0 = 0, s_{16} = Page_Error_i$;
- d. 16 个和为 $Page_Error_i$ 的均匀分布随机整数如下:

$$\{s_{i+1} - s_i | 0 \leq i \leq 15\}$$

至此获得一个完整的块错误数分布数据。

4.7 本章小结

本章详细说明了整个系统各个模块的具体实现, 给出了系统核心条件生成对抗网络的生成器与判别器的结构, 并给出了对模型训练的流程图。说明了对于页相对错误分布, 错误总数的结果整合过程, 最终如何生成完整的块错误数据。对于系统的各辅助模块的实现进行了阐述, 表明了这些辅助模块在整个系统中所发挥的作用。

5 测试与分析

上一章对系统的具体实现进行了阐述，本章则将对系统的生成效果进行测试，来判断模型的表现如何。分别从块错误总数和块错误相对分布这两个层面对生成数据与真实数据进行比对，进而得出结果。

5.1 测试环境

将系统部署在远程服务器端进行测试，服务器配置如下：

表 5-1 服务器配置

属性	信息
CPU	I5-3470
Memory	16GB
GPU	GTX 1070ti
OS	Ubuntu 18.04 64 位

5.2 闪存仿真器数据生成测试

5.2.1 文本解析模块测试

原始的记录文件分为多个文件夹，每个文件夹以测试日期命名，文件夹内包含多个记录文件，每个文件包含一个 chip 的块记录，文件以 chip 编号命名，例如：“001.log”。编写测试程序，在最外层遍历所有的文件夹，在每个文件夹内遍历每一个记录文件，将一个文件夹内所有的记录文件所提取出的块保存为三个 npy 文件，分别记录块的错误矩阵，块的各页错误总数矩阵，块的测试条件。再对所有文件夹的块的错误矩阵，块的各页错误总数矩阵，块的测试条件进行合并，最终得到三个 npy 格式的文件，保存了所得到的所有块的上述三类信息。文件解析过程主要脚本如下，在第四章系统实现部分便详细介绍了 handle_file 函数的具体运行流程。

```
for file in file_list:
```

```
data_cur_path = log_file_path + file + "/"
for chip in chip_list:
    """插入数据"""
    file_name = str(chip).zfill(3) + ".log"
    handle_file(data_cur_path + file_name, chip, connect, testID, action, p
e_interval)
    block_err_data_list.clear()
    page_err_data_list.clear()
    pe_data_list.clear()
```

数据合并主要脚本:

```
result = np.load(data_root_path + file_prefix + "_" + file_list[0] + ".np
y").astype(np.float32)
file_list.remove(file_list[0])

for file in file_list:
    result = np.concatenate((result,
                                np.load(data_root_path + file_prefix + "_" +
file + ".np
y").astype(np.float32)),
                                axis=0)
```

5.2.2 错误相对分布生成测试

测试流程如图 5-1 所示, 先对模型进行对抗训练, 再使用训练成熟的生成器生成假数据并与真实数据进行对比。

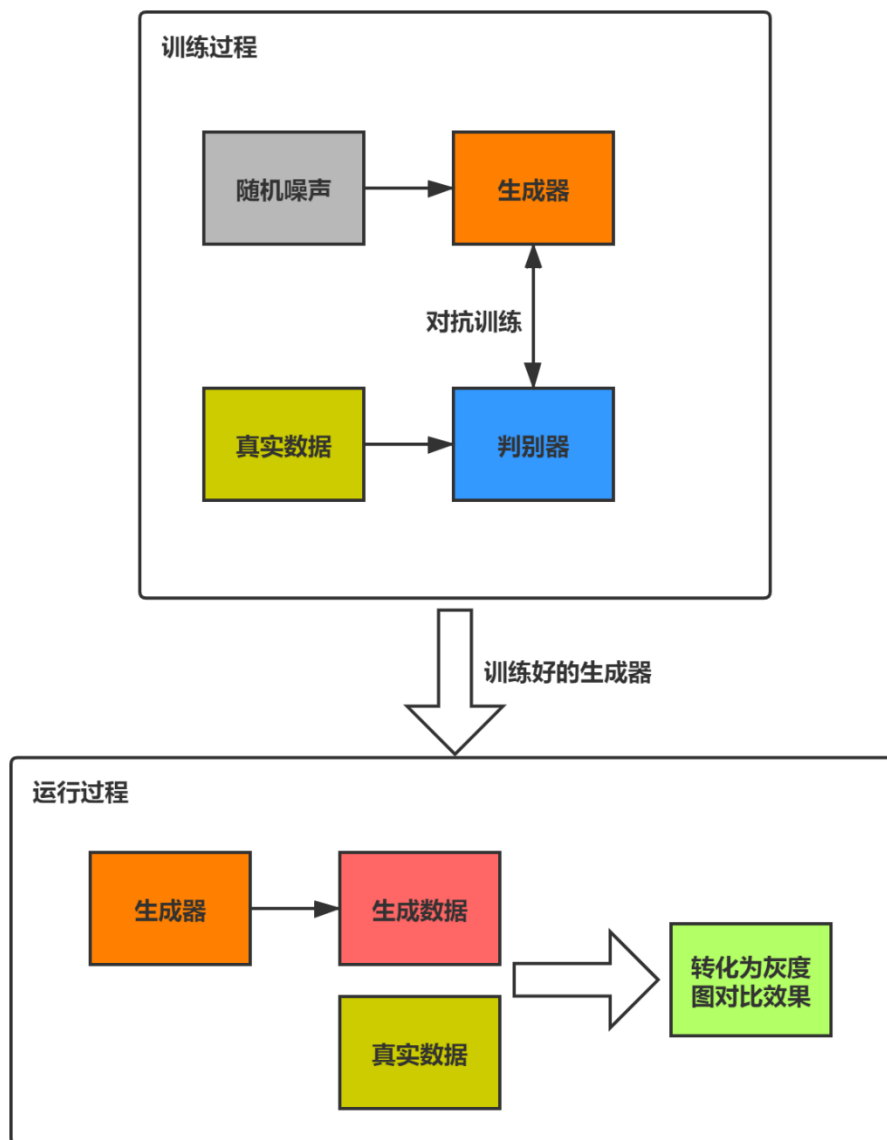


图 5-1 错误相对分布生成测试图

a. 模型训练

模型训练脚本如下：

```
import os
command = "python ./model_cGAN/probability_distributions_gen.py --train \  
--cuda \  
--latent_dim 20 \  
--epochs 100 \  
--batch_size 64 \  
"
```

```
--err_data_name data.npy \
--condition_data_name condition.npy \
--save_model_epoch 20"
```

```
os.system(command)
```

表 5-2 解释了脚本中各项参数的含义和取值。

表 5-2 训练参数表

脚本参数名	解释	设置值
--train	控制模型为训练模式	True
--cuda	控制是否使用 GPU	True
--latent_dim	控制生成器输入端噪声值维度	20
--epochs	控制总训练轮次	100
--batch_size	控制一个数据 batch 的尺寸	64
--err_data_name	保存块错误数据的文件名	data.npy
--condition_data_name	保存测试条件的文件名	condition.npy
--save_model_epoch	控制每次保存模型的训练轮次间隔	20

使用采集得到的 62182 个块错误数据集对模型进行训练，指定模型训练过程中的各项参数，如随机噪音维度设置为 20，训练轮次为 100 轮，投入训练的一个数据批次大小为 64，每隔 20 轮保存一次模型。保存的模型将用于数据生成，并与真实数据做对比。

b. 模型运行

经过对模型的训练，保存了不同训练轮次结束后的模型参数，将每个轮次的模型参数加载入模型，控制生成器产生数据进行效果比对，模型运行脚本如下：

```
import os
```

```
command = "python ./model_cGAN/probability_distributions_gen.py --eval \
--cuda \
--latent_dim 20 \
--g_load_model_path generator_epoch_%s.pth \
--d_load_model_path discriminator_epoch_%s.pth \
\
--gen_start_pe 0 \
--gen_end_pe 17000 \
--gen_interval_pe 500 \
--generator_data_num 200"

for epoch in range(20, 120, 20):
    os.system(command % (epoch, epoch))
```

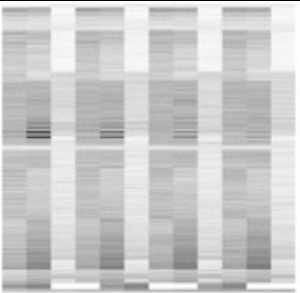
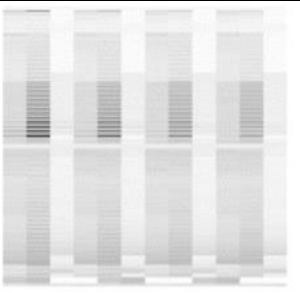
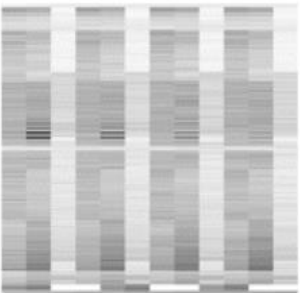
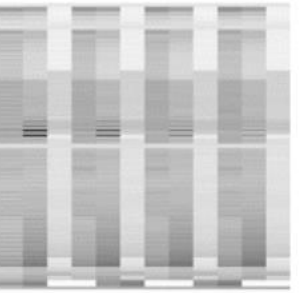
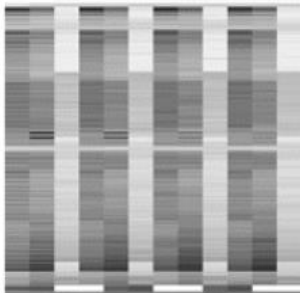
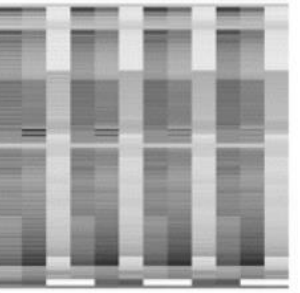
表 5-3 给出了脚本中各项参数的具体含义。

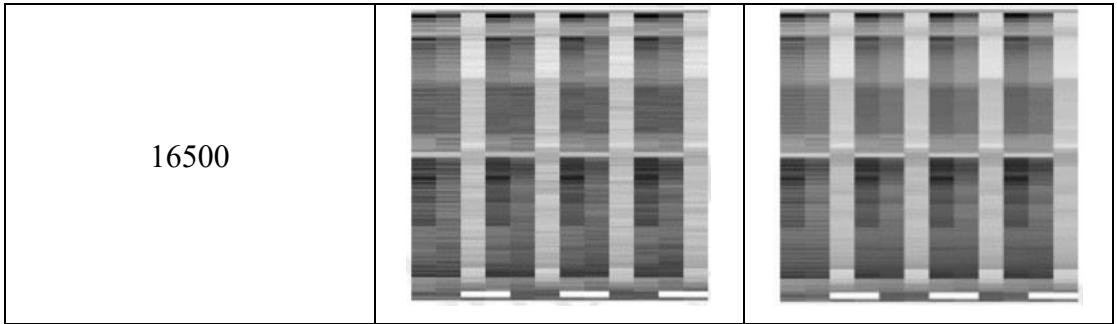
表 5-3 运行参数表

脚本参数名	解释	设置值
--eval	控制当前模型处于运行状态	True
--cuda	控制是否使用 GPU 设备	True
--latent_dim	控制生成器输入端噪声值维度	20
--g_load_model_path	训练好的生成器参数保存地址	依次加载训练轮次为 20, 40, 60, 80, 100 的模型
--d_load_model_path	训练好的判别器参数保存地址	依次加载训练轮次为 20, 40, 60, 80, 100 的模型
--gen_start_pe	待生成 P/E 值集合的最小值	0
--gen_end_pe	待生成 P/E 值集合的最大值	17000
--gen_interval_pe	待生成 P/E 值集合相邻值的间隔	500
--generator_data_num	每个 P/E 值生成的数据个数	200

如以上脚本，分别把训练 20，40，60，80，100 轮的模型参数载入模型，对于 P/E 值起始于 0，结束于 17000，间隔为 500 的生成条件集合 {0, 500, 1000, ... 15500, 16000, 16500} 中的每个生成条件均产生 200 个数据并保存。生成数据取值范围处于 [-1, 1] 之间，需要通过归一化将其转换到 [0, 1] 之间，之后得到生成的相对分布矩阵，将其转换为相对分布灰度图，并与真实数据的相对分布灰度图进行对比，训练 60 轮次的模型效果最好，效果对比如表 5-4 所示。

表 5-4 错误次数相对分布生成效果

P/E 周期	生成块数据灰度图	真实块数据灰度图
0		
1000		
6000		



5.2.3 错误总数生成测试

错误总数生成测试流程如图 5-2 所示，其训练过程主要用于对数据集进行数学统计并保存统计结果；生成过程利用统计结果构造插值函数，进一步获得拟合真实分布的正态分布，来进行随机采样，获得块错误总数生成数据。最终绘制生成数据与真实数据的频数分布直方图进行对比。

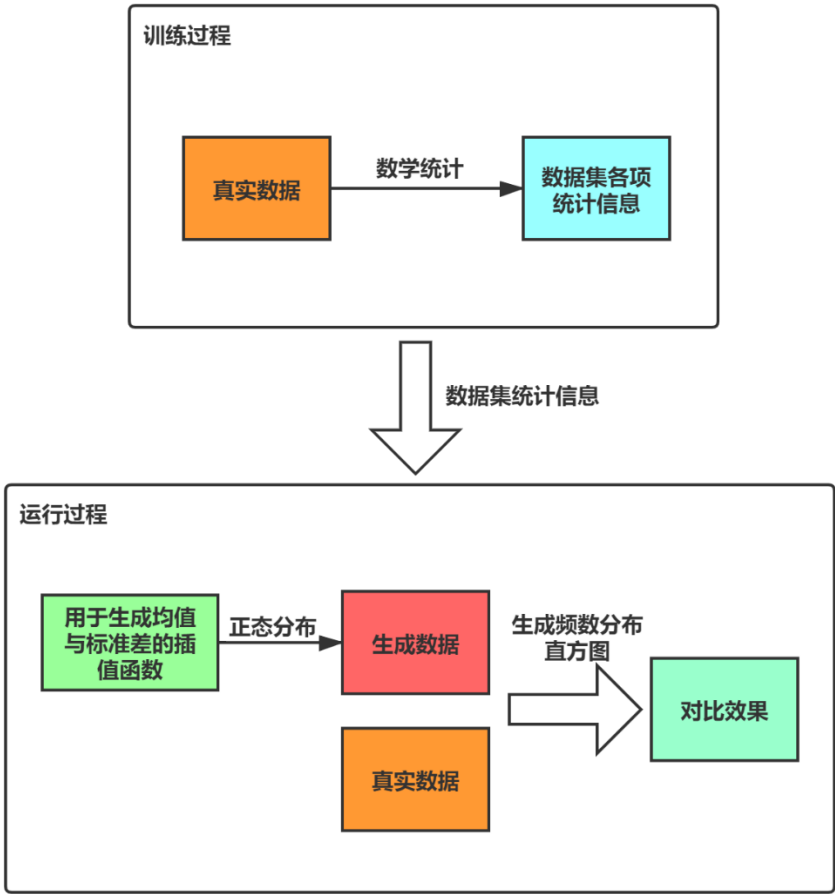
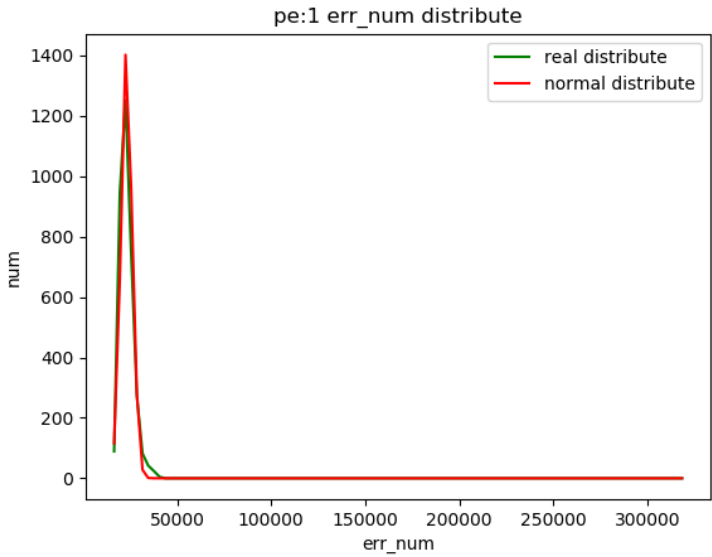
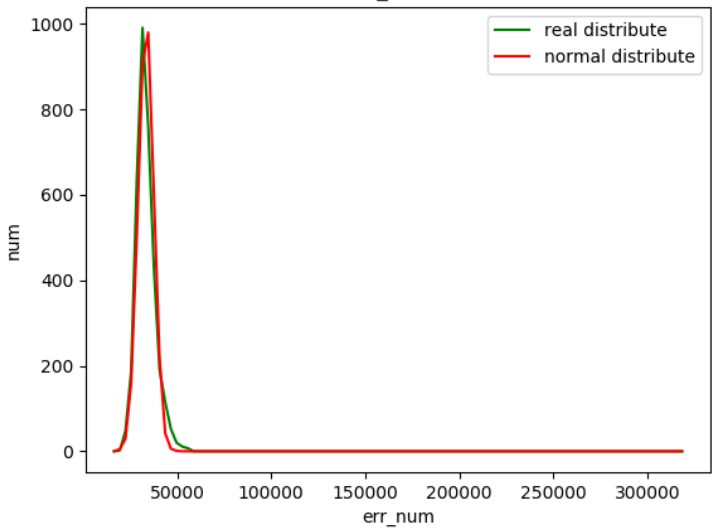


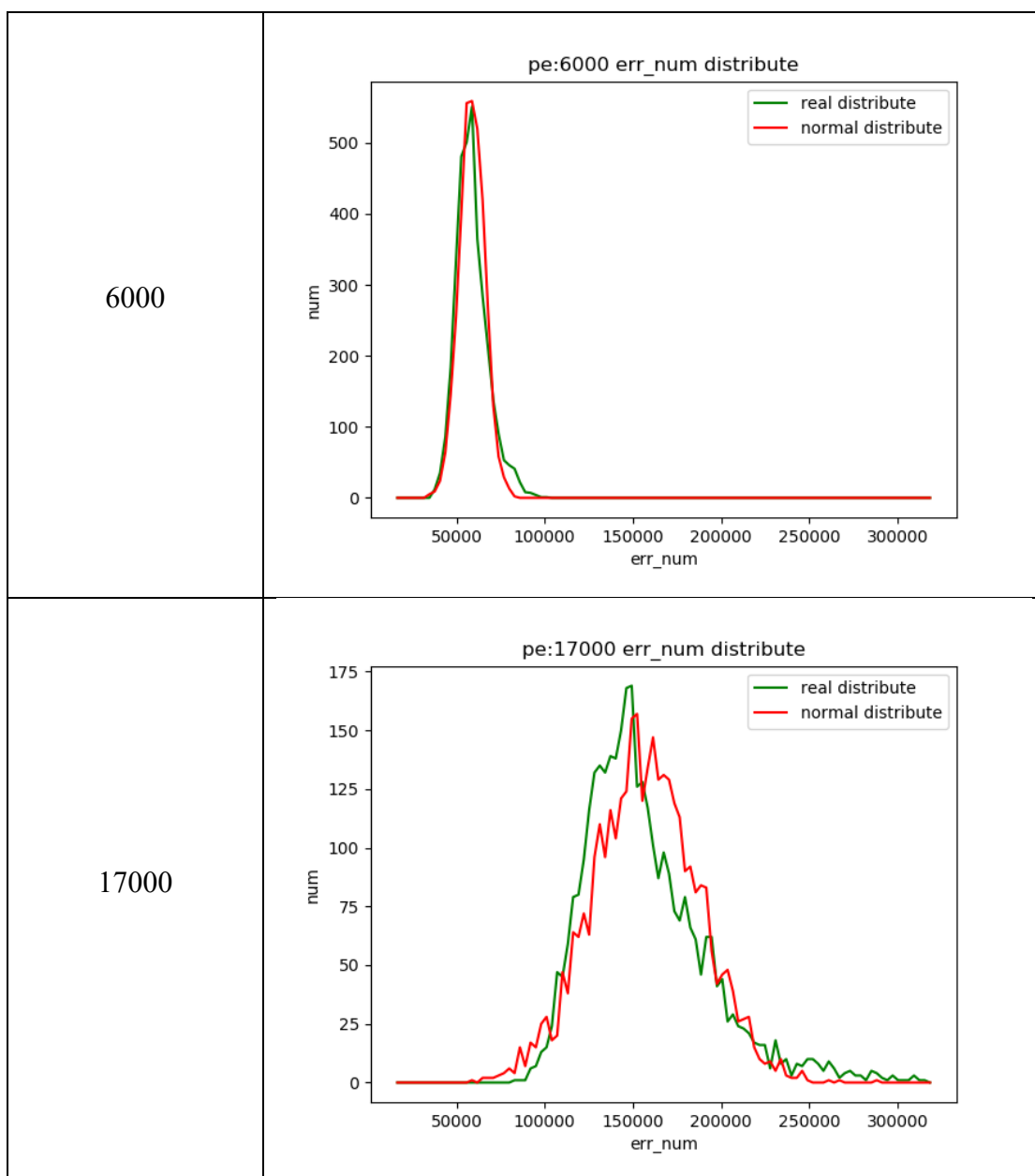
图 5-2 块错误总数生成测试图

对于生成条件集合 $\{1, 1000, 2000, \cdots 16000, 17000\}$ 的每一个生成条件，利

用块错误总数生成模块，生成 3000 个块错误总数，之后并绘制出其频数分布直方图，并与真实数据的频率分布直方图进行对比，如表 5-5 展示了 P/E 值为 1, 1 000, 6000, 17000 时的对比效果，绿色为真实块数据分布，红色为正态分布生成随机数的分布，可见正态分布对真实数据分布拟合得很好。

表 5-5 块错误总数生成效果

P/E 周期	真实数据分布与生成数据分布
1	
1000	



5.3 本章小结

本章对于整个系统进行测试，利用条件生成对抗网络的生成器生成一定数量的错误相对分布矩阵，利用正态分布生成一定数量的错误总数，分别与真实数据进行对比，来测试整个系统的生成效果。可以看出，条件生成对抗网络对于块的错误相对分布的学习比较成功；以正态分布所得到的错误总数与真实数据分布拟合的很好。

6 总结与展望

对于闪存的测试与错误数据的收集，在闪存特性分析，闪存寿命预测等领域有着重要的应用。而对真实闪存介质进行测试记录需要耗费大量的时间，因此提出了使用机器学习领域的生成对抗网络来构造闪存仿真器以批量生成闪存错误数据，并做了以下工作：

1) 对原始闪存记录文件进行解析，提取出了易于计算机处理的闪存块错误次数矩阵。

2) 对闪存错误数据进行统计与分析，抽取了闪存块错误分布的各项特征，对其错误分布进行了一个初步的预判，用以指导整个系统的方案设计。

3) 提出了闪存页相对错误分布的概念，并利用条件生成对抗网络（CGAN）和真实数据集进行训练，最终得到了可产生媲美真实数据相对分布的生成器。

4) 通过对数据的整体分析，提出了使用正态分布来拟合闪存块错误总数真实分布的方案。

5) 对以上方案进行整合，得到了可根据生成条件产生完整块错误数据的闪存仿真器。

除了已完成的方案，还有一些工作正待完成，包括：

1) 目前的仿真器仅接受 P/E 这一个主要生成条件，可以考虑当有多个生成条件时的情况。

2) 方案论证部分的方案 2 未经实现，在模型中使用了卷积神经网络，同时通过将数据转换为 16 通道，避免了页分布与页内分布的分开处理。

致 谢

论文完成之际，我要感谢我的导师吴非教授，她在毕业设计的选题，研究和论文撰写上全程参与，并给予了很大帮助，提出了许多建设性的意见。正是吴非老师提供的服务器资源与数据资源使得我的研究工作能够顺利开展并取得一些成果；同时也要感谢实验室中的学长的在细节上悉心指导，他对于工程上的一些经验的传授，使得我少走了许多弯路，更快地完成了编码工作。在学术上，为我推荐了许多与课题相关的优秀论文，为我的问题解决指明了正确的方向；还要感谢我的母校华中科技大学四年来的悉心栽培，正是在本科阶段那些基础课程打下的扎实基础，同时培养了我计算机学科的思维方式，为最终的毕业设计提供了知识储备与技术储备；也要感谢我的父母，他们含辛茹苦将我抚养成人，悉心教导，支持我接受高等教育，在大学这个平台垫起人生的新高度；最后要感谢国家维护了一个稳定向上的社会环境使得我们可以学习知识和开展研究工作。

参考文献

- [1] 中华人民共和国工业和信息化部, 中华人民共和国国家发展与改革委员. 信息产业发展指南. 2016.
- [2] David Reinsel, 武连峰, John F. Gantz, John Rydning. IDC: 2025 年中国将拥有全球最大的数据圈[EB/OL].<https://www.seagate.com/files/www-content/our-story/trends/files/data-age-china-regional-idc.pdf>, 2019-1.
- [3] Bryan S. Kim, Jongmoo Choi, and Sang Lyul Min. 2019. Design tradeoffs for SSD reliability. In Proceedings of the 17th USENIX Conference on File and Storage Technologies (FAST'19). USENIX Association, USA, 281–294.
- [4] F. Wu et al., "Characterizing 3D Charge Trap NAND Flash: Observations, Analyses and Applications," 2018 IEEE 36th International Conference on Computer Design (ICCD), Orlando, FL, USA, 2018, pp. 381-388.
- [5] N. R. Mielke, R. E. Frickey, I. Kalastirsky, M. Quan, D. Ustinov and V. J. Vasudevan, "Reliability of Solid-State Drives Based on NAND Flash Memory," in Proceedings of the IEEE, vol. 105, no. 9, pp. 1725-1750, Sept. 2017.
- [6] Y. Cai, E. F. Haratsch, O. Mutlu and K. Mai, "Threshold voltage distribution in MLC NAND flash memory: Characterization, analysis, and modeling," 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2013, pp. 1285-1290.
- [7] Y. Cai, O. Mutlu, E. F. Haratsch and K. Mai, "Program interference in MLC NAND flash memory: Characterization, modeling, and mitigation," 2013 IEEE 31st International Conference on Computer Design (ICCD), Asheville, NC, 2013, pp. 123-130.
- [8] Y. Cai, Y. Luo, E. F. Haratsch, K. Mai and O. Mutlu, "Data retention in MLC NAND flash memory: Characterization, optimization, and recovery," 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA), Burlingame, CA, 2015, pp. 551-563.
- [9] Y. Cai, Y. Luo, S. Ghose and O. Mutlu, "Read Disturb Errors in MLC NAND Flash Memory: Characterization, Mitigation, and Recovery," 2015

- 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Rio de Janeiro, 2015, pp. 438-449.
- [10] Y. Cai, S. Ghose, E. F. Haratsch, Y. Luo and O. Mutlu, "Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives," in *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1666-1704, Sept. 2017.
 - [11] B. Schroeder, A. Merchant and R. Lagisetty, "Reliability of nand-Based SSDs: What Field Studies Tell Us," in *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1751-1769, Sept. 2017.
 - [12] Y. Luo, S. Ghose, Y. Cai, E. F. Haratsch and O. Mutlu, "HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature Awareness," 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, 2018, pp. 504-517.
 - [13] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Nets. In *NIPS'2014*.
 - [14] 周志华.机器学习[M].清华大学出版社:北京,2016.
 - [15] bonelee.GAN 的原理入门[EB/OL].<https://www.cnblogs.com/bonelee/p/9166084.html>,2018-6-11.
 - [16] 博文视点.GAN: 实战生成对抗网络[M].电子工业出版社:北京,2018.
 - [17] lqfarmer.历史最全 GAN 网络及其各种变体整理[EB/OL].<https://zhuanlan.zhihu.com/p/34016536>,2018-2-26.
 - [18] Radford, A., Metz, L. and Chintala, S. (2015). Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks. *arXiv:1511.06434*.
 - [19] 伊恩·古德费洛, 约书亚·本吉奥, 亚伦·库维尔.深度学习[M].人民邮电出版社:北京,2017.
 - [20] Arjovsky, M., Chintala, S. and Bottou, L. (2017). Wasserstein GAN. *arXiv:1701.07875*.
 - [21] 郑华滨.令人拍案叫绝的 Wasserstein GAN[EB/OL].<https://zhuanlan.zhihu.com>

m/p/25071913,2017-2-5.

- [22] J. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, 2017, pp. 2242-2251.
- [23] Mirza, M. and Osindero, S. (2014). Conditional Generative Adversarial Nets. arXiv:1411.1784.
- [24] SSDFans.深入浅出 SSD[M].机械工业出版社:北京,2018.
- [25] 郑泽宇, 顾思宇.Tensorflow: 实战 Google 深度学习框架[M].电子工业出版社:北京,2017.
- [26] 陈云.深度学习框架 PyTorch: 入门与实践[M].电子工业出版社:北京,2018.
- [27] Vishnu Subramanian.PyTorch 深度学习[M].人民邮电出版社:北京,2019.
- [28] Eric Matthes. Python 编程[M].人民邮电出版社:北京,2016.
- [29] Wesley J. Chun. Python 核心编程[M].人民邮电出版社:北京,2008.
- [30] Ben Forta.MySQL 必知必会[M].人民邮电出版社:北京,2009.