

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、保密囗，在 年解密后适用本授权书

2、不保密囗 。

（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

闪存块在使用过程中会产生多次错误，这些错误分布在块的每一个位置上，错误特性与闪存块所处使用条件有关；收集这些块错误数据对于闪存寿命预测和闪存错误特性分析都至关重要。而对闪存块进行测试并采集真实错误数据则需要耗费大量的时间。

生成对抗网络是近年来机器学习领域的一个极具创造性的成果。它通过生成器与判别器的对抗学习，来获得能力强大的生成器，用以生成足够以假乱真的假数据。本文从原始记录文件构造数据集，通过对真实数据集进行统计与分析，根据闪存错误分布特性设计方案：使用真实数据集对条件生成对抗网络进行训练，所得生成器可在不同的输入条件下生成块的各页相对错误分布；使用正态分布来生成块的错误总数，再与各页相对错误分布进行合并，并在页内使用均匀随机分布生成页内错误数据，最终获得可与真实数据相媲美的生成数据。

**关键词**：条件生成对抗网络；闪存；块；错误分布；数据生成；

Abstract

Flash blocks generate multiple errors during use. These errors are distributed at each position of the block. The error distribution characteristics are related to the conditions in which the flash blocks are used. Collecting these block error data is useful for flash life prediction and flash error analysis. Vital. It takes a lot of time to test the flash blocks and collect real error data.

Generative Adversarial Network is a very creative achievement in the field of machine learning in recent years. It obtains a powerful generator through the adversarial learning of the generator and the discriminator, which is used to get generated data comparable to real data. This paper constructs a data set from the original record file, and conducts statistics and analysis on the real data set, and designs a scheme based on the error distribution characteristics of the flash memory: using the real data set to train the Conditional Generative Adversarial Network. The resulting generator can generate the relative error distribution of each page of the block under different input conditions. Use the normal distribution to generate the total number of errors for the block. The total number of errors in the block is combined with the relative error distribution of each page, and the in-page error data is generated using a uniform random distribution within the page, and finally the generated data comparable to the real data is obtained.

**Keywords:** Conditional Generative Adversarial Network(CGAN), Flash Memory, Block, Error distribution, Generate data

目 录

[摘 要 I](#_Toc35509060)

[Abstract II](#_Toc35509061)

[1 绪 论 1](#_Toc35509062)

[1.1 课题背景 1](#_Toc35509063)

[1.2 国内外研究现状 2](#_Toc35509064)

[1.3 研究目的和主要内容 3](#_Toc35509065)

[1.4 论文结构 3](#_Toc35509066)

[1.5 课题来源 4](#_Toc35509067)

[2 方案论证 5](#_Toc35509068)

[2.1 系统需求分析 5](#_Toc35509069)

[2.2 开发工具分析及选择 5](#_Toc35509070)

[2.3 基本方案制定 6](#_Toc35509071)

[2.4 本章小结。 7](#_Toc35509072)

[3 系统设计 8](#_Toc35509073)

[3.1 功能需求 8](#_Toc35509074)

[3.2 系统总体设计 8](#_Toc35509075)

[3.3 本章小结 9](#_Toc35509076)

[4 系统实现 10](#_Toc35509077)

[4.1 文本解析模块 10](#_Toc35509078)

[4.2 数据库连接模块 11](#_Toc35509079)

[4.3 数据统计与分析模块 12](#_Toc35509080)

[4.4 块相对错误分布生成模块 15](#_Toc35509081)

[4.5 块错误总数生成模块 19](#_Toc35509082)

[4.6 结果整合 20](#_Toc35509083)

[4.7 本章小结 21](#_Toc35509084)

[5 测试与分析 22](#_Toc35509085)

[5.1 测试环境 22](#_Toc35509086)

[5.2 功能测试 22](#_Toc35509087)

[5.3 性能测试 22](#_Toc35509088)

[5.4 本章小结 22](#_Toc35509089)

[6 总结与展望 23](#_Toc35509090)

[致 谢 24](#_Toc35509091)

[参考文献 26](#_Toc35509092)

# 绪 论

本章介绍了本文的选题背景，说明了本设计课题的来源，目的，意义与应解决的主要问题及应达到的技术要求，分析了国内外研究现状及存在的问题。

## 课题背景

### 研究背景和趋势

生成对抗网络(GAN)是一种非监督式学习的方法，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。判别器接受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，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。

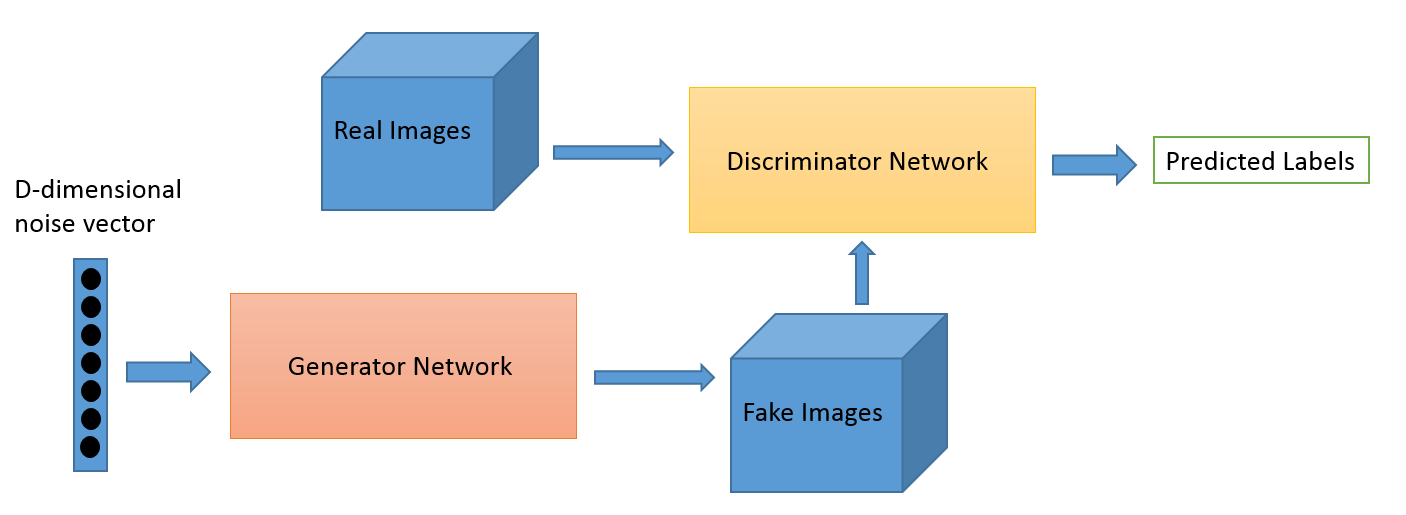


图1‑1生成对抗网络(GAN)

闪存在进行寿命预测以及其他工作的过程中，需要针对闪存在不同的测试条件下对闪存块的错误信息进行采集，这种针对真实闪存介质的测试与信息采集需要耗费较多的时间，因此考虑使用软件模拟仿真的方式来获得可以使用的闪存块错误信息数据。

### 面临的问题和挑战

测试所用的闪存，一个闪存块(Block)包含2304个页(Page)，每个页又被分为16个区域，对闪存块进行测试采样时，需要对每个块收集这230416个位置的错误次数，因此若想使用传统的方式对这生成一个块的2304个假数据较为困难。

原始的GAN网络本身则存在着许多问题，包括训练不稳定，需要小心的平衡生成器和判别器的训练程度，以防止其中一方的训练程度远超另一方（通常是判别器的训练程度较高）；生成器的生成样本容易缺乏多样性；并且生成器的结果由输入的随机值确定，因此输出结果具有随机性不易于控制。

## 国内外研究现状

### 生成对抗网络的改进与变种

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的变种，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(DCGAN)，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WGAN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，去掉了损失函数的log层，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），并且不再使用基于动量的优化算法，最终解决了GAN网络训练不稳定的问题，增加了生成样本的多样性。

Jun-Yan Zhu提出了循环对抗生成网络(CycleGAN)，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。

### 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(CGAN)，为GAN网络的一个变种，对于生成器，除了一组随机取样值*z*外，还需添加一组值*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，可以通过输入网络的标签值来控制产生数据的类型，有别于GAN网络随机生成假数据；

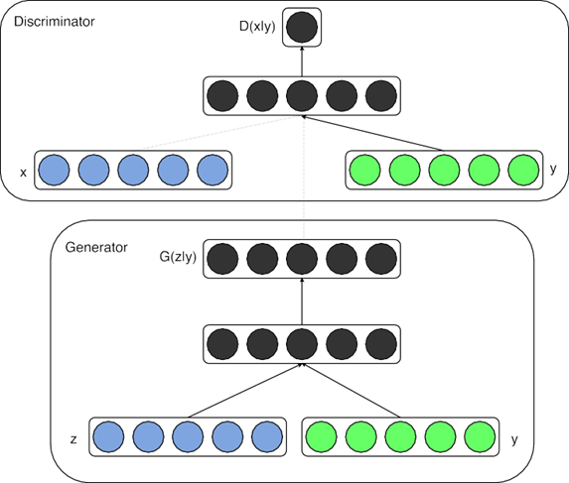


图1‑2条件生成对抗网络(CGAN)

## 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，它可以对不同特征进行定量组合，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。例如对于闪存寿命预测等问题需要针对闪存进行大量的测试来获得真实的块错误信息，进行这些测试需要花费大量时间，其次当一个块的P/E次数达到一定量时，该闪存块就会损坏。因此如果能使用仿真器生成的数据代替真实采集的数据，就可以节省采集数据所花费的时间。

## 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；介绍了相关背景知识包括闪存块的特性；生成对抗网络网络的结构，及其在不同应用条件下相关变种； 第二章则进行了方案论证，对闪存块数据进行了分析，设计了多种方案并进行分析，最终选择了一种方案进行实现，并根据需求选择了要使用的开发工具。

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，以及各个模块之间的相互关系。

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。

## 课题来源

该课题是本人在大四上学期在武汉光电国家研究中心实习时选择的课题，该课题由华中科技大学吴非带领研究。。

吴非老师实验室的学长已对闪存在不同条件下进行了测试，得到了大量的块错误数据记录文件，本设计便以这些记录文件为基础，通过对这些原始数据的提取整理，分析特征，进而设计整个系统来达到预期目的。

# 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。

## 系统需求分析

块（Block）作为闪存的一个基本存储单元，它由2304个页（Page）组成，每个页又被划分为16个区域。块在多次P/E的过程中，这2304 × 16个位置均会产生一定量的错误。现对闪存进行测试，得到了一系列闪存记录文件，其中详细记录了块的在闪存中的位置，测试条件，以及块中每个位置测错误次数等信息。从这些记录文件中取得大量的块在不同P/E次数下的错误信息组织成真实数据集。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2-1，为真实数据的形式，记录了在一定P/E次数下，一个闪存块每个位置的错误次数，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。

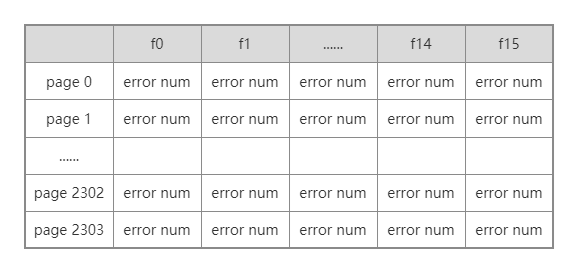


图2-1数据形式

## 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，Pytorch，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。

## 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：块错误数据维度为2304×16，判别器与生成器内部使用全连接神经网络，考虑直接将块错误数据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，原因在于数据的尺寸过大，使用全连接神经网络导致模型拥有大量的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48×48×16作为输入，它相当于分辨率为48×48的16通道图像，块错误数据中的分别为一个通道，而在page的维度2304组织成48×48。因此在CGAN的内部使用卷积神经网络，来处理这种具有图片特点的数据。

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，将所有的页错误数除以该块中的最大页错误数，得到该块所有页的相对错误分布（数据尺寸为(2304, )）。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）每个P/E次数都对应一个正态分布，使用该正态分布产生块的错误总数。

（3）整合（1）（2）得到块的各页错误总数，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。

## 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，首先对块各页错误总数的相对分布进行生成，之后生成块的错误总数，由这两步可得到各页的错误数量，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。

# 系统设计

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。

## 功能需求

（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，以及块每个位置的错误次数，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，在不同的P/E下，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。

## 系统总体设计

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，最终可得到错误相对分布模型，同时块错误生成模块可以生成块的错误总数，最终整合这些信息，得到最终的块错误数据。

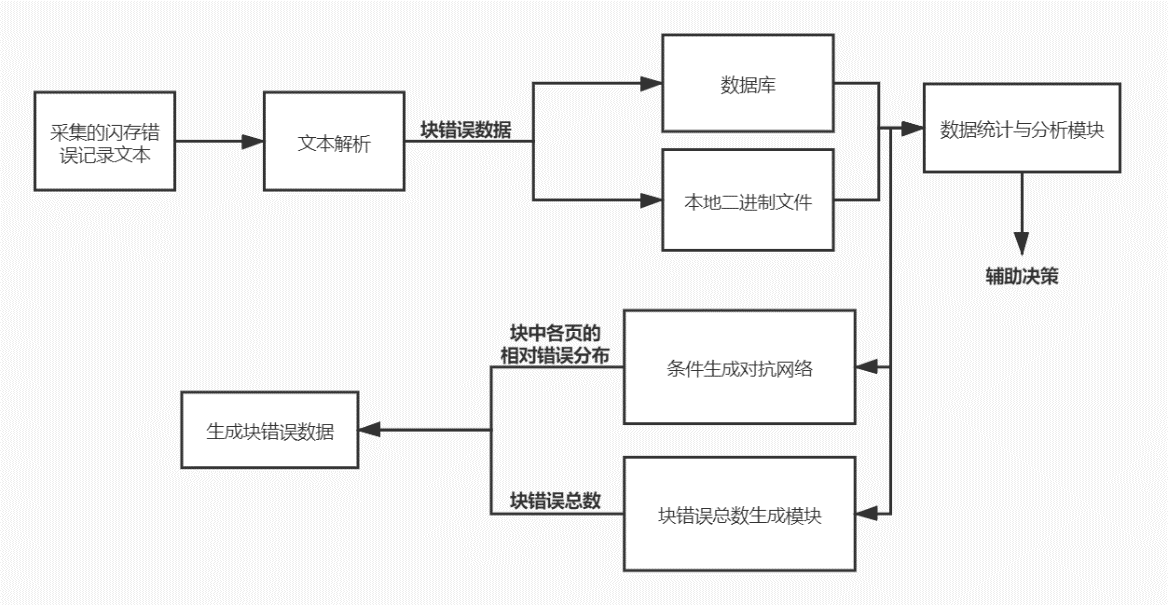


图3-1系统总体设计

## 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。

# 系统实现

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。

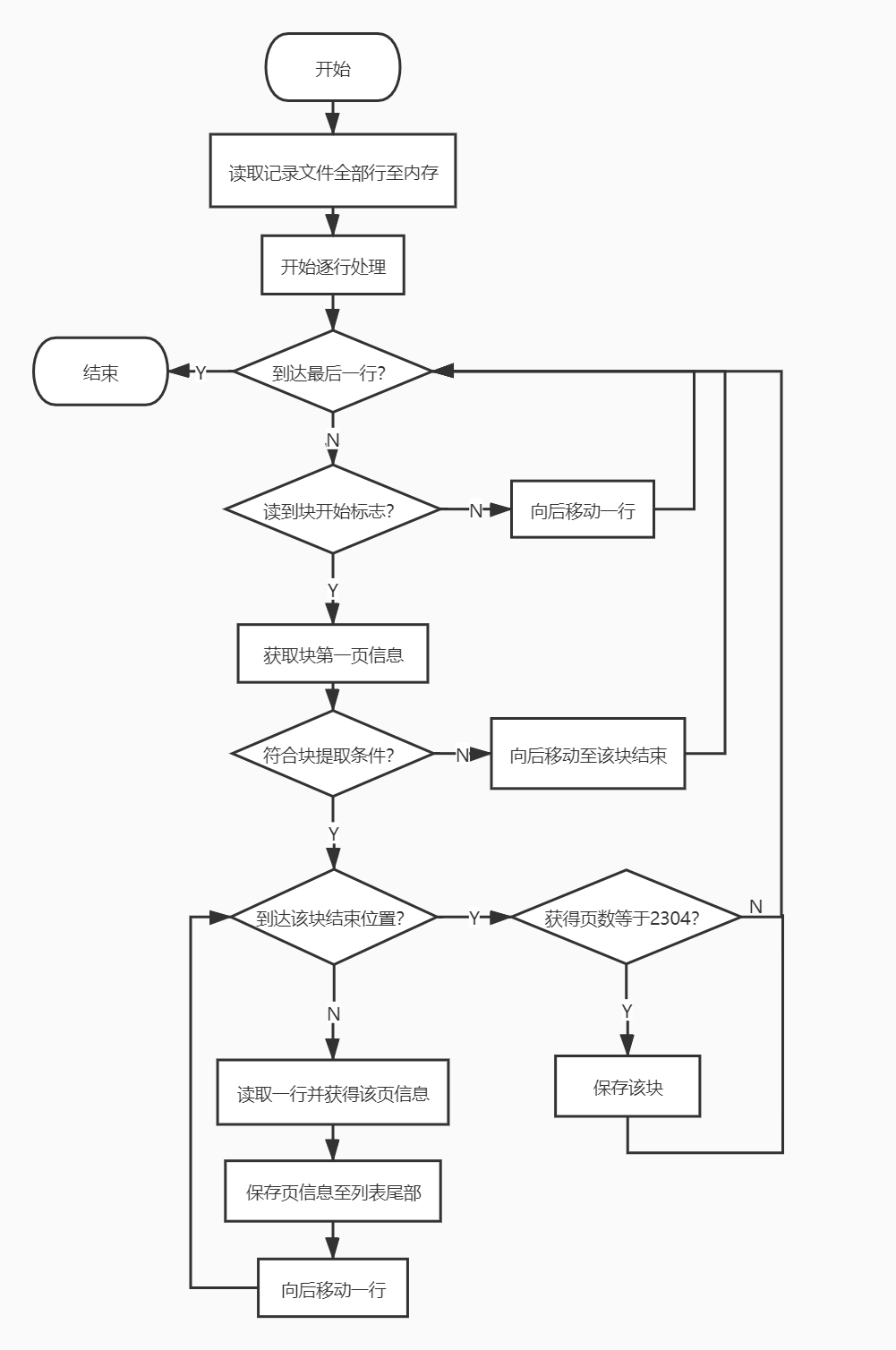
## 文本解析模块

对闪存进行测试可获得一系列错误记录文件，它们以文本的形式保存着闪存块的测试条件，各页错误信息，块属性等大量信息。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次数矩阵，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。

一个记录文件保存了闪存中一个chip中若干个块的信息，采用逐行处理的方式来提取信息。在对闪存块进行测试采集信息时，P/E测试点的间隔为100，实际上当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测试点的间隔值改为1000或者其他较大的值，来使得不同P/E情况下，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，因此对文本进行逐行处理，剔除不满足要求的块，具体流程图如图4-x，因为才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值等信息是否满足条件。

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。得到的块信息或被导入数据库，或以二进制格式保存至本地（npy格式文件）；这些块信息包括完整块错误矩阵，仅记录各页错误总数的块错误矩阵及其测试条件；



## 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。

## 数据统计与分析模块

### 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，每个位置的错误次数或多或少，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；转换方式如下：

针对给定P/E值，将该P/E值对应的所有块错误次数分布矩阵累加，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：

下图展示了P/E值分别为1,8000和16500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。同样后期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，与原始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。

### 块错误总数统计与分析

**a.** **块错误总数整体分析**

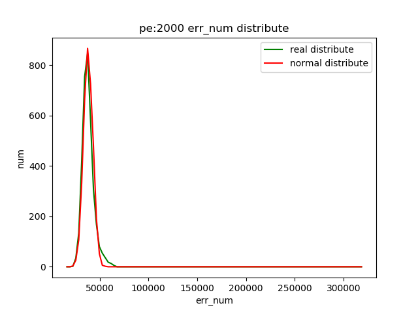
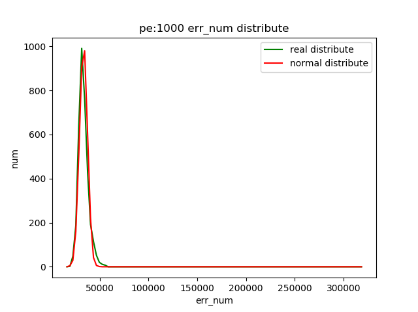
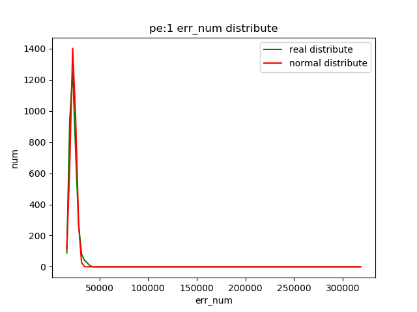
通过解析闪存错误记录文本文件，在P/E值集合：{1, 1000, 2000, … 16000, 17000} 的所有测试点收集块错误分布数据，共获得块错误分布数据62182个，每个块错误数据尺寸为2304×16，每个P/E值对应约3500个块数据。统计不同P/E值下，块错误总数的最小值，平均值以及标准差，如图4-x，可以考虑在无法获得全部P/E下块错误数据的情况下，可以考虑使用插值法获得拟合函数，来代替未被测试到的P/E点。

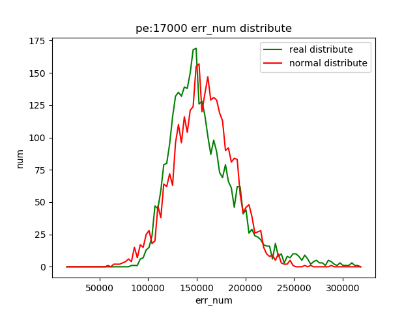
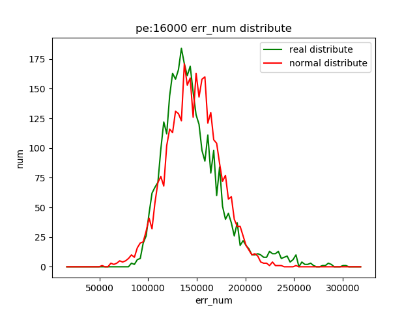
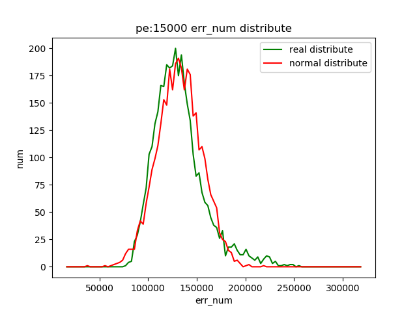
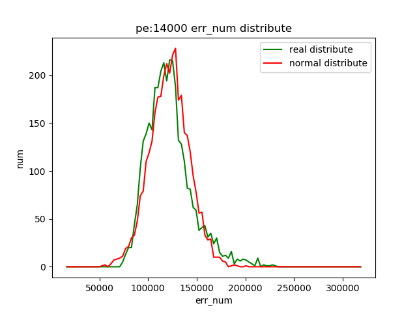
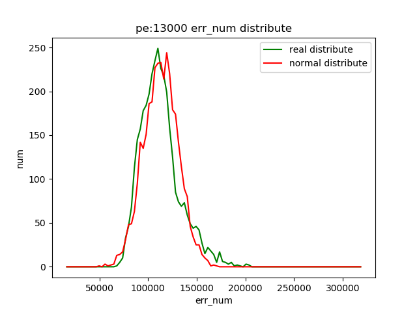
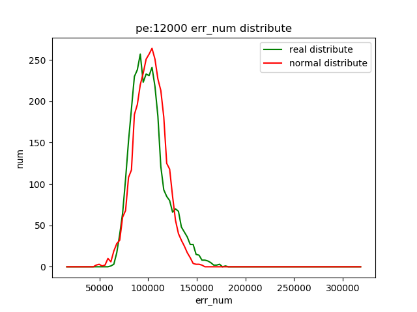
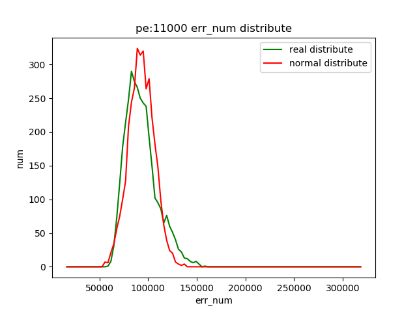
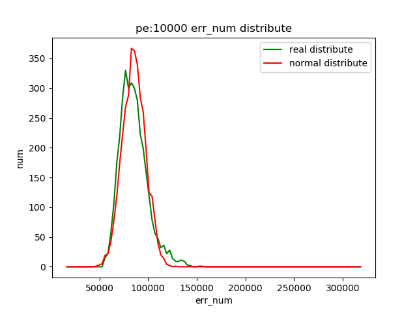
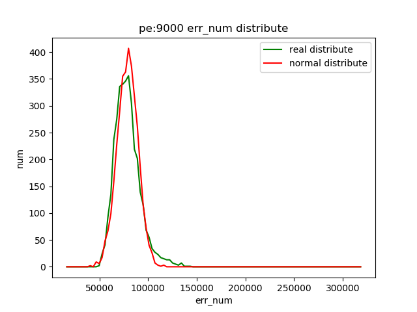
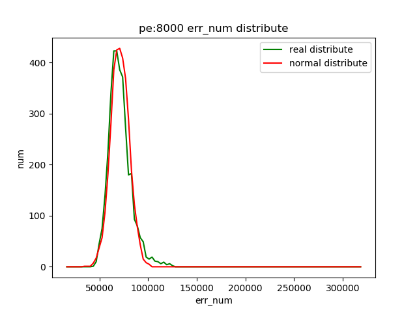
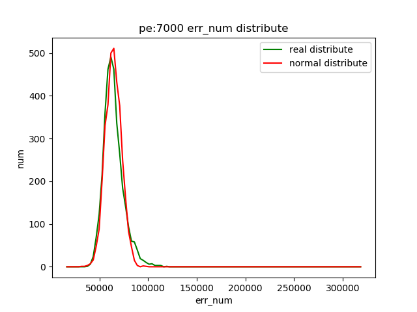
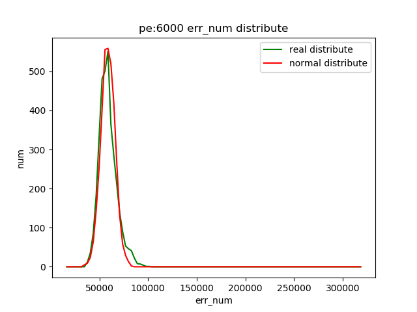
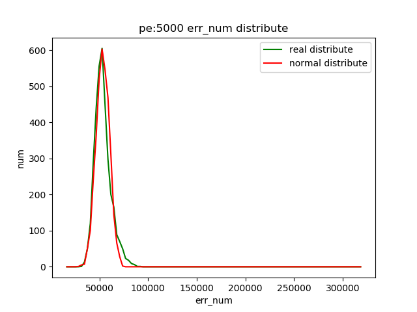
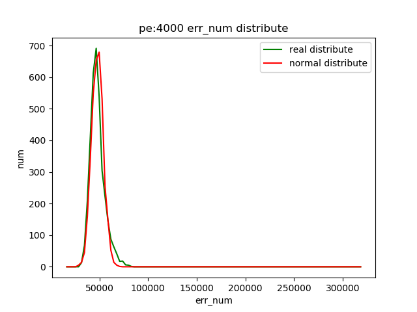
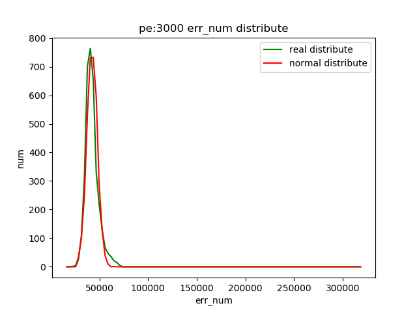


图4-x块错误总数-P/E

**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下块错误总数均值，std为该P/E下块错误总数标准差。频数分布直方图如下，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。





## 块相对错误分布生成模块

### 数据集定义

由于块相对错误分布生成模块的目的为生成一个块内各页的相对错误分布，因此该模块所用到的数据集为块的页错误数及该块对应的P/E次数。图4-1展示了数据的格式，每条数据尺寸为(2304, )，即包含了每页的错误总数，共2304页。

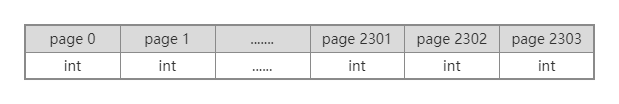


图4-1数据格式

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为(2304, )的矩阵，记录了每页的错误次数，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。

矩阵中的每个浮点值表示了该页的错误次数相对于最大的页错误次数的比率，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，需对数据集进行归一化，处理公式为：

最终矩阵内元素的取值范围为[-1, 1]。

### 模型定义

根据前文，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。

**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次数（P/E次数需要经过归一化和正则化）。输出一尺寸为(2304, )的浮点型矩阵。其内部的具体结构如图4-2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]。

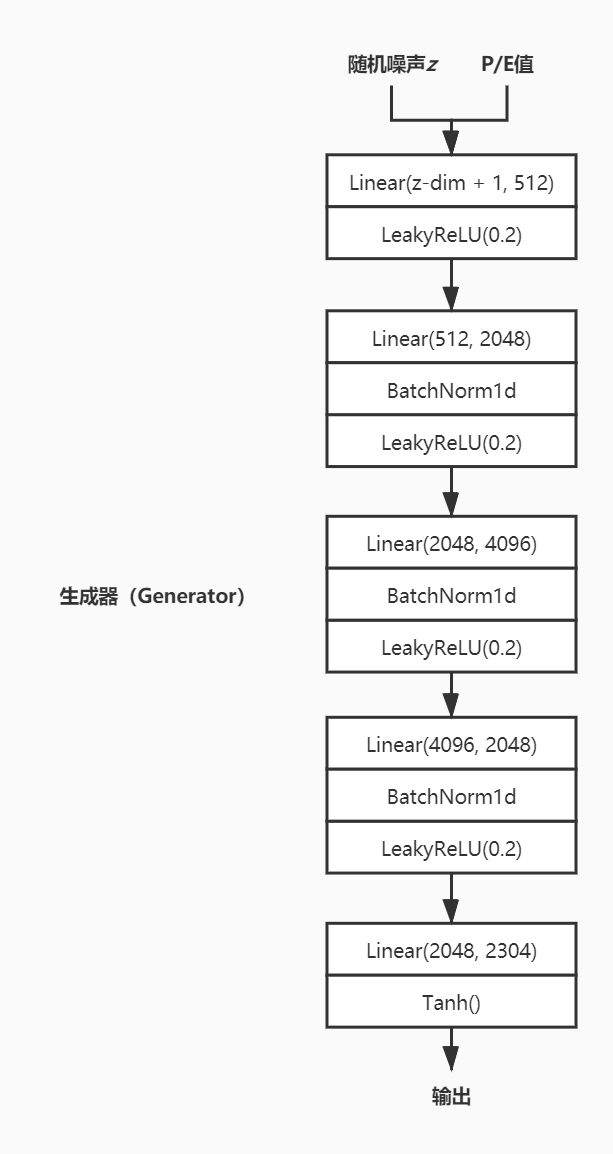
****

图4-1生成器

**b. 判别器定义**

判别器的具体结构如图4-1所示，其输入端接受块数据与P/E次数，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。

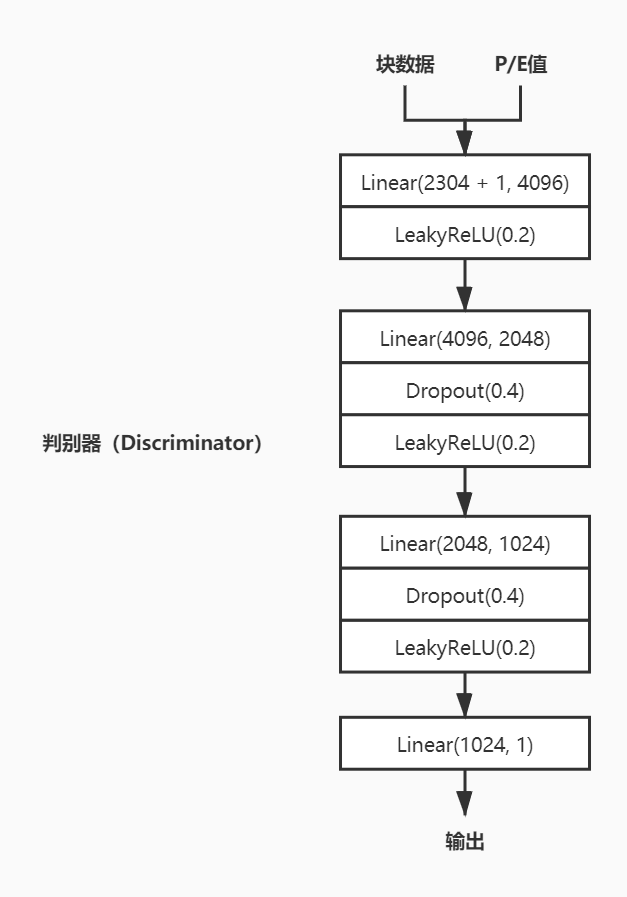


图4-1判别器

c. **训练过程**

模型训练流程如图4-x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真实数据训练判别器，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）。

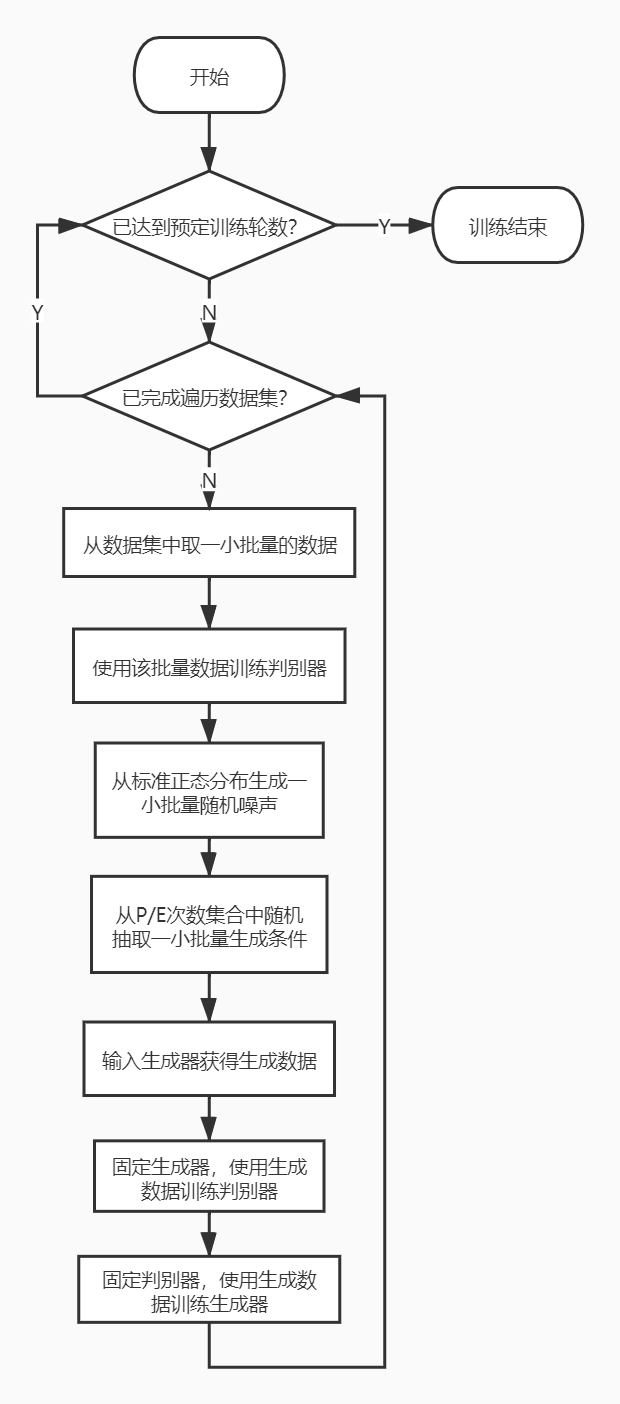


图4-1训练流程

## 块错误总数生成模块

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次数条件下的真实分布。

**a. 统计与记录**

对于拥有的全部块错误数据，按P/E次数进行分类，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次数一同保存下来。

**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, … 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，以统计所得的有限个点获得拟合函数，即以点集{, , … }进行插值，获得P/E🡪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来拟合块错误总数的真实分布。

使用Python中的标准正态分布来生成一组值*y*，则即为正态分布的对应值。

## 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，第i页的相对错误比值即为；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：

(4-1)

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(4-1)可获得每页的错误总数，再生成和为的16个均匀分布的随机数作为页内每个区域的错误次数，具体操作如下：

**a.** 在区间生成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；

**c.** 将0和添加入分割点集，得到，其中=0，=；

**d.** 16个和为的均匀分布随机整数如下：

至此获得一个完整的块错误次数分布数据。

## 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。

# 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。

## 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

|  |  |
| --- | --- |
| 名称 | 情况 |
| CPU |  |
| Memory | 16GB |
| GPU | GTX 1070 |
| OS | Linux Srever |

## 功能测试

### 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，每个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。文件解析过程主要代码如下，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, pe\_interval)*

*if action == import\_local:*

*global block\_err\_data\_list, page\_err\_data\_list, pe\_data\_list*

*np.save(local\_data\_path + "block\_err\_data\_%s.npy" % file,*

*np.array(block\_err\_data\_list, dtype=np.int32))*

*np.save(local\_data\_path + "page\_err\_data\_%s.npy" % file,*

*np.array(page\_err\_data\_list, dtype=np.int32))*

*np.save(local\_data\_path + "pe\_data\_%s.npy" % file,*

*np.array(pe\_data\_list, dtype=np.int32))*

*block\_err\_data\_list.clear()*

*page\_err\_data\_list.clear()*

*pe\_data\_list.clear()*

数据合并主要代码：

*result = np.load(data\_root\_path + file\_prefix + "\_" + file\_list[0] + ".npy").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 + ".npy").astype(np.float32)),*

*axis=0)*

测试结果如图：

### 块相对错误分布生成模块测试

**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)*

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。

部署脚本到服务器运行，结果如下：

**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 18000 \*

*--gen\_interval\_pe 1000 \*

*--generator\_data\_num 200"*

*for epoch in range(20, 120, 20):*

*os.system(command % (epoch, epoch))*

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18000，间隔为1000的生成条件集合{0, 1000, 2000, … 16000, 17000}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，结果如下：

### 块错误总数生成模块测试

对于生成条件集合{0, 1000, 2000, … 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，结果如下：

## 本章小结

本章对于实现完成的整个系统进行测试，对利用条件生成对抗网络的生成器生成一定数量的假分布数据，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各页错误相对分布的学习比较成功；以正态分布所得到块错误总数在大部分P/E值下与真实数据分布拟合的很好。

# 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，并给予了很多帮助，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，开展研究

参考文献

1. Palkar S, Lan C, Han S, et al. E2: a framework for NFV applications[C]. Symposium on Operating Systems Principles. ACM, 2015:121-136.
2. Bremler-Barr A, Harchol Y, Hay D. OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions[C]. Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:511-524.
3. Sekar V, Egi N, Ratnasamy S, et al. Design and implementation of a consolidated middlebox architecture[C]. Usenix Conference on Networked Systems Design and Implementation. 2011:24-24.
4. Anderson J W, Braud R, Kapoor R, et al. xOMB: extensible open middleboxes with commodity servers[C]. Eighth ACM/IEEE Symposium on Architectures for NETWORKING and Communications Systems. ACM, 2012:49-60.
5. A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella. Stratos: Virtual Middleboxes as First-class Entities[R]. Technical report, UW-Madison 2012.
6. Zhang W, Liu G, Zhang W, et al. OpenNetVM: A Platform for High Performance Network Service Chains[C]. The Workshop on Hot Topics in Middleboxes and Network Function Virtualization. ACM, 2016:26-31.
7. Gember-Jacobson A, Viswanathan R, Prakash C, et al. OpenNF: enabling innovation in network function control[C]. ACM Conference on SIGCOMM. ACM, 2015:163-174.
8. Rajagopalan S, Dan W, Jamjoom H, et al. Split/merge: system support for elastic execution in virtual middleboxes[C]. Usenix Conference on Networked Systems Design and Implementation. 2013:227-240.
9. J. Khalid, A. Gember-Jacobson, R. Michael, A. Abhashkumar, and A. Akella. Paving the Way for NFV: Simplifying Middlebox Modifications Using StateAlyzr[C]. In Proc. of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI’16), 2016.
10. Ballani H, Costa P, Gkantsidis C, et al. Enabling End-Host Network Functions[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4):493-507.
11. Paxson V. Bro: a system for detecting network intruders in real-time[C]. Conference on Usenix Security Symposium. USENIX Association, 1998:3-3.
12. Yoan N. Iptables[M]. Miss Press, 2013.
13. Sherry J, Gao P X, Basu S, et al. Rollback-Recovery for Middleboxes[C]. ACM Conference on Special Interest Group on Data Communication. ACM, 2015:227-240.
14. Chrome G. HTTP persistent connection[J]. 2015.
15. Surhone L M, Tennoe M T, Henssonow S F, et al. Ffmpeg[M]. 2010.
16. Housley R, Hoffman P. Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP[J]. Ietf Rfc Sri Network Information, 1999, 11(3):82--89.
17. Maheshwari A, Sharma A, Ramamritham K, et al. TranSquid :Transcoding and Caching Proxy for Heterogenous E-Commerce Environments[C]. International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/e-Business Systems. IEEE Computer Society, 2002:50.
18. J. Khalid, A. Gember-Jacobson, R. Michael, A. Abhashkumar, and A. Akella. Paving the Way for NFV: Simplifying Middlebox Modifications Using StateAlyzr[C]. In Proc. of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI’16), 2016.
19. NFV Paper [EB/OL].https://portal.etsi.org/NFV/NFV White Paper2.pdf,2017-05-05.
20. Hwang J, Ramakrishnan K K, Wood T. NetVM: high performance and flexible networking using virtualization on commodity platforms[C]. Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2014:445-458.
21. S. Han, K. Jang, A. Panda, S. Palkar, D. Han, and S. Ratnasamy. SoftNIC: A Software NIC to Augment Hardware[R]. Technical report, EECS Department, University of California, Berkeley, 2015..
22. Martins J, Ahmed M, Raiciu C, et al. ClickOS and the art of network function virtualization[C]. Usenix Conference on Networked Systems Design and Implementation. USENIX Association, 2014:459-473.
23. Panda A, Han S, Jang K, et al. NetBricks: taking the V out of NFV[C]. Usenix Conference on Operating Systems Design and Implementation. USENIX Association, 2016:203-216.
24. A. Gember, R. Grandl, A. Anand, T. Benson, and A. Akella. Stratos: Virtual Middleboxes as First-class Entities[R]. Technical report, UW-Madison 2012.
25. Palkar S, Lan C, Han S, et al. E2: a framework for NFV applications[C]. Symposium on Operating Systems Principles. ACM, 2015:121-136.
26. Rajagopalan S, Dan W, Jamjoom H. Pico replication:a high availability framework for middleboxes[C]. Symposium on Cloud Computing. 2013:1-15.
27. Sherry J, Gao P X, Basu S, et al. Rollback-Recovery for Middleboxes[C]. ACM Conference on Special Interest Group on Data Communication. ACM, 2015:227-240.
28. Qazi Z A, Tu C C, Chiang L, et al. SIMPLE-fying middlebox policy enforcement using SDN[J]. Computer Communication Review, 2013, 43(4):27-38.
29. Zhang W, Hwang J, Rajagopalan S, et al. Flurries: Countless Fine-Grained NFs for Flexible Per-Flow Customization[C]. International on Conference on Emerging NETWORKING Experiments and Technologies. ACM, 2016:3-17.
30. Bremler-Barr A, Harchol Y, Hay D. OpenBox: A Software-Defined Framework for Developing, Deploying, and Managing Network Functions[C]. Conference on ACM SIGCOMM 2016 Conference. ACM, 2016:511-524

此页以后的内容为装订材料！先仔细阅读！

撰写论文时此页以后的内容全部删掉！

装订毕业论文时应将任务书原件装订到最后



**本科生毕业设计任务书**

|  |  |
| --- | --- |
| 题 目 | 基于生成对抗网络的闪存仿真器设计与实现 |
|  |  |

（任务起止日期：2020年1月1日～2020年6月10日）

|  |  |
| --- | --- |
| 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 |

教研室（系、所）负责人 2020年1月6日 审查

院（系）负责人 2020年1月9日 批准

|  |
| --- |
| 课题内容 |
| 实现一个简单的分布式网络系统，实现文件的分布式网络化存储。，实现文件的分布式网络化存储。，实现文件的分布式网络化存储。，实现文件的分布式网络化存储。，实现文件的分布式网络化存储。，实现文件的分布式网络化存储。，实现文件的分布式网络化存储。，实现文件的分布式网络化存储。 |
| 课题任务要求 |
| 研究学习文件系统的相关原理及实现研究分布式系统概念及应用研究网络文件系统的原理、架构及实现。研究学习文件系统的相关原理及实现研究分布式系统概念及应用研究网络文件系统的原理、架构及实现。研究学习文件系统的相关原理及实现研究分布式系统概念及应用研究网络文件系统的原理、架构及实现。 |
| 主要参考文献 |
| 1. 董晓明，谢长生．基于对象的进化存储系统研究．计算机科学，2005，32(11): 223~226 2. 庞丽萍编．操作系统原理（第二版）．武汉：华中理工大学出版社，1994.9．225~270 3. （美）Nils J. Nilsson著；郑扣根等译．人工智能（Artificial Intelligence）．北京：机械工业出版社，2000.9．177~194 4. （美）Tom Mitchell著；曾华军等译．机器学习（Machine Learning）．北京：机械工业出版社，2003.1．38~56 5. 谢长生，董晓明，万继光，谭志虎，刘瑞芳．磁盘阵列控制器的设计与原型实现．小型微型计算机系统，2006, 27(1): 173~176 6. （美）Nils J. Nilsson著；郑扣根等译．人工智能（Artificial Intelligence）．北京：机械工业出版社，2000.9．177~194 |
| 同组设计者 |
| 无 |

**成 绩 评 定**

**指导教师评定意见**

装订论文时应将成绩评定页放最后，导师评语放前面！

一、对毕业设计（论文）的学术评语（应具体、确切、实事求是）

|  |
| --- |
|  |

二、对毕业设计评分

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 调研论证  (10分) | 外文翻译  (5分) | 设计(论文)撰写质量  (10分) | 学习态度  (10分) | 基本理论和基本技能  (50分) | 创 新  (15分) | 合 计  (100分) |
| 得分 | **8** | **4** | **8** | **8** | **40** | **12** | **80** |

指导教师签字：**吴非** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日

**答辩小组评定意见**

答辩小组意见和综合评定页放最后，双面！

一、评语（根据学生答辩情况及其设计（论文）质量综合评价）

|  |
| --- |
|  |

二、评分

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 答 辩 情 况 | | 论 文 质 量 | | 合 计  (100分) |
| 答辩情况  (15分) | 回答问题情况  (25分) | 规范要求与文字表达  (20分) | 学术水平  (40分) |
| 得分 |  |  |  |  |  |

答辩小组长签字：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日

**毕业答辩及成绩评定说明**

1. 毕业答辩
2. 答辩前，答辩小组应详细审阅每个学生的毕业设计（论文），为答辩做好准备。
3. 严肃认真组织答辩，开好答辩会。
4. 指导教师应参加所指导学生的答辩会，但评定其成绩时宜回避。
5. 答辩中要做好记录以供成绩评定时参考。
6. 成绩评定
7. 答辩前每个学生都要将自己的毕业设计（论文）在指定时间内交给指导教师，由指导教师审阅，写出评语并预评分。
8. 答辩工作结束后，答辩小组应举行专门会议进行讨论，在参考指导教师预评结果的基础上，结合学生毕业设计（论文）质量和学生答辩情况，综合评定每个学生的成绩。
9. 院（系）对专业答辩小组提出的优秀和不及格的毕业设计（论文），要组织院（系）级答辩，最终确定成绩，并向学生公布。
10. 各专业学生的最后成绩应符合正态分布规律。
11. 请用蓝、黑钢笔手写或五号宋体字编辑，签名须手写，A4纸双面打印。

**毕业设计（论文）成绩评定**

|  |
| --- |
| 班号：**计算机1604** 学生姓名：**孟嵩淼**  综合成绩：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_分（折合等级\_\_\_\_\_\_\_\_\_\_\_\_\_\_）  答辩小组长（签名）：\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 年 月 日 |