中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

2022—2023 学年第 2 学期
《人工智能基础》课程报告

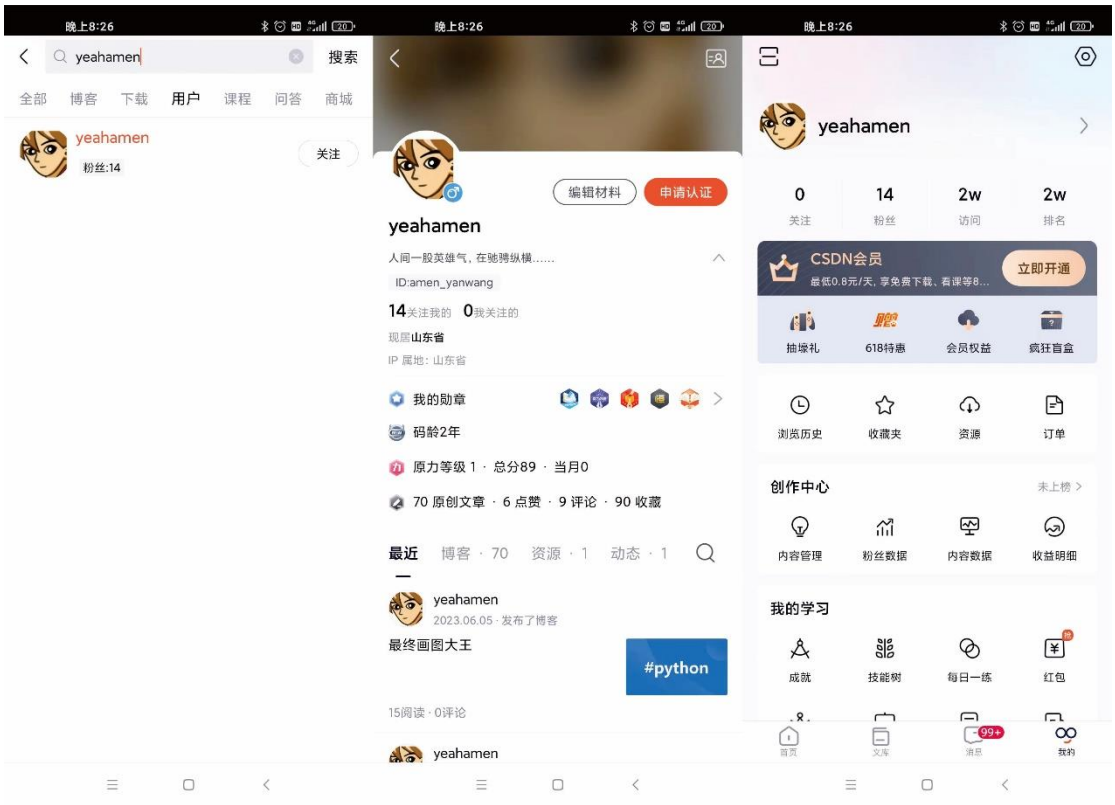
选题名称	CNN 识别岩心及半监督 K-means 确定震中位置			
小组成员	学号	姓名	任务分工	备注
	2001030118	王岩	1、2、4、5、报告整理	组长
	2001030104	郭文静	1、3、4、7	成员
	2001030107	杜双全	1、3、7	成员
报告评分				

2023 年 6 月

目　录

0 说明

本次《人工智能基础》期末报告所有修改、编写的程序是经过查阅大量网络文章并参考网络实验平台的部分内容介绍，独立应用于实践内容的，其真实有效性都已经上传至组长王岩的 CSDN 个人账户：yeahamen。



CNN 识别岩心及半监督 K-means 确定震中位置

1 研究背景

大数据时代利用机器（人工智能）代替人力完成工作无疑有很多优势，比如更节约资源、工作质量也可以得到更稳定的保证。

（1）CNN 鉴别岩心

本小组成员曾负责地质学类大学生创新创业训练项目：《济阳坳陷沾化凹陷沙四上亚段膏岩的成因机制》，2021.9—2023.5，已结题。在这个训练过程中，做的最基础的工作是识别岩心，一共完成了 4 口取芯井约 1200 余块岩心的观察与鉴定。显然，这个工作是枯燥、让人疲惫的。工作过程中就一直在想，有没有可以自动识别并标记岩性的工具，来代替我做这个工作？由于知识与眼界的限制，这个问题一直没有找到答案。

本学期伊始，《机器学习》这门课就提上了日程。学到卷积神经网络部分时，老师介绍通过监督学习可以将训练集的图片进行训练，得出一个模型，然后将与训练集相似的测试集图片（无标签）输入该模型，就可以给其加上标签。很快，甚至是转眼之间，我就想到尝试利用这个工具去识别岩心。并且《人工智能基础》课程实验平台也提供了几个很好的例子来帮助我们理解，所以本次人工智能报告第一部分工作就是利用卷积神经网络（CNN）来鉴别岩心，并取得了不错的效果。

（2）半监督 Kmeans 确定震中位置

在《人工智能基础》课程的期末任务发布之前，本专业刚刚完成了《地震学》考试。主要学习了地震的发生、传播与观测；震源基本参数的确定；地震勘探与地震波传播理论等内容。其中震源基本参数的确定包括震中位置的确定。什么是震中位置？就是震源与地心的连线与地表的交点。当然还有很多专业名词，我们放到后面去解释。对于远震（震中距 Distance>1000Km），不同国家的地震台站往往会得出不同的震中位置，如果要想得到比较好的震源参数数据，则需要对这些位置取平均值，但面对大量的数据与地震波传播各向异性的影响，这种方法显然是不充分的。利用半监督 K 均值算法，我们可以将高质量台站数据与低质量台站数据充分地分开利用，而不是总体取均值。

因此，本报告的第 2 项部分内容就是半监督 K-means 算法确定震中位置，该方法在一定意义上可以作为实际地震中位置。

2 研究内容与目标

2.1 卷积神经网络（CNN）识别岩心

2.1.1 研究内容

研究内容或对象是取自济阳坳陷沾化凹陷沙四上亚段罗 14 井（2939.8m—3029.51m）、罗 2 井（3082m—3151.6m）、罗 4 井（3145m—3226.4m）、新罗 39 井（2989m—2995.5m）的岩心，经过进一步筛选，整理出比较完整、图像比较清晰的 320 块岩心照片作为训练集，40 块岩心照片作为测试集。这些岩心一共分为四类：膏岩、灰岩、膏质灰岩、灰质膏岩。

2.1.1 研究目标

首先构建卷积神经网络模型，将训练集用于模型训练。然后将测试集输入训练好的模型，让模型给测试集的每张岩心照片都加上标签，然后与真实的标签进行对比，以此来衡量岩心识别效果。预计目标是能把 40 张训练集照片全部识别准确。

2.2 半监督 K-means 算法确定震中位置

2.2.1 研究内容

研究内容或对象是位于喜马拉雅周边地震带的成都—重庆一带区域。我们在地图上标出有限个离散点来模拟高质量或低质量台站处理数据得到的伪震中位置，然后利用半监督 K-means 算法来处理这些离散点，以期得到真正的震中位置。

2.2.1 研究目标

较完整地利用高质量台站数据（有标签）与低质量台站数据（无标签），所得到地震中位置是经过 K-means 算法对上述数据进行两轮不同地训练得来的，预计识别地震中位置在一定意义上可以作为真实的震中位置。

3 方法介绍

3.1 卷积神经网络（CNN）

3.1.1 概述

现在的人脸识别，医学影像，自动驾驶等的开发都有卷积神经网络的身影。他比普通的神经网络效率更高。比较典型的网络有牛津大学的 Visual Geometry Group 在 2015 年发布的共 16 层的卷积神经网络（VGG16 模型），约有 1.38 亿个网络参数，可以为初学者提供卷积神经网络的学习与初步体验。

该模型是针对 ImageNet 挑战赛设计的，该挑战赛的数据集为 ILSVRC-2012 图像分类数据集，该数据集的训练集有 1281167 张彩色图片，分为了 1000 个类别，验证机有 5000 张图片，每一类有 50 个样本。每一张图片的形状大小为 224*224*3。VGG16 总共包含 16 个子层，第 1 层卷积层由 2 个 conv3-64 组成，第 2 层卷积层由 2 个 conv3-128 组成，第 3 层卷积层由 3 个 conv3-256 组成，第 4 层卷积层由 3 个 conv3-512 组成，第 5 层卷积层由 3 个 conv3-512 组成，然后是 2 个 FC4096，1 个 FC1000。总共 16 层，这也就是 VGG16 名字的由来。随后，Visual Geometry Group 又发布了 VGG19 模型，是在 VGG16 的基础上在其第 3、4、5 卷积层分别添加一个卷积核，层与层之间依然用最大池化层（maxpool）连接，从而构成 19 层的卷积神经网络。

我们鉴于对神经网络初步认识的需求也编写了程序尝试让这个模型识别任意一张图片，实实在在地体会到了人工智能地神奇之处，由于篇幅的原因，不在本文展示，可见于 http://t.csdn.cn/OmMTO。

3.1.2 卷积神经网络的架构

一个卷积神经网络主要由以下 4 层组成：

输入层 / Input layer；卷积层 / CONV layer；池化层 / Pooling layer；全连接层 / FC layer

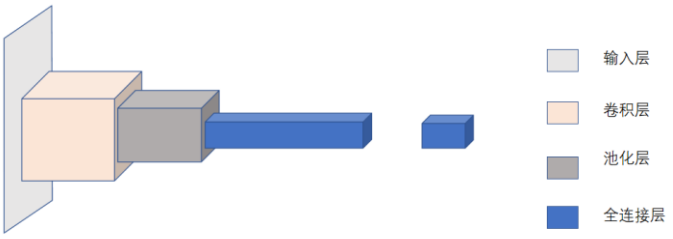


图 3-1 卷积神经网络关系图

3.1.3 卷积神经网络处理举例

我们基于自己的理解，对卷积神经网络处理图像（或者说矩阵）的过程进行举例：实战内容是以 256*256*3 输入层开始，选取 5*5*32 的卷积核进行卷积，在经过 4*4 最大池化层、2*2 最大池化层、然后是全连接层结束为组成部分的整个神经网络。由于数据量较大，在此处我们以 6*6 的单层像素矩阵输入，由 3*3 的卷积核卷积，在经过 2*2 的最大池化核池化，再进行全连接层处理输出结果，来说明我们对 CNN 处理流程的理解。

（1） **输入层**(Input layer)

输入层通常是输入卷积神经网络的原始数据或经过预处理的数据，可以是图像识别领域中原始三维的多彩图像，也可以是音频识别领域中经过傅利叶变换的二维波形数据，甚至是自然语言处理中一维表示的句子向量。实际的彩色图像是由 R、G、B 三个通道（可以理解为三层）的二维像素矩阵组成的，灰白图像是由一个通道（一层）的二维像素矩阵组成的，像素点的范围为（0-255）。

我们以输入 6*6 的单层像素矩阵处理举例（图 3-2）。

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

图 3-2 输入图像

（2）**卷积层**（CONV layer)

这一层是卷积神经网络最重要的一个层次，该层中卷积操作是通过卷积核对每个通道的矩阵从左到右从上至下进行互相关运算（先是从左到右，再是从上至下），就像一个小的窗口一样，从左上角一步步滑动到右下角，滑动的 stride (步长)可以根据数据量大小给定，互相关运算就是对应位置相乘再相加。

我们采用 3*3 的卷积核（步长 stride=1）来处理 6*6 的单层像素矩阵（如图 3-3）。（蓝色计算举例：1*253+0*25+1*130+0*159+1*78+0*233+1*214+0*245+1*0=657），自左上角只右下角共经历了 16 次卷积运算，得到了（6-3+1）*（6-3+1）即 4*4 的卷积层。

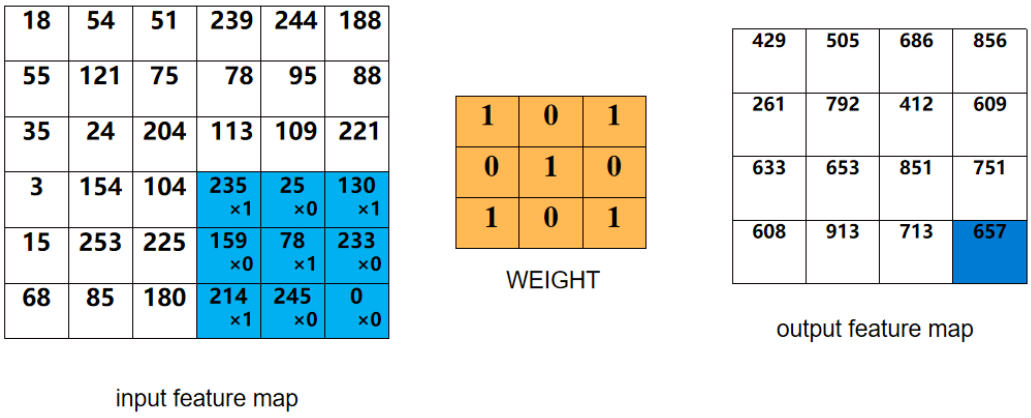


图 3-3 卷积核处理输入得到卷积层

（3）**最大池化层**（MAX Pooling layer）

池化层夹在连续的卷积层中间， 用于压缩数据和参数的量，减小过拟合。简而言之，如果输入是图像的话，那么池化层的最主要作用就是压缩图像。

继续对上面卷积层结果进行池化：采用 2 * 2 的最大池化核，即选取 2*2 区域内最大值，池化核移动方向顺序与池化核一样，其步长一般等于其边长（此处是 2），最大池化的结果显然是（4-2）*（4-2）的池化层矩阵。

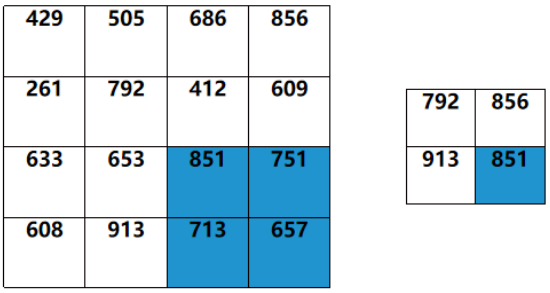


图 3-4 卷积层被最大池化得池化层

（4）Dropout **层处理过程**

Dropout 在训练过程中，本质是对于每个神经元以一定的概率进行丢弃。通常情况下，保留概率为 p，丢弃概率为 1-p，可以通过一些随机采样的方法来实现。在前向传播过程中，被丢弃的神经元的输出值会被置为 0；在反向传播过程中，被丢弃的神经元也不参与误差反向传播，从而减少神经网络的参数量和模型复杂度，进而**避免过拟合**。

一般的神经网络与应用了 Dropout 的网络进行对比（图 3-5），Dropout 通过随机选择并删除神经元，停止向前传递信号，可以达到防止过拟合的目的。

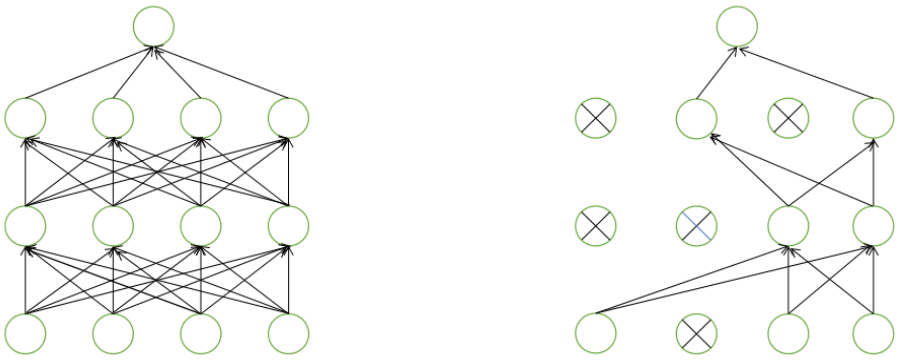


图 3-5　Dropout 层作用（舍弃部分神经元，防止过拟合）

（5）Flatten 层处理过程

Flatten 很简单，只是将输入的多维数据拉成一维的，直观上可理解为将数据“压平”不影响 batch 的大小。通过对池化层的拉平处理，2*2 的矩阵变为了 1*4 的向量（图 3-6）。

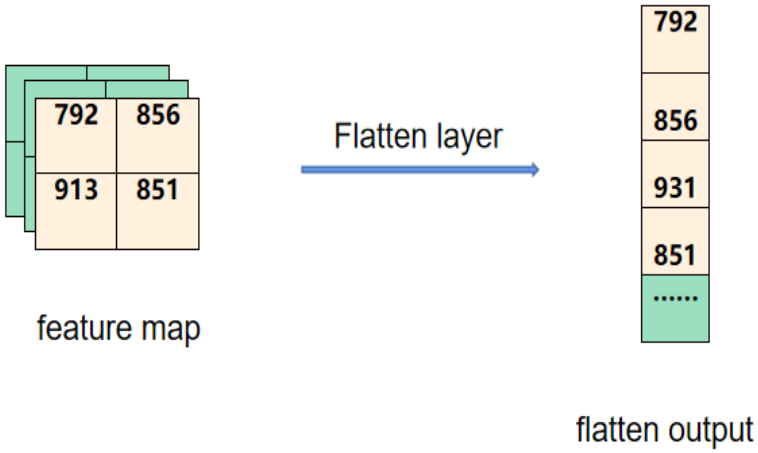


图 3-6　Flatten 层的输出展平示意图

（6） 批标准化层

批标准化(Batch Normalization)层可以抑制梯度消散。

深度神经网络的训练复杂的原因是在训练时每层的输入都随着前一层的参数的变化而变化。针对误差难继续传递问题，批标准化对每一层的批量输入数据 x 进行标准化，使之尽量避免落入饱和区，具体来讲就是使之均值为 0，方差为 1。对每一批输入数据 $B=\{x_1,x_2,\cdots,x_m\}$：

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{3-1}$$

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i = \gamma\hat{x_i} + \beta$$

其中，ε 为防止除 0 的很小的常数。前三步分别为计算均值、计算方差、标准化，最后一步是对归一化后的结果进行缩放和平移，其中的 γ 和 β 是要学习的参数。

（7） **全连接层**（Fully connected layer）

然而，为了生成最终的输出，我们需要应用全连接层来生成一个等于我们需要的类的数量的分类器（标签的个数就是神经元的个数）。对于前层是全连接的全连接层可以转化为卷积核为 1x1 的卷积，我们将如果将上面经过池化处理的结果直接进行全连接层操作会出现：铺平展开就可以得到全连接层（图 3-7），这与 Flatten 层很相似。

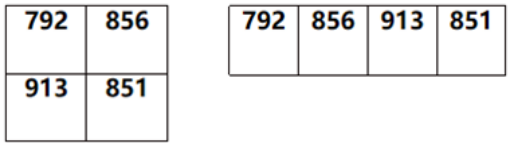


图 3-7　全连接层示意

（8）输出过程

Soft max 函数可以用下面的表达式表示：

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n}\exp(a_i)} \tag{3-2}$$

上式表示假设输出层共有 n 个神经元，计算第 k 个神经元的输出y_k。Soft max 函数的分子是输入信号a_k的指数函数，分母是所有输入信号的指数函数的和。

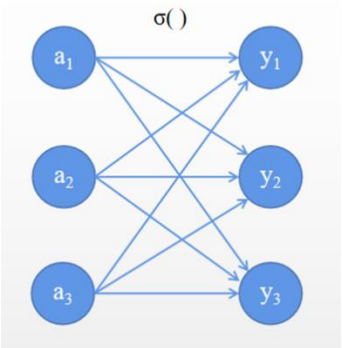


图 3-8 soft max 函数输出过程

从图 3-8 可以看出，soft max 函数的输出通过箭头与所有的输入信号相连。这是因为，输出层的各个神经元都受到所有输入信号的影响。

3.2 半监督 K-means 算法

3.2.1 K-means 聚类算法概述

K-means 算法是基于数据划分的无监督聚类算法，是数据挖掘领域的十大算法之一。其中 K 表示类别数，means 表示均值。它是一种通过均值对数据点进行聚类的算法。它不需要给训练数据打上标签,只需预设一个分类个数，然后数据会被 K-means 算法自动划分 K 个类别。K-means 算法通过预先设定的 K 值及每个类别的初始中心对相似（距离近）的数据点进行划分，并通过划分后的均值迭代优化获得最优的聚类结果。

3.2.2 算法目的

K-means 算法主要解决的问题就是把多个点群划分为多个簇，在图 3-9 的左边有一些点，我们用肉眼可以看到有四个点群，但是我们怎么通过计算机程序找出这四个点群来呢？于是就出现了我们所要讲的 K-means 算法。

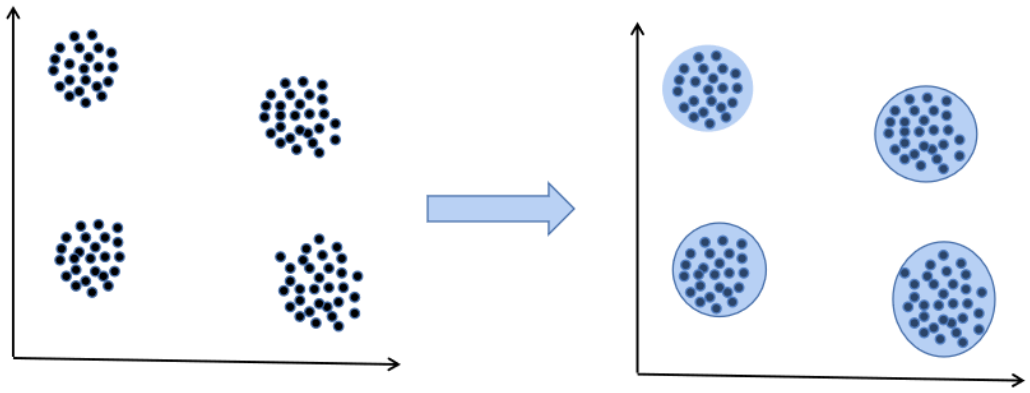


图 3-9 四个点群的具体分布图

3.2.3 核心原理

为具体说明 K-means 算法的核心原理，特示例 K-means 算法的处理流程（图 3-10）：A，B，C，D，E 是在图中的五个点，而灰色的点是我们的聚类中心，也就是我们用来找点群的点。因为有两个聚类中心，所以 K=2。K-means 聚类算法可总结为四个基本步骤：

（1）为待聚类的点给出初始聚类中心，可通过肉眼观察先确定 K 的个数。这里在图中取 K=2 个聚类中心的位置。

（2）计算每个点到聚类中心的距离，将每个点归到到离该点最近的聚类中心对应的点群中去，这一过程就是聚类。同一个点群，所有样本点到中心的距离之和越小，认为点群中的样本越相似，点群内差异就越小，在此距离常用欧式距离计算（式 3-3）。

$$d(X,Y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_n-y_n)^2} \quad (3\text{-}3)$$

对图 3-2 中的所有点求到这 K 个聚类中心的距离并进行比较，假如点样本点 A 离聚类质心 K1 最近而离 K2 较远，那么就将 A 聚类到属于 K1 对应的点群。（上图 3-9 中，我们可以看到 A，B 属于聚类质心 K1，C，D，E 属于聚类质心 K2）。

（3）计算每个聚类中所有点的坐标平均值，并将这个平均值作为新的聚类中心。图 3-2 的第三步即为移动聚类中心到属于它的点群的中心（即每判断完毕一个点，就要更新一次质心位置）。

（4）反复执行（2）、（3），直到聚类中心不再进行大范围移动或者聚类次数达到要求为止（如 A、B、C 都与 K1 构成一个点群；D、E 都与 K2 构成一个点群）。

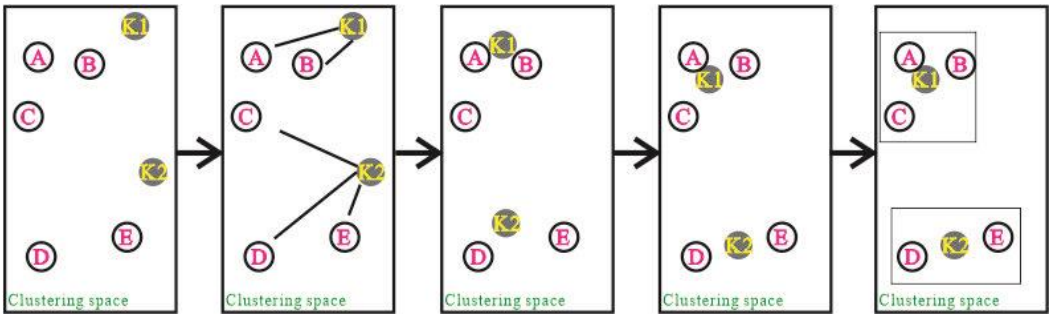


图 3-10　K-means 算法的流程示意图

3.2.4　半监督 K-means 聚类算法

由于 K-means 算法是无监督的，在训练过程中难免会产生数据点分配不当或质心个数受各个样本点的影响较大，导致产生分配极为不均的问题。

我们小组对于半监督 K-means 算法的理解是这样的：对于给定的数据集，我们把它先利用 python 脚本输出到合适的坐标系中展示出来，利用肉眼观察数据集大概分为几类，我们在各个类中都估计出 2-3 个点或多个点，来作为“有标签数据”。先将这些有标签数据进行训练，当然这也会出现类别不均衡的问题，但几率大大降低，并且由于我们人为选取的“有标签数据”样本量较小，多次重复对计算机的内存要求也不高，因此在一定意义上可以作为半监督算法的应用思想。

4 应用工具和环境搭建

本次报告利用的工具包括：IDLE（Python3.10 64-bit）、PPT2019、ArcMap10.4.1、CorelDRAW2019 等编程及绘图软件。总共利用到了 14 个模块引用（表 4-1）。

表 4-1 Python 编程引用模块

import os
import cv2
import csv
import torch
import pylab
import PySide2
import datetime
import numpy as np
import pandas as pd
from PIL import Image
import tensorflow as tf
import tensorflow.keras as ka
from torchvision import models
import matplotlib.pyplot as plt
from tensorflow.keras import Model
from torchvision import transforms
from torch.utils.data import Dataset
from torchvision.io import read_image
from torch.utils.data import DataLoader
import tensorflow.keras.applications.vgg19 as vgg19
import tensorflow.keras.preprocessing.image as imagepre
from tensorflow.keras.layers import Conv2D,BatchNormalization,Activation,MaxPooling2D,Dropout,Flatten,Dense

5 实现过程及效果分析

5.1 CNN 识别岩心（自定义数据集）

5.1.1 引言

卷积神经网络的神奇性在于其能够对训练过的样本集中的内容提供可靠的识别概率，本组成员王岩曾参加地质类创新创业项目，实践过程中最基础的“岩心识别”工作花费了大量时间，如果利用机器来完成这项繁重的工作，无疑是提高效率的一大利器。当然，标题中所提到的自定义数据集就是岩心照片，首先抱着试一试的态度，准备的训练集是由 30 张岩心照片组成的（图 5-1），把它分为三个类别：0、石膏岩（白色）、1、石灰岩（灰色）、2、灰质膏岩（灰白相间）（数字为标签）（图 5-2）；测试集是由五张照片组成的（图 5-3、5-4），同样包括三个类别。

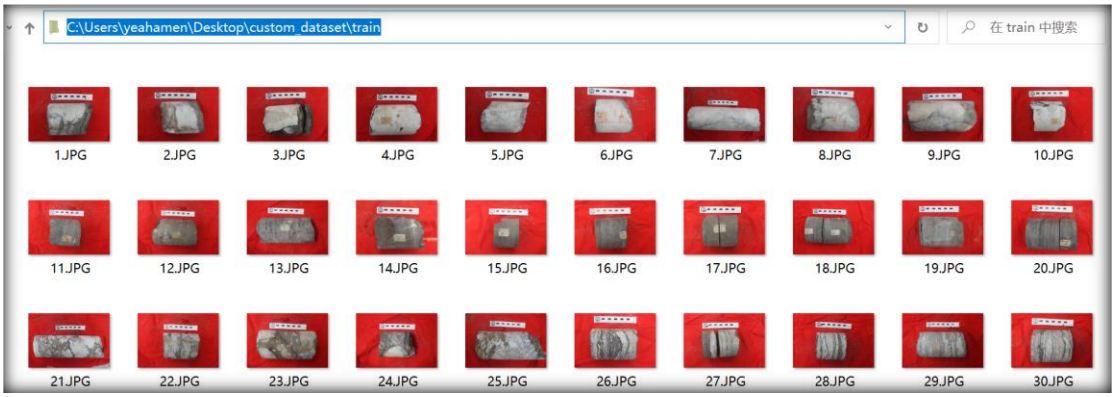


图 5-1 自定义数据集（custom_dataset）—训练集（train）展示

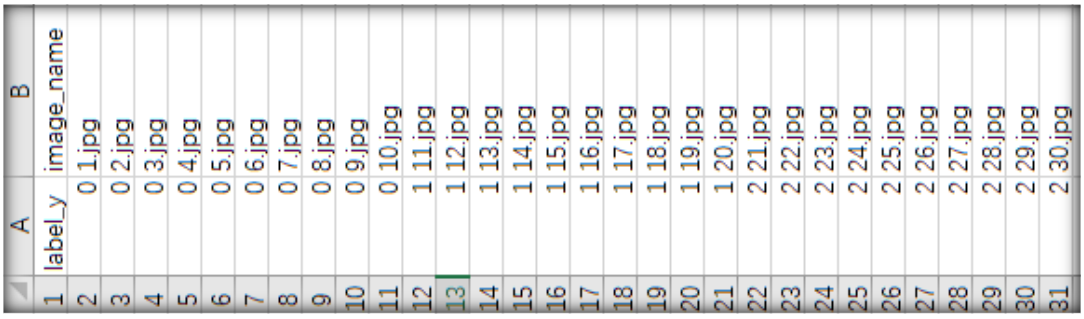


图 5-2 自定义数据集（custom_dataset）—训练集标签（train_label）展示

图 5-3 自定义数据集（custom_dataset）—测试集（test）展示

	A	B
1	test_label_y	test_image_name
2	2	1.jpg
3	0	2.jpg
4	0	3.jpg
5	1	4.jpg
6	1	5.jpg

图 5-4 自定义数据集（custom_dataset）—测试集标签（test_label）展示

由于每张照片形状大小都是 3456*5184*3，像素点过多导致电脑运行程序时出现内存不足的错误，因此只能把每张图像的像素点数目减少（**程序为附件 1**）运行后，原来的测试集与训练集图像均变为 256*256 大小的图片（5-5、5-6），标签不需要进行改动，因此与原来保持一致。

图 5-5 训练集图像修改尺寸后图像（train_revise）展示

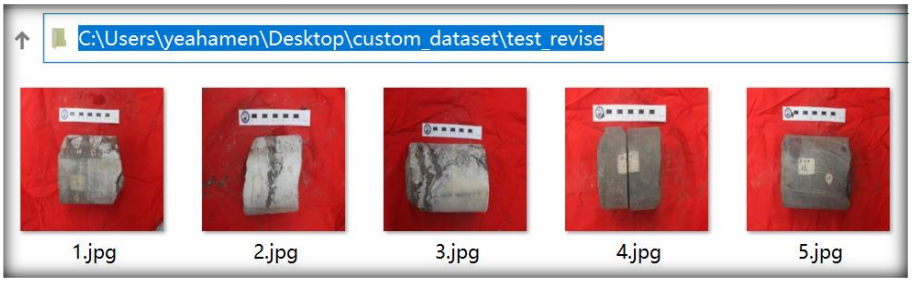


图 5-6 测试集图像修改尺寸后图像（test_revise）展示

准备好数据集之后就可以进行编写程序(因为后面对数据集进行了改进但程序改动不大，所以此处少量数据集不放程序，而放在后面)。运行完毕之后发现由于数据量太少(图 5-7)，即使训练 100 轮的，发现精度仍在 0.6 附近波动（图 5-8），但转念一想：即使精度达到 1 由于少量的样本也无法来证明模型的正确性。

图 5-7　30 个训练集 5 个测试集形状

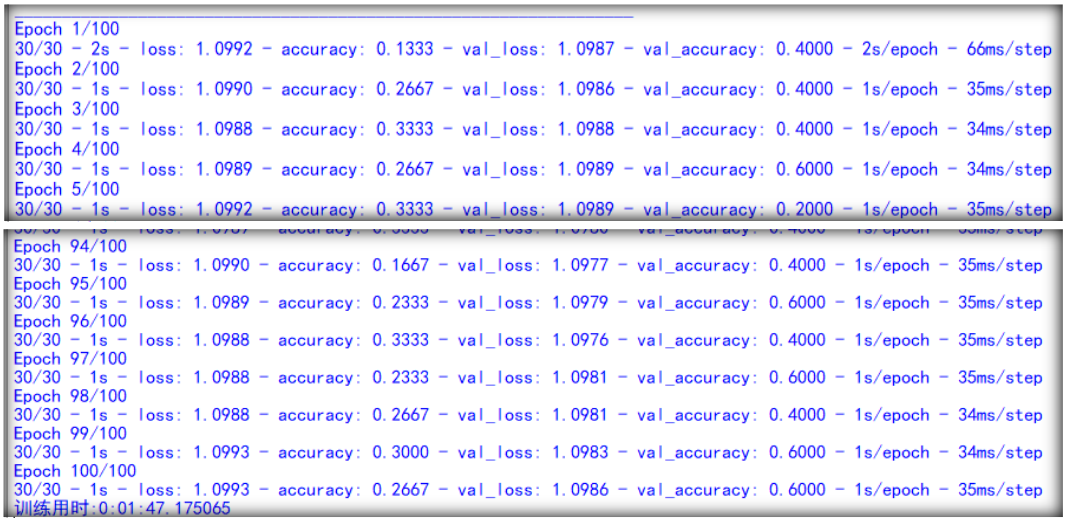


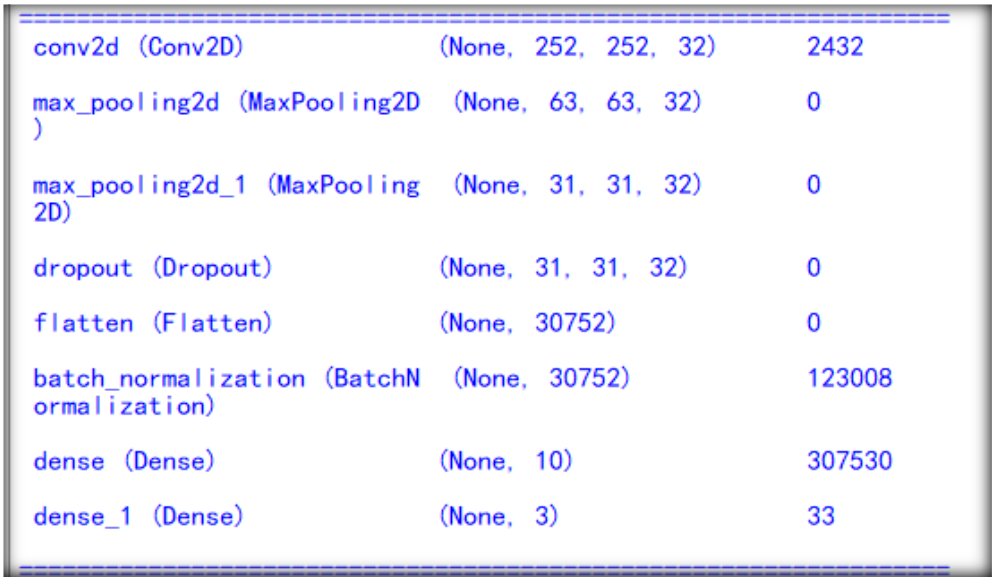
图 5-8　30 个训练集 5 个测试集训练 100 轮识别精度展示

本训练模型的参数为（图 5-9）：由 256*256*3 输入层开始，经历 5*5*32 卷积核生成 252*252*32 卷积层、然后是 4*4 最大池化核生成 63*63*32 池化层 1、2*2 最大池化核生成 31*31*3 池化层 2、再次是 Droupout 层、Flattern 层、批标准化层、10 神经元全连接层 1（relu）、直到 3 神经元全连接层 2（softmax）结束（最后的是对应三个标签区分识别结果）。另外，批样本容量为 1。

所以 30+5 个样本太少，不足以支持本次卷积神经网络模型实践的可靠性，因此进一步扩大了样本数量：训练集由 30 张增加到 320 张彩色照片并分为四类（0—膏岩、1—灰岩、2—灰质膏岩、3—膏质灰岩），测试集由 5 张增加到 40 张彩色照片也是四类。模型参数仅

把全连接层 1 神经元数由 10 增加到 50，全连接层 2 神经元数由 3 增加到 4（图 5-14）。另外，批样本容量为 5（**此处程序放在附件 2**）。

通过不断增加训练轮数，训练效果有显著的变化（图 5-10—5-13），损失值不断降低，测试精度不断升高，在经历 40 轮训练后，损失值下降到 0.1 个百分点一下，测试精度稳定在 0.9 以上，意味着本次实践的成功！



conv2d (Conv2D)	(None, 252, 252, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 31, 31, 32)	0
dropout (Dropout)	(None, 31, 31, 32)	0
flatten (Flatten)	(None, 30752)	0
batch_normalization (BatchNormalization)	(None, 30752)	123008
dense (Dense)	(None, 10)	307530
dense_1 (Dense)	(None, 3)	33

图 5-9　卷积神经网络训练模型参数（train_30-test_5）

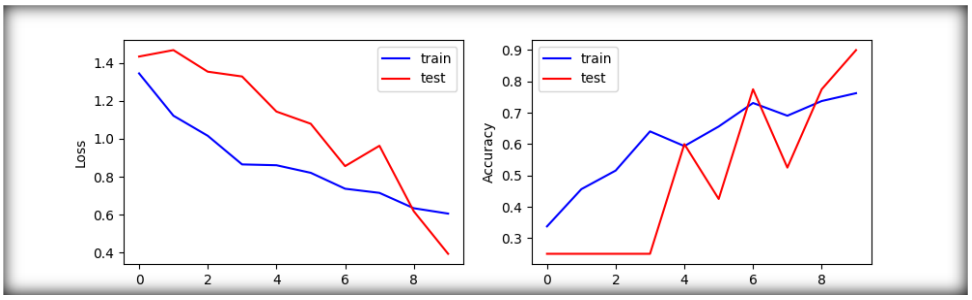


图 5-10　训练 10 轮损失值与精度曲线

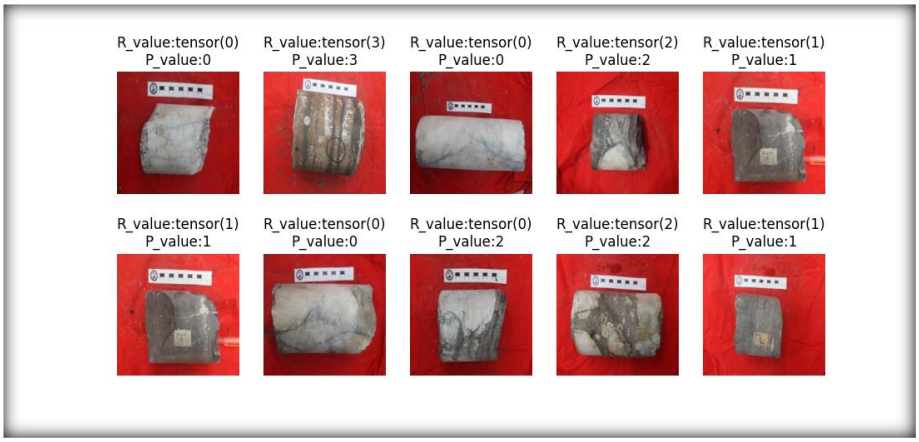


图 5-11　训练 10 轮预测结果

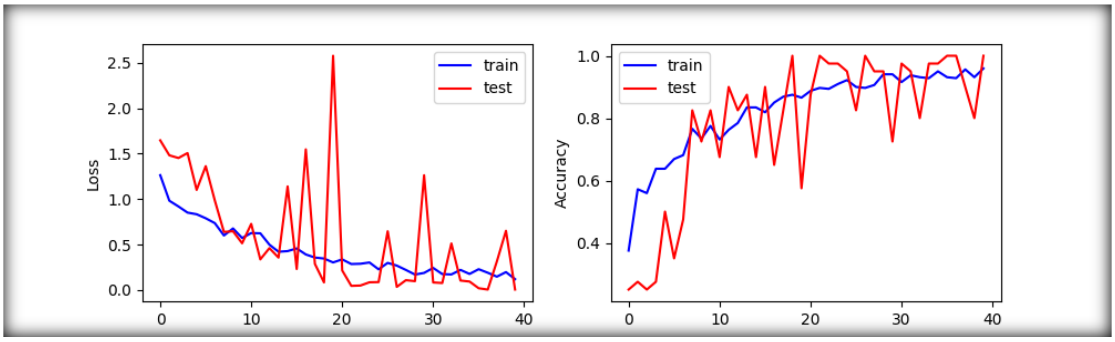


图 5-12 训练 40 轮损失值（左）与精度（右）曲线

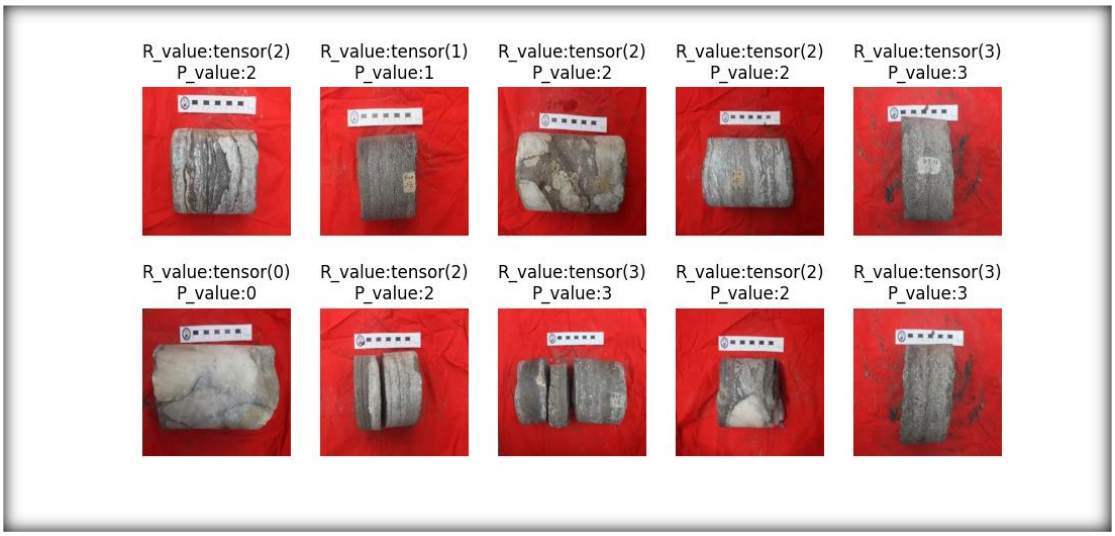


图 5-13 训练 40 轮预测结果

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 252, 252, 32)	2432
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
max_pooling2d_1 (MaxPooling2D)	(None, 31, 31, 32)	0
dropout (Dropout)	(None, 31, 31, 32)	0
flatten (Flatten)	(None, 30752)	0
batch_normalization (BatchNormalization)	(None, 30752)	123008
dense (Dense)	(None, 50)	1537650
dense_1 (Dense)	(None, 4)	204

图 5-14 卷积神经网络训练模型参数（train_320-test_40）

5.2 半监督 K-means 算法确定地震震中位置

5.2.1 震中（Epicenter）

地震发生时，震源与地心的连线在地表的投影称为震中。震中分为两类：

（1）宏观震中，在地震现场进行勘察，并勾画等震线（地表等烈度的连线），以此方法确定的烈度最大的位置为宏观震中，是具有一定范围大小的区域（图 5-15）；

图 5-15 台湾省高雄市 6.7 级地震宏观震中位置

（2）微观震中，震源在地表的投影，或根据地震台站信息计算得来的震中位置称为微观震中，理论上是一个点位。

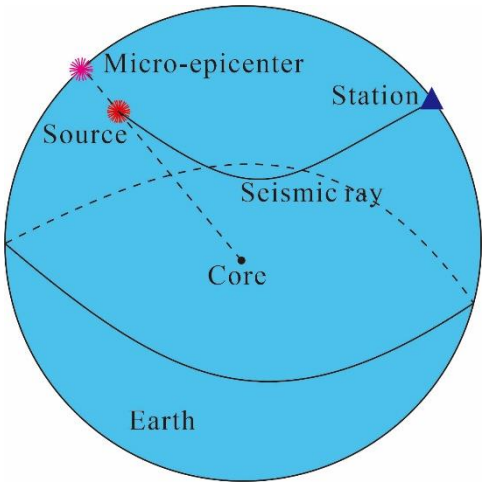


图 5-16 地震理论模型

本次实战目的是根据少量高质量台站数据（有标签）与大量低质量台站数据（无标签），利用半监督 K-means 聚类方法确定微观震中位置。

5.2.2 震中测定方法

根据《地震学》的学习与思考，震中位置确定方法可以分为单台法、多台法、以及虚波速度法等。本次实战的理论基础是针对远震震中确定的多台法即震中交切法（图 5-17）。

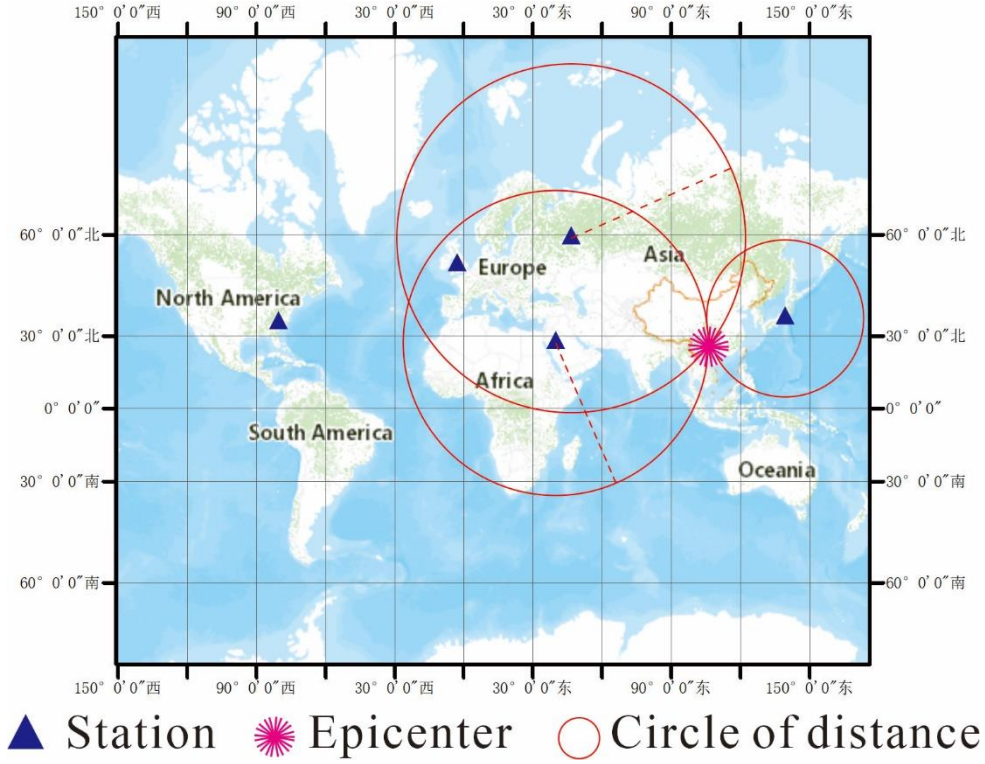


图 5-17 理想震中交切法

5.2.3 理想震中交切法

震中交切法的必要要素为：台站（Station）、震中距（Distance of epicenter 简记为 Distance）、辅助圆（Circle of distance）。

实际操作方法是根据多个地震台站的观测资料，计算出震中距，然后以台站为中心，台站对应的震中距为半径画辅助圆，最少有三个辅助圆相交于一点，那么该点就可以视为震中位置或者称为微观震中位置。

5.2.4 震中距计算方法

针对台站的观测资料质量高低，可以将台站分为高质量台站与低质量台站。

（1）低质量台站

较差的台站可以获得的信息有地震 P 波的到时T_p与地震 S 波的到时T_s（图 5-18），计算到时差T_s-T_p，根据当地台站的走时表就可以查出震中距（震中到台站的距离），根据震中距画圆，并采用震中交切法可以得到**震中的大致位置**。

（2）高质量台站

质量较高的台站还可以获得 P 波或 S 波的地动位移资料，地动位移可以很好的反应地震波传播的方向，也就是将地动位移分解为三个分量（图 5-19），然后根据地震射线到达地面的角度就可以推算出震中的方向，然后根据震中距就可以得到比较可靠的**伪震中**。

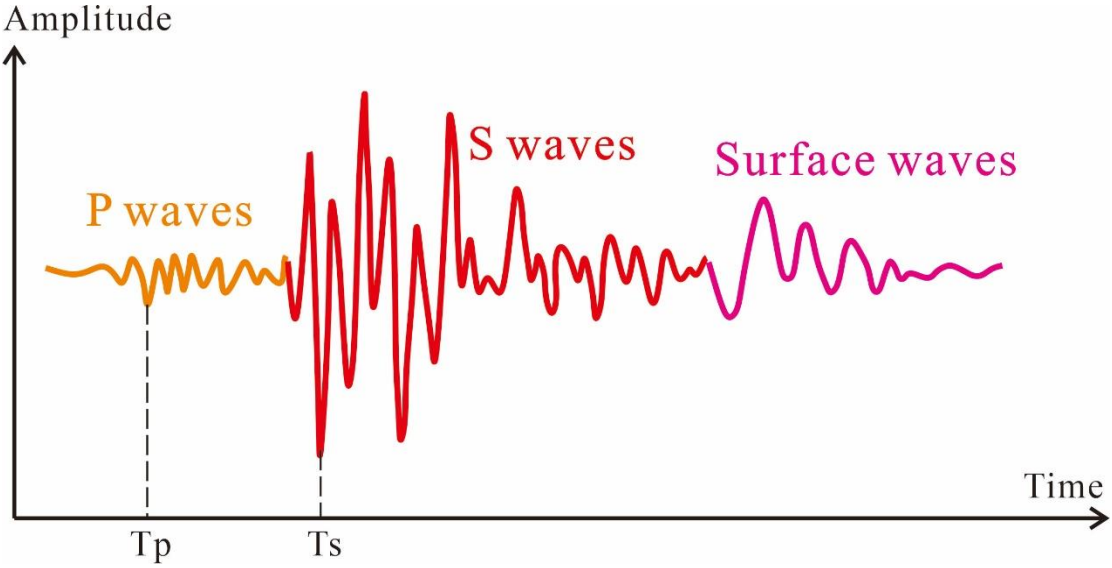


图 5-18 地震波形记录

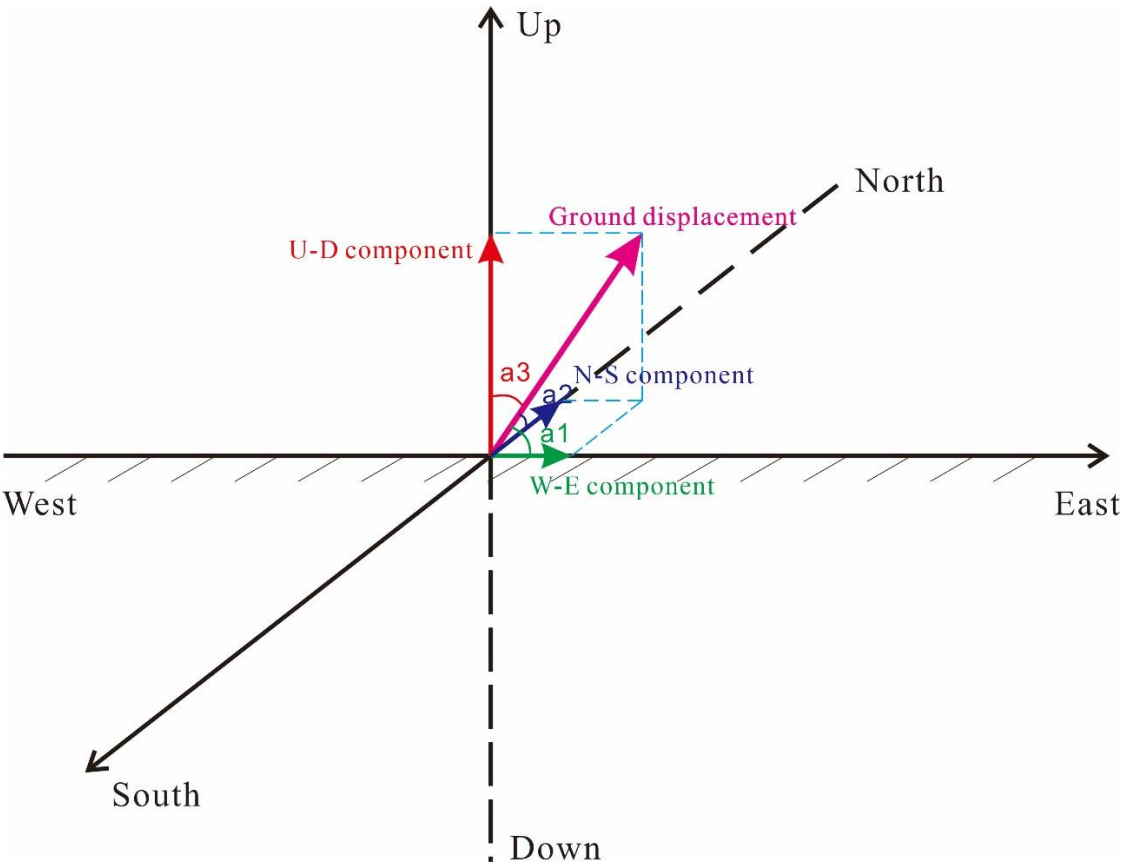


图 5-19 地动位移及其三分量

5.2.5 实际震中交切法

对于远震（震中距 Distance>1000Km）、**低质量台站**条件，由于不同地区台站测量误差以及处理、地球曲率、交汇计算等诸多因素的影响，三个台站的辅助圆一般不会交切到一点，而是会交于三个点，这三个交点可以将其称为 1 型伪震中（Puppet Epicenter 1）（图 5-20）。

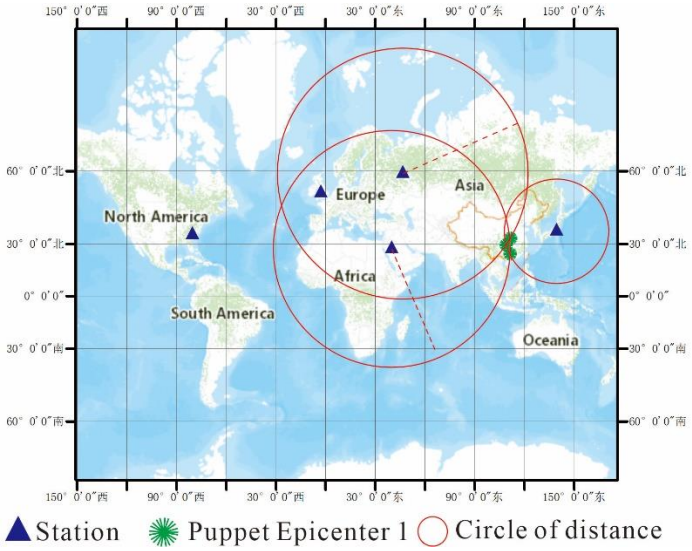


图 5-20 1 型伪震中位置

对于远震（震中距 Distance>1000Km）、**高质量台站**条件，可以获得地震波传播的方位，即震中的具体方位，震中距资料又可以查得到，那么由此确定的震中位置可以称为 2 型伪震中（Puppet Epicenter 2）（图 5-21）。显然这类伪震中位置的确定误差仅仅是震中距计算与测量误差，要比 1 型伪震中**更精准**。

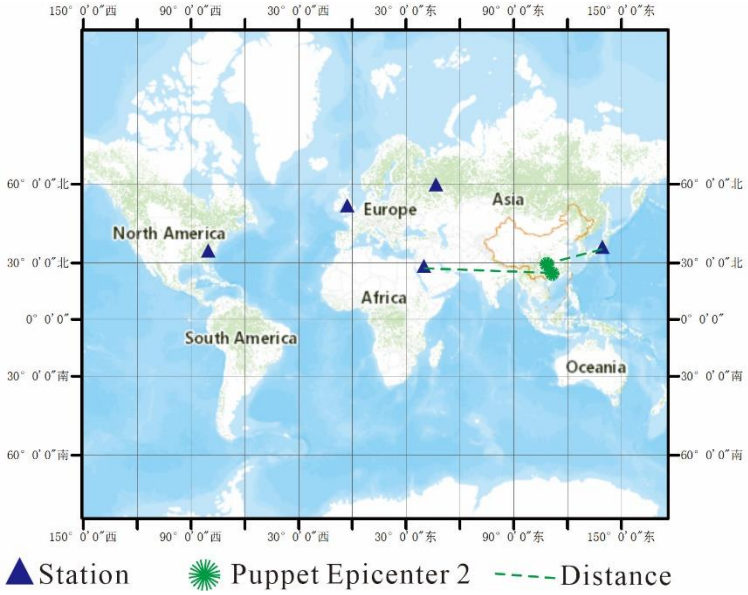


图 5-21 2 型伪震中位置

5.2.6 半监督 K-means 聚类算法确定震中位置

实际数据获取因为时间关系无法充分保证，但由于通过数据计算得到的震中位置是一系列离散的点位，那么我们可以通过在实际地图上标出少量的距离紧凑的点位来替代高质量台站确定的较高精度的 2 型伪震中位置（图 5-22 黄色），用较多的较离散的点来代替低质量台站确定的较低精度的 1 型伪震中位置（图 5-22 蓝色）。以喜马拉雅周边地震带的成都—重庆附近区域为例（我将**程序放到附件 3**）。

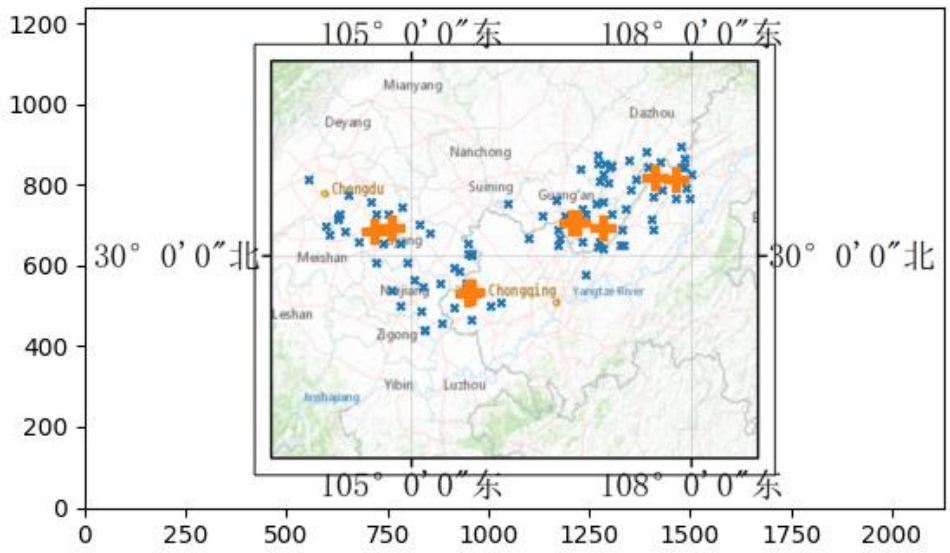


图 5-22 成都—重庆伪震中位置（蓝色是 1 型，黄色是 2 型）

5.2.7 K-means 聚类算法处理有标签数据

对于精度较高的台站计算出来的 2 型伪震中位置，可以将其视为有标签数据，但这个标签不需要打，因为其具有明显的簇分类趋势，那么利用 K-means 算法处理有标签数据，就可以得到第一类聚类中心（图 5-23）。（这里把**程序放到附件 4**）。

为方便编程，训练数据为把这些点数对应缩小与平移，但不改变其相对分布。

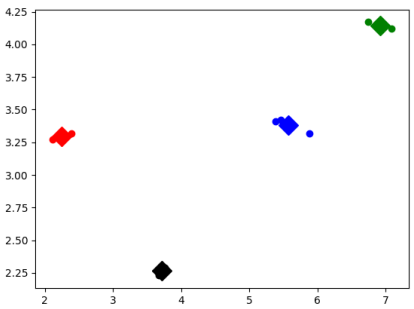


图 5-23 有标签数据（高精度）训练结果

我们将该质心投影到地质区划图上（图 5-24）（**程序放到附件 5**）：

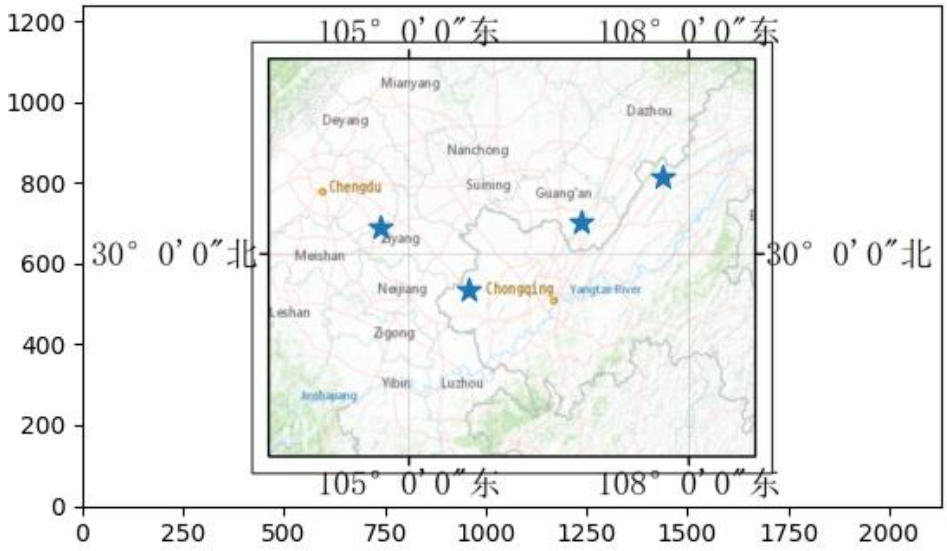


图 5-24 有标签数据（高精度）训练质心投影

5.2.8 K-means 聚类算法处理无标签数据

已经有了第一轮数据的质心，那么训练无标签数据时，以该系列质心为初始质心未知，对无标签数据进行训练，代码仍以附件 4 为主，只需修改初始质心的获取方式即可。训练得到的质心稳定有四个，这也就说明了半监督比无监督的稳定性好很多（图 5-25，图 5-26）。

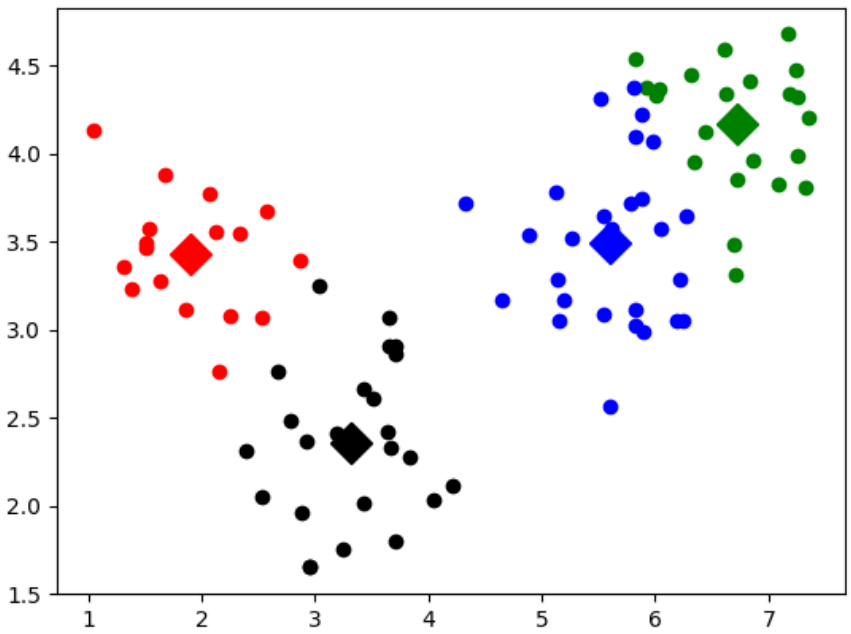


图 5-25 **半监督**训练无标签数据质心分布

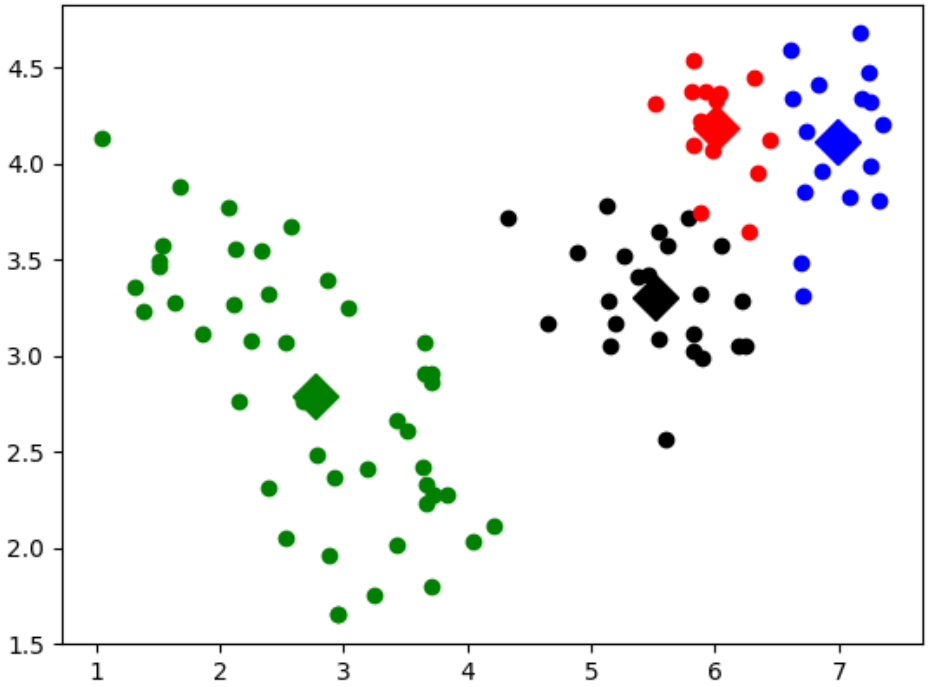


图 5-26 **无监督**训练无标签数据质心分布

最后把两轮训练的震中位置即所有伪震中位置绘制到地震图上进行对比（**我把程序放到附件 6**）（图 5-27）。（**写文件、读文件等的程序放到附件 7、8**）

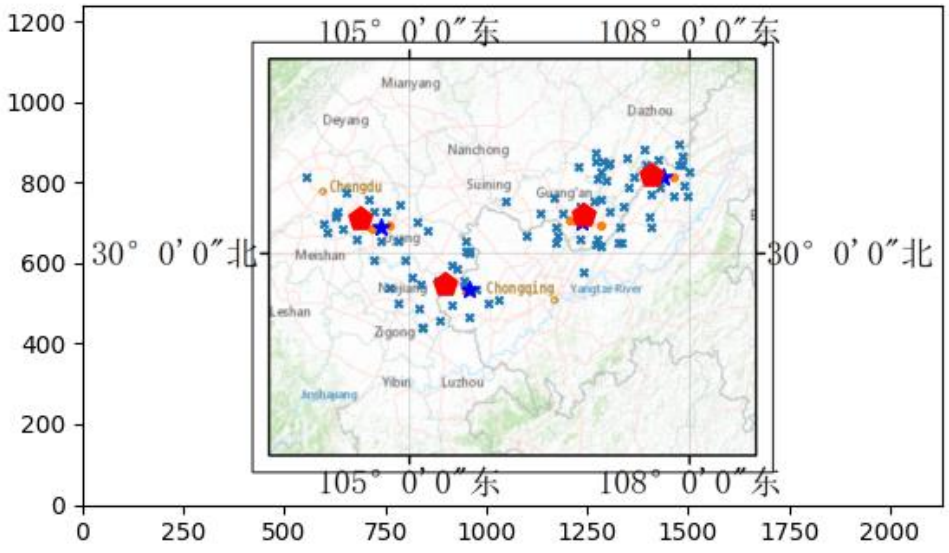


图 5-27 **红色五边形**为最终震中位置

最终我们完成了利用 K-means 算法来求取一定量台站数据背景下的地震震中位置。

6 问题与解决

本次报告构思与编程过程遇到的问题数不胜数，其弯路或错误都已经再程序中做好注释，整个编程经历可以在 CSDN 账户：yeahamen 中见到。不断地编程尝试解决了几乎所有的问题，并且得到的满足结果均在报告的其他部分展示，此处不在赘述。

7 学习/课程体会

王岩：

思想方面：编程与编写报告过程是十分辛苦耗时的，回想对《人工智能基础》的构思已经过去了一个多月的时间，编程与报告撰写少说也占据了完整的 7 日，程序中总是会有很多小错误，不断地修改、等待运行结果的过程实在是锻炼人的心性。结合大创写论文的经历，与编程解决关于本专业的某个简单问题进行对比，这都是需要做大量的工作。有时会对科研工作的热情泄气，完成一项令自己满意的任务实在是累。这可能就是磨练的过程吧！当然，奋斗的路上总是会有失败，正所谓胜败乃兵家常事，只要坚持不懈，一切困难都将变为冢中枯骨！

知识方面：熟练了 CNN 与 K-means 算法的核心原理当然是不需要多说的，利用这些算法来解决目前遇到的感兴趣的问题是十分令人高兴的。但起初的研究目标是强化学习中的生成对抗网络，对于这个网络我有大致的理解：设置一个模型，将数据（比如说地震资料）输入模型得出输出结果，把输出结果与带标签的样本同时传送给鉴别器，鉴别器区分输出结果的真假，鉴别器与生成器不断对抗学习，使得生成器输出的结果越来越真，就达到了研究目的。但这个网络的搭建遇到了很多困难，没有成功实现。当然对于强化学习的兴趣是来源于一位学者发表的将强化学习应用到本专业的断层识别领域（DOI：10.1190/geo2021-0383.1）的一篇文章。通读完毕之后，发现这篇文章的创新点似乎就是将人工智能算法应用到地球物理领域，这种 AI 与传统行业的交叉研究似乎非常火热，本人也将不断追逐，以在研究生阶段从事 AI 在地球动力学中的应用为目标，加油！

郭文静：

本次实验选择了卷积神经网络和半监督 K-means 算法作为研究对象，将在人工智能课程学习到的知识具体应用到地球物理专业中，详细地讲解了利用卷积神经网络来鉴别岩心和利用半监督 K-means 算法来确定震中的位置的问题，并取得了不错的效果。卷积神经网络是个复杂的模型，训练起来非常耗时，有很多细节需要研究者去选择，同时，需要自己收集数据集进行处理；同时，在编程方面，我们也不断试错，重复修改错误代码。尽管遇到了许多难题，但小组成员一起讨论，及时查阅资料，最终合力完成了此次报告。虽然过程是曲折艰辛的，但好在结果是令人满意的，此次实验我收获颇多。首先，我知道了人工智能与本专

业并不是毫无关联的两个方向，相反，人工智能的应用反而提高了我们解决各种专业问题的效率，相信未来人工智能与各个方面的融合应用也将成为一种势在必行的大潮流；其次，此次实验锻炼了我处理问题，解决问题的能力，同时，让我对 CNN、K-means 算法及其处理问题的过程有了更透彻的理解，提高了自己的知识水平；最后，此次实验让我认识到合作的魅力所在，我们小组成员齐心协力，互相帮助，最终圆满地完成了此次报告！

杜双全：

本次报告，我们选取了与专业相关且是我们最近在学习的课程中关于机器学习的卷积处理和地震学中震中距离确定的方法来研究。我们基于用 CNN 识别岩心及半监督 k-means 确定震中位置为研究方向。首先我们了解研究背景，明确了如何使用 CNN 识别进行卷积的原理方法和 k-means 半监督方法。在报告的撰写过程中，我明白了撰写报告需要严谨每一步，每个的原理都需要进行充分的理解，不能一知半解。所以在此次学习过程，我更加清楚的认识了卷积的过程以及原理和半年度 k-means 确定震中的方法，。同时我们小组分工合作，对于不同的环节进行研究，在研究的过程中，我明白了知识的获取是不能走捷径的，他是需要时间学习的得到的；其次，在团队合作中，我们互相帮助，互相理解，友好的完成了此次报告，这对于我在今后的学习和研究，和研究有很大的作用。以后未来我们研究可能会更严谨，更加注重团队合作。此次学习我明白了自己的许多不足，更加深刻了解了 CNN 原理和半监督 k=means 算法及处理过程，对自己所学知识有了更深的了解。

8 附件（数据集）

8.1 CNN 识别岩心部分数据集

8.1.1 训练集部分（320 张四类）

0：膏岩　　1：灰岩

2：灰质膏岩　3：膏质灰岩

下面是标签文件：

label_y	image_name
0	1.jpg
0	2.jpg
0	3.jpg
0	4.jpg
0	5.jpg
0	6.jpg
0	7.jpg
0	8.jpg
0	9.jpg
0	10.jpg
1	11.jpg
1	12.jpg
1	13.jpg
1	14.jpg
1	15.jpg
1	16.jpg
1	17.jpg
1	18.jpg
1	19.jpg
1	20.jpg
2	21.jpg
2	22.jpg
2	23.jpg
2	24.jpg
2	25.jpg
2	26.jpg
2	27.jpg
2	28.jpg
2	29.jpg
2	30.jpg
0	31.jpg
1	32.jpg
3	33.jpg
3	34.jpg
3	35.jpg
1	36.jpg
1	37.jpg
2	38.jpg
3	39.jpg
1	40.jpg

3 41.jpg
2 42.jpg
0 43.jpg
2 44.jpg
3 45.jpg
2 46.jpg
1 47.jpg
1 48.jpg
3 49.jpg
3 50.jpg
2 51.jpg
2 52.jpg
3 53.jpg
3 54.jpg
1 55.jpg
0 56.jpg
0 57.jpg
0 58.jpg
0 59.jpg
0 60.jpg
0 61.jpg
0 62.jpg
2 63.jpg
2 64.jpg
0 65.jpg
0 66.jpg
0 67.jpg
0 68.jpg
3 69.jpg
3 70.jpg
0 71.jpg
0 72.jpg
0 73.jpg
0 74.jpg
0 75.jpg
0 76.jpg
2 77.jpg
0 78.jpg
0 79.jpg
0 80.jpg
0 81.jpg
0 82.jpg

2 83.jpg
0 84.jpg
3 85.jpg
0 86.jpg
0 87.jpg
2 88.jpg
0 89.jpg
2 90.jpg
0 91.jpg
2 92.jpg
2 93.jpg
2 94.jpg
3 95.jpg
0 96.jpg
0 97.jpg
0 98.jpg
0 99.jpg
0 100.jpg
2 101.jpg
2 102.jpg
0 103.jpg
0 104.jpg
2 105.jpg
2 106.jpg
2 107.jpg
0 108.jpg
0 109.jpg
0 110.jpg
0 111.jpg
0 112.jpg
0 113.jpg
0 114.jpg
3 115.jpg
3 116.jpg
3 117.jpg
2 118.jpg
2 119.jpg
2 120.jpg
2 121.jpg
2 122.jpg
1 123.jpg
1 124.jpg

2 125.jpg
2 126.jpg
2 127.jpg
3 128.jpg
1 129.jpg
2 130.jpg
0 131.jpg
3 132.jpg
3 133.jpg
2 134.jpg
2 135.jpg
2 136.jpg
0 137.jpg
2 138.jpg
3 139.jpg
3 140.jpg
1 141.jpg
0 142.jpg
2 143.jpg
0 144.jpg
2 145.jpg
3 146.jpg
2 147.jpg
2 148.jpg
2 149.jpg
2 150.jpg
3 151.jpg
3 152.jpg
3 153.jpg
3 154.jpg
2 155.jpg
3 156.jpg
2 157.jpg
2 158.jpg
2 159.jpg
2 160.jpg
3 161.jpg
2 162.jpg
0 163.jpg
3 164.jpg
2 165.jpg
3 166.jpg

0 167.jpg
3 168.jpg
0 169.jpg
0 170.jpg
2 171.jpg
0 172.jpg
2 173.jpg
3 174.jpg
0 175.jpg
2 176.jpg
2 177.jpg
2 178.jpg
0 179.jpg
0 180.jpg
0 181.jpg
0 182.jpg
0 183.jpg
0 184.jpg
0 185.jpg
0 186.jpg
0 187.jpg
0 188.jpg
0 189.jpg
2 190.jpg
2 191.jpg
2 192.jpg
2 193.jpg
0 194.jpg
0 195.jpg
3 196.jpg
0 197.jpg
0 198.jpg
0 199.jpg
0 200.jpg
2 201.jpg
2 202.jpg
2 203.jpg
2 204.jpg
1 205.jpg
1 206.jpg
1 207.jpg
1 208.jpg

1 209.jpg
1 210.jpg
1 211.jpg
1 212.jpg
1 213.jpg
1 214.jpg
1 215.jpg
1 216.jpg
1 217.jpg
1 218.jpg
1 219.jpg
1 220.jpg
1 221.jpg
1 222.jpg
1 223.jpg
1 224.jpg
1 225.jpg
1 226.jpg
1 227.jpg
1 228.jpg
1 229.jpg
1 230.jpg
1 231.jpg
1 232.jpg
1 233.jpg
1 234.jpg
1 235.jpg
1 236.jpg
1 237.jpg
1 238.jpg
1 239.jpg
1 240.jpg
1 241.jpg
1 242.jpg
1 243.jpg
1 244.jpg
1 245.jpg
1 246.jpg
1 247.jpg
1 248.jpg
1 249.jpg
1 250.jpg

1 251.jpg
1 252.jpg
1 253.jpg
1 254.jpg
1 255.jpg
1 256.jpg
1 257.jpg
1 258.jpg
1 259.jpg
1 260.jpg
1 261.jpg
1 262.jpg
1 263.jpg
2 264.jpg
2 265.jpg
2 266.jpg
2 267.jpg
2 268.jpg
2 269.jpg
2 270.jpg
2 271.jpg
2 272.jpg
2 273.jpg
2 274.jpg
3 275.jpg
3 276.jpg
3 277.jpg
3 278.jpg
3 279.jpg
3 280.jpg
3 281.jpg
3 282.jpg
3 283.jpg
3 284.jpg
3 285.jpg
3 286.jpg
3 287.jpg
3 288.jpg
3 289.jpg
3 290.jpg
3 291.jpg
3 292.jpg

3 293.jpg
3 294.jpg
3 295.jpg
3 296.jpg
3 297.jpg
3 298.jpg
3 299.jpg
3 300.jpg
3 301.jpg
3 302.jpg
3 303.jpg
3 304.jpg
3 305.jpg
3 306.jpg
3 307.jpg
3 308.jpg
3 309.jpg
3 310.jpg
3 311.jpg
3 312.jpg
3 313.jpg
3 314.jpg
3 315.jpg
3 316.jpg
3 317.jpg
3 318.jpg
3 319.jpg
3 320.jpg

8.1.2 测试集部分（40 张四类）

0：膏岩　　1：灰岩

2：灰质膏岩　3：膏质灰岩

下面是标签文件：

test_label_y	test_image_name
0	1.jpg
0	2.jpg
0	3.jpg
0	4.jpg
0	5.jpg
0	6.jpg
0	7.jpg
0	8.jpg
0	9.jpg
0	10.jpg
1	11.jpg
1	12.jpg
1	13.jpg
1	14.jpg
1	15.jpg
1	16.jpg
1	17.jpg
1	18.jpg
1	19.jpg
1	20.jpg
2	21.jpg
2	22.jpg
2	23.jpg
2	24.jpg

2 25.jpg
2 26.jpg
2 27.jpg
2 28.jpg
2 29.jpg
2 30.jpg
3 31.jpg
3 32.jpg
3 33.jpg
3 34.jpg
3 35.jpg
3 36.jpg
3 37.jpg
3 38.jpg
3 39.jpg
3 40.jpg

8.2 半监督 K-means 确定地震震中位置数据集

8.2.1 少量（区别大）的数据（相当于有标签）（无监督训练）

X_coordinate	Y_coordinate
758.5	692
716.5	684.5
950.5	528.5
958	536
958	536
1282	692
1219	707
1207	705.5
1411	819.5
1463.5	812

8.2.2 大量（区别小）的数据（相当于无标签）（半监督训练）

X_coordinate	Y_coordinate
758.5	540.5
779.5	501.5
722.5	608
779.5	654.5
878.5	555.5
737.5	656
928	585.5
949	654.5
652	776
842.5	441.5
949	630.5
956.5	630.5
1048	752
914.5	593
956.5	623
886	456.5
956.5	464
556	813.5
1097.5	669.5
1240	578
1268.5	752
787	744.5

914.5	495.5
800.5	608
842.5	441.5
1391.5	882.5
1475.5	896
1366	812
1489	792.5
1408	771.5
1307.5	729.5
1243	729.5
1232.5	741.5
1405	716
1282	756.5
1340.5	741.5
1333	687.5
607	678.5
646	686
1132	725
1274.5	660.5
1285	642.5
1336	651.5
1274.5	809
1301.5	843.5
1306	849.5
1498	765.5
1289.5	851
1273	875
1271.5	851
1463.5	768.5
1348	861.5
1486	866
1273	647
1328.5	651.5
1174	651.5
1231	657.5
1178.5	669.5
1171	687.5
974.5	536
946	557
950.5	543.5
832	488
1006	498.5

1031.5	510.5
679	660.5
830.5	702.5
626.5	714.5
631	729.5
596.5	698
626.5	717.5
839.5	549.5
1406.5	690.5
710.5	759.5
719.5	728
1189	722
1168	761
817	566
749.5	726.5
1394.5	845
1228	840.5
1424.5	855.5
1487.5	842
1477	845
1504	825.5
1429	788
854.5	681.5
1351	786.5
1297	804.5
1282	827
758.5	692
716.5	684.5
950.5	528.5
958	536
958	536
1282	692
1219	707
1207	705.5
1411	819.5
1463.5	812

9 附件（程序）

附件 1

```
'''导入库'''
from PIL import Image
import os
'''原始图像所在文件夹地址'''
path = "C:/Users/yeahamen/Desktop/custom_dataset/test/"#train
filelist = os.listdir(path)
total_num = len(filelist)
'''循环访问每张图片并改变像素大小为256*256'''
for i in range(total_num):
    jpg_name = path + str(i+1) + '.jpg'
    im1 = Image.open(jpg_name)
    im2 = im1.resize((256,256))
    '''把修改后的图像存储到指定文件夹'''
im2.save(os.path.join(r'C:/Users/yeahamen/Desktop/custom_dataset/test_revis
e',os.path.basename(jpg_name)))#train_revise
```

附件 2

```
#导入库
import os
import cv2
import torch
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from torchvision.io import read_image
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from torchvision import transforms
import tensorflow.keras as ka
import datetime
import tensorflow as tf
import os
import PySide2
from tensorflow.keras.layers import
Conv2D,BatchNormalization,Activation,MaxPooling2D,Dropout,Flatten,Dense
from tensorflow.keras import Model
import tensorflow as tf

'''加载数据集'''
#创建自定义数据集类，参考可见：http://t.csdn.cn/gkVNC
class Custom_Dataset(Dataset):
    #函数，设置图像集路径索引、图像标签文件读取
    def __init__(self, img_dir, img_label_dir, transform=None):
        super().__init__()
        self.img_dir = img_dir
        self.img_labels = pd.read_csv(img_label_dir)
        self.transform = transform

    #函数，设置数据集长度
    def __len__(self):
        return len(self.img_labels)

    #函数，设置指定图像读取、指定图像标签索引
    def __getitem__(self, index):
        #'所在文件路径+指定图像名'
        img_path = os.path.join(self.img_dir + self.img_labels.iloc[index,
1])
```

```
        #读指定图像
        #image = cv2.imdecode(np.fromfile(img_path,dtype=np.uint8),-1)
        image=plt.imread(img_path)
        #height,width = image.shape[0],image.shape[1]  #获取原图像的垂直方向尺寸和水平方向尺寸。
        #image = image.resize((height//4,width//4))

        #'指定图像标签'
        label = self.img_labels.iloc[index, 0]
        return image, label

'''画图函数'''
def tensorToimg(img_tensor):
    img=img_tensor
    plt.imshow(img)
    #python3.X必须加下行
    plt.show()

#标签指示含义
label_dic = {0: '膏岩', 1: '灰岩', 2: '灰质膏岩',3: '膏质灰岩'}
'''图像集及标签路径'''
label_path =
"C:/Users/yeahamen/AppData/Local/Programs/Python/Python310/train_label.csv"
img_root_path = "C:/Users/yeahamen/Desktop/custom_dataset/train_revise/"
test_image_path="C:/Users/yeahamen/Desktop/custom_dataset/test_revise/"
test_label_path="C:/Users/yeahamen/AppData/Local/Programs/Python/Python310/
test_label.csv"
#加载图像集与标签路径到函数
#实例化类
dataset = Custom_Dataset(img_root_path, label_path)
dataset_test = Custom_Dataset(test_image_path,test_label_path)

'''查看指定图像（18）'''
#索引指定位置的图像及标签
image, label = dataset.__getitem__(18)
#展示图片及其形状（tensor）
print('单张图片(18)形状：',image.shape)
print('单张图片(18)标签：',label_dic[label])

#批量输出
dataloader = DataLoader(dataset, batch_size=1, shuffle=True)
'''查看图像的形状'''
```

```
for imgs, labels in dataloader:
    print('一批训练为1张图片（随机）形状：',imgs.shape)
    #一批图像形状：torch.Size([5, 256, 256, 3])
    print('一批训练为1张图片（随机）标签：',labels)
    #标签：tensor([3, 2, 3, 3, 1])
    break
    #仅需要查看一批

'''查看自定义数据集'''
showimages=[]
showlabels=[]
#把图片信息依次加载到列表
for imgs, labels in dataloader:
    c = torch.squeeze(imgs, 0)#减去一维数据形成图片固定三参数
    d = torch.squeeze(labels,0)
    showimages.append(c)
    showlabels.append(d)
#依次画出图片
def show_image(nrow, ncol, sharex, sharey):
    fig, axs = plt.subplots(nrow, ncol, sharex=sharex, sharey=sharey,
figsize=(10, 10))
    for i in range(0,nrow):
        for j in range(0,ncol):
            axs[i,j].imshow(showimages[i*4+j])
            axs[i,j].set_title('Label={}'.format(showlabels[i*4+j]))
    plt.show()
    plt.tight_layout()
#给定参数
#show_image(2, 4, False, False)

'''创建训练集与测试集'''
dataloader_train = DataLoader(dataset, batch_size=320, shuffle=True)
for imgs, labels in dataloader_train:
    x_train=imgs
    y_train=labels
print('训练集图像形状：',x_train.shape)
print('训练集标签形状：',y_train.shape)
dataloader_test = DataLoader(dataset_test, batch_size=40, shuffle=True)
for imgs, labels in dataloader_test:
    x_test=imgs
    y_test=labels
print('测试集图像形状：',x_test.shape)
```

```
print('测试集标签形状：',y_test.shape)

'''将图像转变为网络可用的数据类型'''
X_test = x_test#这里保留是为了预测时查看原始图像
Y_test = y_test#这里保留是为了预测时查看原始标签
x_train,x_test =
tf.cast(x_train/255.0,tf.float32),tf.cast(x_test/255.0,tf.float32)
y_train,y_test = tf.cast(y_train,tf.int16),tf.cast(y_test,tf.int16)


#参考：http://t.csdn.cn/eRQX2
print('注意：',x_train.shape)
'''归一化灰度值'''
x_train = x_train/255
x_test = x_test/255

'''标签转为独热编码，注意：如果标签不是从0开始，独热编码会增加1位（即0）'''
y_train = ka.utils.to_categorical(y_train)
y_test = ka.utils.to_categorical(y_test)
print('独热后训练集标签形状：',y_train.shape)
print('独热后测试集标签形状：',y_test.shape)
#获取测试集特征数
num_classes = y_test.shape[1]

'''CNN模型'''
#输入256*256*3
model = ka.Sequential([ka.layers.Conv2D(filters =
32,kernel_size=(5,5),input_shape=(256,256,3),data_format="channels_last",ac
tivation='relu'),
                       #卷积252*252*32、卷积层；参量依次为：卷积核个数、卷积核尺
寸、单个像素点尺寸、使用ReLu激活函数、解释可见：http://t.csdn.cn/6s3dz
                       ka.layers.MaxPooling2D(pool_size=(4,4),strides =
None,padding='VALID'),
                       #池化1—63*63*32、最大池化层,池化核尺寸4*4、步长默认为4、无
填充、解释可见：http://t.csdn.cn/sES2u
                       ka.layers.MaxPooling2D(pool_size=(2,2),strides =
None,padding='VALID'),
                       #池化2—31*31*32再加一个最大池化层，池化核尺寸为2*2、步长默
认为2、无填充
                       ka.layers.Dropout(0.2),
                       #模型正则化防止过拟合，只会在训练时才会起作用，随机设定输入
的值x的某一维=0，这个概率为输入的百分之20，即丢掉1/5神经元不激活
```

```
                        #在模型预测时，不生效，所有神经元均保留也就是不进行
dropout。解释可见：http://t.csdn.cn/RXbmS、http://t.csdn.cn/zAIuJ
                        ka.layers.Flatten(),
                        #拉平432*648*32=8957952;拉平池化层为一个向量
                        ka.layers.BatchNormalization(),
                        #批标准化层，提高模型准确率
                        ka.layers.Dense(50,activation='relu'),
                        #全连接层1，10个神经元，激活函数为ReLu
                        ka.layers.Dense(num_classes,activation='softmax')])
                        #全连接层2，4个神经元（对应标签0-3），激活函数为softmax，
作用是把神经网络的输出转化为概率，参考可见：http://t.csdn.cn/bcWgu；
http://t.csdn.cn/A1Jyn
'''模型参数展示、编译与训练'''
model.summary()
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['ac
curacy'])
startdate = datetime.datetime.now()
#训练轮数epochs=n，即训练n轮
history =
model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=40,batch_s
ize=5,verbose=2)
#训练样本、训练标签、指定验证数据为测试集、训练轮数、显示每一轮训练进程，参考可
见：http://t.csdn.cn/oE46K
#获取训练结束时间
enndate=datetime.datetime.now()
print("训练用时:"+str(enndate-startdate))


#模型损失值与精度画图展示
#参考http://t.csdn.cn/fUdtO
print(history.history)
loss = history.history['loss']          #训练集损失
val_loss = history.history['val_loss']  #测试集损失
acc = history.history['accuracy']            #训练集准确率
val_acc = history.history['val_accuracy']    #测试集准确率

plt.figure(figsize=(10,3))
plt.subplot(121)
plt.plot(loss,color='b',label='train')
plt.plot(val_loss,color='r',label='test')
plt.ylabel('Loss')
plt.legend()
```

```
plt.subplot(122)
plt.plot(acc,color='b',label='train')
plt.plot(val_acc,color='r',label='test')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.figure(2)
'''使用模型进行预测'''
for i in range(10):#在测试集中随机选10个
    random_test = np.random.randint(1,40)
    plt.subplot(2,5,i+1)
    plt.axis('off')#去掉坐标轴
    plt.imshow(X_test[random_test])#展示要预测的图片
    predict_image = tf.reshape(x_test[random_test],(1,256,256,3))
    y_label_predict = np.argmax(model.predict(predict_image))#使用模型进行预
测

plt.title('R_value:'+str(Y_test[random_test])+'\nP_value:'+str(y_label_pred
ict))#图名显示预测值与实际标签值进行对比
plt.show()
```

附件 3

```
import numpy as np
import matplotlib.pyplot as plt
#matplotlib inline
from matplotlib import image
from matplotlib import pyplot as plt
#单纯画图
def loadDataSet(fileName):
    dataMat = []              # 初始化一个空列表，文件的最后一个字段是类别标签
    fr = open(fileName)       # 读取文件
    for line in fr.readlines():   # 循环遍历文件所有行
        curLine = line.strip().split(' ')   # 切割每一行的数据
        fltLine = list(map(float, curLine))  # 映射所有的元素为 float（浮点
数）类型
        dataMat.append(fltLine)   # 将数据追加到dataMat
    return dataMat       # 返回dataMat
datMat1 = np.mat(loadDataSet('./nolabel.txt'))
#datMat1 = np.mat(loadDataSet('./after_label_training_center_expand.txt'))
data = image.imread('地图.png')
plt.figure(1)
plt.imshow(data,extent=(0, data.shape[1], 0, data.shape[0]))
nolabel_x = []
nolabel_y = []
yeslabel_x = []
yeslabel_y = []
x = list((datMat1)[:,0])
for i in range(0,len(x)-10,1):
    nolabel_x.append(x[i])
print(len(nolabel_x))
for i in range(len(x)-10,len(x),1):
    yeslabel_x.append(x[i])
print(yeslabel_x)
y = list((datMat1)[:,1])
for i in range(0,len(y)-10,1):
    nolabel_y.append(y[i])
for i in range(len(y)-10,len(y),1):
    yeslabel_y.append(y[i])
#画图形状可参考http://t.csdn.cn/wDfGc
plt.scatter(nolabel_x, nolabel_y,marker = 'x',s=10)
plt.scatter(yeslabel_x, yeslabel_y,marker = 'P',s=100)
plt.show()
```

附件 4

```
import numpy as np
import matplotlib.pyplot as plt
#matplotlib inline

# 解析文件，按空格分割字段，得到一个浮点数字类型的矩阵
def loadDataSet(fileName):
    dataMat = []              # 初始化一个空列表，文件的最后一个字段是类别标签
    fr = open(fileName)       # 读取文件
    for line in fr.readlines():   # 循环遍历文件所有行
        curLine = line.strip().split(' ')   # 切割每一行的数据
        fltLine = list(map(float, curLine))  # 映射所有的元素为 float（浮点数）类型
        dataMat.append(fltLine)   # 将数据追加到dataMat
    return dataMat       # 返回dataMat

# 计算欧几里得距离
def distEclud(vecA, vecB):
    return sqrt(sum(power(vecA - vecB, 2))) # 求两个向量之间的距离

# 构建聚簇中心，取k个(此例中为4)随机质心
def randCent(dataSet, k):
    n = shape(dataSet)[1]           #获取特征值数量：3列
    centroids = mat(zeros((k,n)))    #初始化质心为0，创建(k,n)个以零填充的矩阵k=4行,3列
    for j in range(n):              #循环遍历特征值
        #下面计算的分别是横坐标、纵坐标与类别标签
        minJ = min(dataSet[:,j])     #计算每一列的最小值
        maxJ = max(dataSet[:,j])     #计算每一列的最大值
        rangeJ = float(maxJ - minJ)  #计算每一列的范围值
        centroids[:,j] = minJ + rangeJ * random.rand(k, 1) #计算每一列的质心，并将值赋给centroids
    return centroids     #返回质心

def kMeans(dataSet, k, distMeans =distEclud, createCent = randCent):
    m = shape(dataSet)[0]#查看数据集行数100
    print(m)

    # clusterAssment包含两个列:一列记录簇索引值,第二列存储误差(误差是指当前点到簇质心的距离,后面会使用该误差来评价聚类的效果)
    clusterAssment = mat(zeros((m,2)))
```

```
    #centroids = randCent(dataSet, k)  # 创建质心,随机K=4个质心
    centroids = np.mat(loadDataSet('./after_label_training_center.txt'))  # 创建质心,随机K=4个质心
    clusterChanged = True   # 用来判断聚类是否已经收敛，启动初始循环

    while clusterChanged:
        clusterChanged = False;#只有全部的点都被分配完毕后才停止

        # 遍历所有数据找到距离每个点最近的质心,
        # 可以通过对每个点遍历所有质心并计算点到每个质心的距离来完成
        for i in range(m):#一共m行数据
            minDist = inf;#无穷
            minIndex = 1;#任意值

            for j in range(k):#第j个质心
            # 计算数据点到质心的距离
            # 计算距离是使用distMeans函数给出的距离公式,默认距离函数是distEclud欧几里得距离
                distJI = distMeans(centroids[j,:], dataSet[i,:])#第i个数据点与第j个质心比较
                #如果距离比minDist(最小距离)还小,更新minDist(最小距离)和最小质心的index(索引)
                #这里第一个肯定是要更新的，因为任何值都比无穷大，要小
                if distJI < minDist:
                    minDist = distJI; minIndex = j  # 如果第i个数据点到第j个中心点更近，则将i归属为j簇
            # 如果任一点的簇分配结果发生改变,则更新clusterChanged为true.
            if clusterAssment[i,0] != minIndex:
                clusterChanged = True;
            clusterAssment[i,:] = minIndex,minDist**2   # 更新簇分配结果为最小质心的index(索引),minDist(最小距离)的平方(误差)
        for cent in range(k):   # 重新计算中心点，遍历所有质心并更新它们的取值
            ptsInClust = dataSet[nonzero(clusterAssment[:,0].A == cent)[0]] # 通过数据过滤来获得给定簇的所有点
            centroids[cent,:] = mean(ptsInClust, axis = 0)  # 计算所有点的均值,axis=0表示沿矩阵的列方向进行均值计算
    return centroids, clusterAssment    #返回所有的类质心与点分配结果

def showCluster(dataSet,k,centroids,clusterAssment):
    m,n = dataSet.shape
    if n != 2:
        print("数据不是二维的")
```

```
        return 1
    #二维数据标准
    mark = ['or', 'ob', 'og', 'ok', '^r', '+r', 'sr', 'dr', '<r', 'pr']
    if k > len(mark):
        print("k值太大了")
        return 1
    #分类不太多
    # 绘制所有的样本
    for i in range(m):
        markIndex = int(clusterAssment[i,0])
        plt.plot(dataSet[i,0],dataSet[i,1],mark[markIndex])#数据点形状颜色

    mark = ['Dr', 'Db', 'Dg', 'Dk', '^b', '+b', 'sb', 'db', '<b', 'pb']#质心
与形状颜色
    # 绘制质心
    for i in range(k):
        plt.plot(centroids[i,0],centroids[i,1],mark[i],markersize=13)
    plt.show()

datMat1 = np.mat(loadDataSet('./data_epi_no_label.txt'))

datMat = np.mat(loadDataSet('./data_epicenter_labels.txt'))
k = 4

from numpy import *
centroids,clusterAssment = kMeans(datMat,k)
print('带标签训练质心：')
print(centroids)
centroids1,clusterAssment1 = kMeans(datMat1,k)
print('b不带标签训练质心：')
print(centroids1)

plt.figure(1)
showCluster(datMat[:,0:2],k,centroids,clusterAssment)
plt.figure(2)
showCluster(datMat1[:,0:2],k,centroids1,clusterAssment1)
```

附件 5

```
#仅画图
import numpy as np
import matplotlib.pyplot as plt
#matplotlib inline
from matplotlib import image
from matplotlib import pyplot as plt
#单纯画图
def loadDataSet(fileName):
    dataMat = []              # 初始化一个空列表，文件的最后一个字段是类别标签
    fr = open(fileName)       # 读取文件
    for line in fr.readlines():   # 循环遍历文件所有行
        curLine = line.strip().split(' ')   # 切割每一行的数据
        fltLine = list(map(float, curLine))  # 映射所有的元素为 float（浮点
数）类型
        dataMat.append(fltLine)   # 将数据追加到dataMat
    return dataMat       # 返回dataMat

datMat1 = np.mat(loadDataSet('./after_label_training_center_expand.txt'))
data = image.imread('地图.png')
plt.figure(1)
plt.imshow(data,extent=(0, data.shape[1], 0, data.shape[0]))
nolabel_x = []
nolabel_y = []
yeslabel_x = []
yeslabel_y = []
x = list((datMat1)[:,0])
y = list((datMat1)[:,1])
plt.scatter(x, y,marker = '*',s=100)
plt.show()
```

附件 6

```
#仅画图
import numpy as np
import matplotlib.pyplot as plt
#matplotlib inline
from matplotlib import image
from matplotlib import pyplot as plt
#单纯画图
def loadDataSet(fileName):
    dataMat = []              # 初始化一个空列表，文件的最后一个字段是类别标签
    fr = open(fileName)       # 读取文件
    for line in fr.readlines():   # 循环遍历文件所有行
        curLine = line.strip().split(' ')   # 切割每一行的数据
        fltLine = list(map(float, curLine))  # 映射所有的元素为 float（浮点
数）类型
        dataMat.append(fltLine)   # 将数据追加到dataMat
    return dataMat       # 返回dataMat

datMat1 = np.mat(loadDataSet('./after_label_training_center_expand.txt'))#1
轮震中

datMat2 = np.mat(loadDataSet('./two_train_xin.txt'))#2轮震中
datMat3 = np.mat(loadDataSet('./nolabel.txt'))#伪震中
data = image.imread('地图.png')
plt.figure(1)
plt.imshow(data,extent=(0, data.shape[1], 0, data.shape[0]))

x1 = list((datMat1)[:,0])
y1 = list((datMat1)[:,1])
x2 = list((datMat2)[:,0])
y2 = list((datMat2)[:,1])

nolabel_x = []
nolabel_y = []
yeslabel_x = []
yeslabel_y = []
x = list((datMat3)[:,0])
for i in range(0,len(x)-10,1):
    nolabel_x.append(x[i])
```

```
print(len(nolabel_x))
for i in range(len(x)-10,len(x),1):
    yeslabel_x.append(x[i])
print(yeslabel_x)
y = list((datMat3)[:,1])
for i in range(0,len(y)-10,1):
    nolabel_y.append(y[i])
for i in range(len(y)-10,len(y),1):
    yeslabel_y.append(y[i])
#画图形状可参考http://t.csdn.cn/wDfGc
plt.scatter(nolabel_x, nolabel_y,marker = 'x',s=10)
plt.scatter(yeslabel_x, yeslabel_y,marker = 'P',s=10)
plt.scatter(x1, y1,marker = '*',s=50,c='blue')
plt.scatter(x2, y2,marker = 'p',s=100,c='red')
plt.show()
```

附件 7

```
#读csv写txt
import csv        #导入csv模块

import pandas as pd
i=0
list1=[]
list2=[]
try:
    file=open('two_train_xin.csv','r')  #打开文件
except FileNotFoundError:
    print('文件不存在')
else:
    stus=csv.reader(file)         #读取文件内容
    for stu in stus:             #一行是一个数组
        print(stu[0])             #取每个数组的第一个元素
        list1.append(stu[0])
        list2.append(stu[1])
        i = i+1
        if i == 100:
            break
print(list1)
print(list2)
dataframe = pd.DataFrame({'label_y':list1,'image_name':list2})
#将DataFrame存储为csv,index表示是否显示行名，default=True
dataframe.to_csv("two_train_xin.txt",index=False,sep=' ',header =0)
```

附件 8

```
#读txt写csv
import pandas as pd

list1 = []
list2 = []
try:
    file = open('two_train_xin.txt', 'r')
except FileNotFoundError:
    print('File is not found')
else:
    lines = file.readlines()
    for line in lines:
        a = line.split()
        x = a[0]
        y = a[1]
        list1.append(x)
        list2.append(y)
file.close()
for x in list1:
    print(x)
print(list1)
print(list2)
dataframe = pd.DataFrame({'label_y':list1,'image_name':list2})

#将DataFrame存储为csv,index表示是否显示行名，default=True
dataframe.to_csv("two_train_xin.csv",index=False,sep=',',header = 0)
```