计算机视觉大作业2 Part 2:   
基于 GAN 和 CNN 的数字手势的生成与识别

**21121319 刘彦辰**

**目录**

[I. Introduction 简介 1](#_Toc136365548)

[A. Background 背景 1](#_Toc136365549)

[B. My Work 我的工作 1](#_Toc136365550)

[II. Related Work 相关工作 1](#_Toc136365551)

[III. Data Set 数据集 1](#_Toc136365552)

[A. Training Set And Validation Set 训练集和验证集 1](#_Toc136365553)

[B. Testing Set 测试集 2](#_Toc136365554)

[IV. Problem Statement 问题综述 2](#_Toc136365555)

[V. My Model 我的模型 2](#_Toc136365556)

[A. Deep Convolutional Generative Adversarial Network 深度卷积生成对抗网络 3](#_Toc136365557)

[B. Classifier Based on CNN 基于CNN 的分类器 4](#_Toc136365558)

[VI. Experiment 实验 4](#_Toc136365559)

[A. Fake Images Generation 虚假图像的生成 4](#_Toc136365560)

[B. Gesture Classify 手势分类 6](#_Toc136365561)

[VII. Limitations 缺点 7](#_Toc136365562)

[VIII. Conclusion 总结 7](#_Toc136365563)

[Reference 引用 8](#_Toc136365564)

# Introduction 简介

## Background 背景

这次项目我主要希望利用 CNN 和 OpenCV 实现一个实时数字手势识别程序. 数字手势是一种表示 0 – 9 数字的手势, 被广泛用于亚洲地区; 其中不同国家的手势不尽相同. 这种地区性的差别导致数字手势数据集的数量非常稀少, 特别是中国版本的数字手势数据集. 因此, 我希望通过一种方式在一定样本空间下生成更多的高质量手势图像, 加入训练集后对 CNN 进行训练.

一篇关于通过与 GAN(Generative Adversarial Network) [1][3] 联合训练人脸表情识别模型的论文[2]给了我一些启发. 由于人脸表情数据集有大量的未标注数据集, 不同表情的数据量分布也有很大不同, 论文作者构建了一个 GAN, 利用带标签数据集生成虚假的人脸表情图像, 并利用自适应方法提高了模型对人脸表情的检测准确率. 因此, 我认为在已有数字手势数据集中, 可以利用 GAN 生成特定手势加入训练集, 以提高 CNN 模型的准确率.

## My Work 我的工作

我的项目的优势在于, 网络上没有用 GAN 生成数字手势的例子, 因此模型架构和代码以及调参都需要我自己独立进行; 期间遇到了很多困难, 例如难以找到合适数据集, 生成图像质量较低等. 我尝试了各种方式对模型进行改进, 最终达到了比较理想的效果. 在百度 AI Studio data 提供的数字手势图像数据集[5]中, 我的方法训练出的模型的识别准确率达到了 0.9928; 在一个开源数字手势识别 app 项目[6]中的数据集中, 我的模型的准确率相对于直接使用 CNN 高出了 7.9%. 此外, 在对视频流的实时检测中, 我的模型也有较好的表现.

# Related Work 相关工作

在我能力范围内, 没有找到利用 GAN 生成数字手势并应用于实时识别的相关项目. 但是如上文所说, 本次项目的灵感来自于一篇关于联合训练进行人脸表情识别的论文[2]. 此外, 我还找到了南方科技大学学生做的数字手势识别项目[4]和一个数字手势识别app项目[6], 但是其中都提到了数据量少而效果欠佳的问题.

在我的项目中, 我希望通过已有的数字手势数据集生成额外的数字手势图像, 进而加强对识别手势的 CNN 的训练, 以此达到更高的识别准确率.

# Data Set 数据集

## Training Set And Validation Set 训练集和验证集

本次实验的训练集 (80%) 和验证集 (10%) 采用了百度 AI Studio data 提供的数字手势图像数据集[5]. 训练集中每张个数字手势有200张图像. 由于数据提供者是一所土耳其的学校, 只有数字 0, 1, 2, 4, 5 的数字手势与中国的数字手势相同, 因此这里仅做数字 0, 1, 2, 4, 5 数字手势的识别.

上述数据集的优点如下:

1. 由专业采集者采集, 图像质量较好.
2. 背景为白色桌面, 噪声较少.
3. 采集对象较多, 涵盖到不同特征.

缺点如下:

1. 数据数量不够多, 同种手势变化不够丰富.
2. 土耳其的数字手势与中国数字手势不同, 无法训练的数字手势有 3, 6, 7, 8, 9.

此外, 数据集还拆分了一部分 (10%) 作为测试集.

## Testing Set 测试集

测试集分为两部分, 第一部分来自于上节数据集的拆分. 这部分测试数据与上节提到的训练集来自于同一个数据集, 因此数据分布类似, 分类准确度在理论上较高.

第二部分来自另一个数字手势识别 app 项目[6]. 项目的开发者采集了亲友的215张数字手势照片, 然后利用数据增强技术(平移, 拉伸, 旋转)将215张数字手势照片扩充为21592张图片. 该项目的数据集有如下特点:

1. 由于图像来于亲友采集, 质量不统一, 背景较为复杂.
2. 99% 的数据来自于 1% 的图像的数据增强, 不能涵盖所有特征.

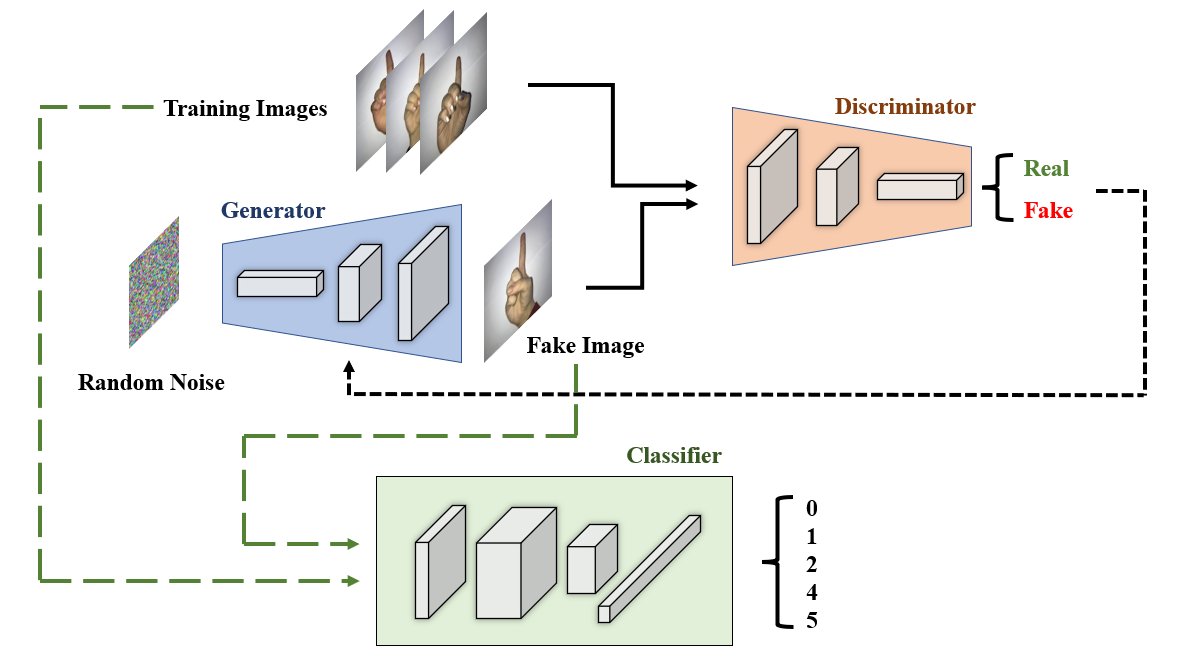
因此我利用这部分数据集模拟真实环境而测试模型的正确率. 这是一项有挑战性的尝试, 因为训练集的背景是单一的灰白色, 而上述项目的测试集的情况更加复杂, 拍摄的角度和对象都有一定程度的偏差; 因此分类的准确度在理论上会比第一部分低.

此外, 我还利用 OpenCV 调用电脑摄像头, 进行对视频流实时数字手势识别, 测试对象为我本人.

# Problem Statement 问题综述

简而言之, 本次项目的目的在于利用 CNN 和 OpenCV 实现中国数字手势的识别, 包括静态图像的识别和实时视频识别; 同时通过 GAN 技术对稀少的训练数据进行扩充, 用以增强识别模型的性能.

# My Model 我的模型



**Fig. 1.** 模型示意图. 我的模型由 Generator, Discriminator, Classifier 三个神经网络组成, 其中: Generator 根据高斯噪声生成 Fake Image; Training Images 和生成的 Fake Image 被送入 Discriminator 进行判别, 判别结果反馈给 Generator 形成竞争; 在一定迭代次数后将 Generator 生成的Fake Images 在经由 Discriminator 筛选后与 Training Images 一起送入 Classifier 训练, 最终得到能够区分 0, 1, 2, 4, 5 不同手势数字的分类模型.

Fig. 1. 展示了我的模型的架构, 主要的难点在于在 GAN 中如何提升生成图像的质量. 因此接下来对于模型的介绍将分为两部分. 第一部分将介绍 GAN 的搭建, 第二部分介绍 CNN Classifier 的实现.

## Deep Convolutional Generative Adversarial Network 深度卷积生成对抗网络

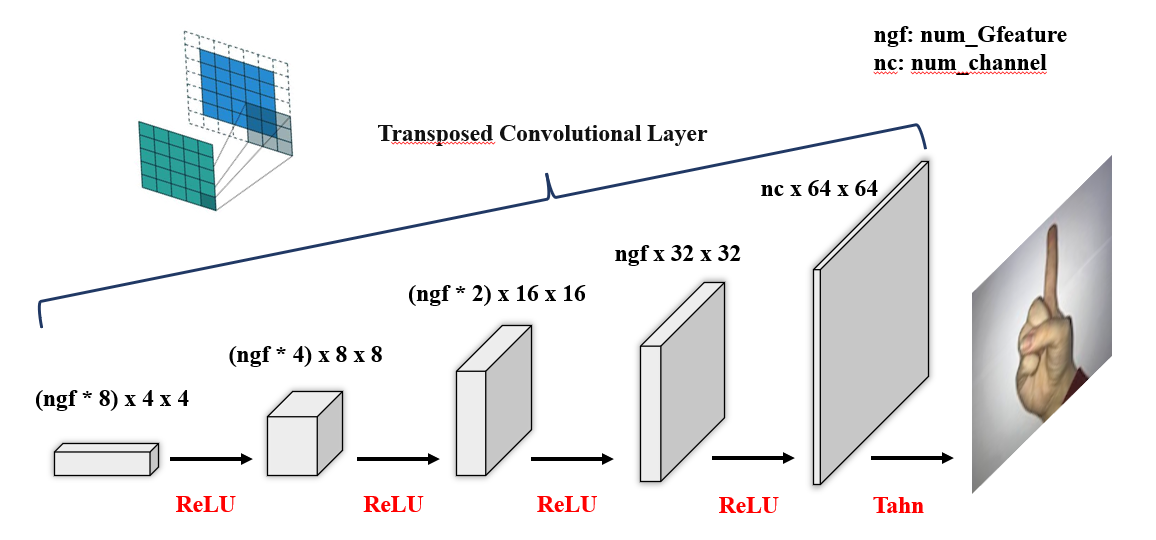
GANs 是一种用以生成图像的神经网络模型, 在2014年被 Ian Goodfellow 首先提出[1]. 在原始的 GAN 中, 模型有两个独立的网络模型: *Generator* 和 *Discriminator*. Generator 的作用是生成虚假的图像, Discriminator 用来判别生成的图像是真还是假, 将结果反馈给 Generator. 在 Fig. 1. 中, 我的模型也参考了这样的架构. 在训练过程中, Generator 总是通过优化参数来更好地 “欺骗” Discriminator (以假乱真), 而 Discriminator 总是通过优化参数来更好地识别出生成的虚假图像. 最后最优的情况是, 对于生成图像, Discriminator 对于真假的判断维持在 50% 的置信度.

在构造网络前, 我参考论文做了如下几个定义:

1. 表示 Discriminator 认为 来自于训练集而非 Generator 的概率. 因此如果输入图像 来自于训练集, 的值应该高; 反之应该较低. 可以将这个过程看作传统的二分类问题.
2. 设 是从标准正态分布中采样的潜在空间向量 (latent space vector), 因此 表示一个将 映射入数据空间 (data-space) 的函数. 换句话说, 给定 , 的结果是生成的虚假图像.
3. 根据上面的定义, 是 Discriminator 认为生成图像 是真实图像的概率.

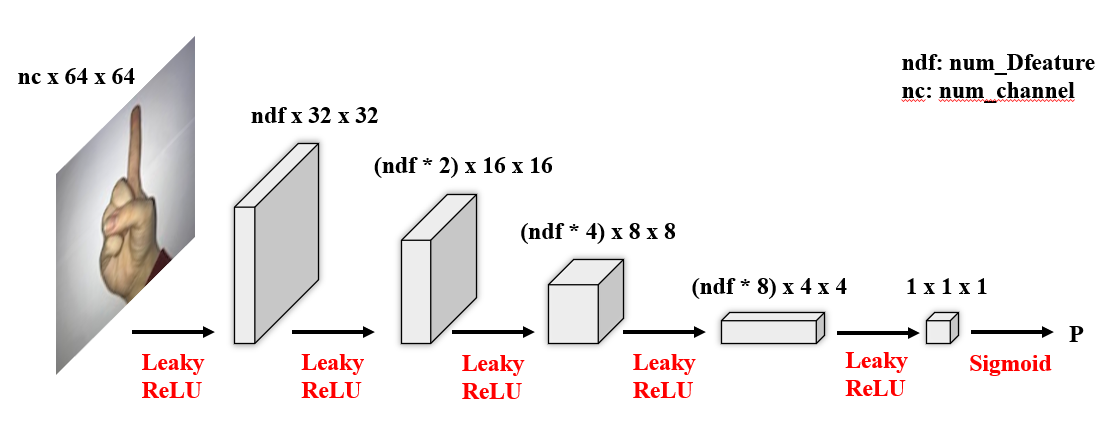
在 Goodfellow 的原论文[1]中, Discriminator 需要最大化 , 而Generator 需要最大化 . 于是损失函数被定义为:

因此我使用了 Pytorch 自带的 BCELoss 类计算 Loss.



**Fig. 2.** Generator Network 的架构. 高斯噪声向量经过多层反卷积层生成虚假图像. 反卷积层的激活函数使用了 ReLU; 输出时使用了 Tanh. 在每层反卷积层前还加入了 BN 层以提高神经网络的稳定性.

Fig. 2. 展示了我搭建的 Generator 网络, 主要由反卷积层 (Transposed Convolutional Layer) 进行图像生成. 反卷积是一种特殊的正向卷积, 先按照一定的比例通过补 0 来扩大输入图像的尺寸, 接着旋转卷积核, 再进行正向卷积.



**Fig. 3.** Discriminator Network 架构, 主要由多层卷积层进行正向卷积, 最终输出是真实图像的概率. 每层卷积层前也都加入了 BN 层以提高神经网络的稳定性.

Fig. 3. 展示了我搭建的 Discriminator 网络. 激活函数用到了 LeakyReLU, 可以避免“死亡神经元”问题. 在 ReLU 中, 如果输入是负数, 则输出为零; 这意味着该神经元不会对网络的输出产生任何影响, 导致在训练过程中, 这些神经元可能会停止更新. LeakyReLU 通过在输入为负数时输出一个小的斜率而不是零来解决这个问题; 这使得神经元在训练过程中保持激活状态, 并且可以更快地收敛.

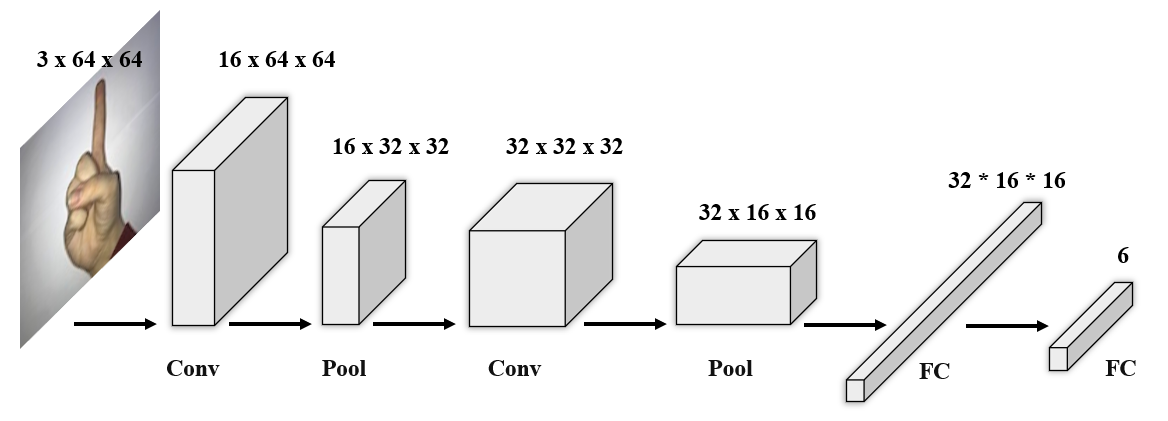
**Table I**

GAN 的超参数列表. 其中, beta1 是 Adam 优化器的超参数.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Hyperparameter** | **batch\_size** | **num\_channel** | **num\_Zlatent** | **num\_Gfeature** |
| **Value** | 128 | 3 | 110 | 74 |
| **Hyperparameter** | **num\_Dfeature** | **num\_epochs** | **learning\_rate** | **beta1** |
| **Value** | 64 | 100 | 0.000088 | 0.65 |

## Classifier Based on CNN 基于CNN 的分类器

在处理好数据集后, 我写了一个 CNN 用于特征提取与分类.



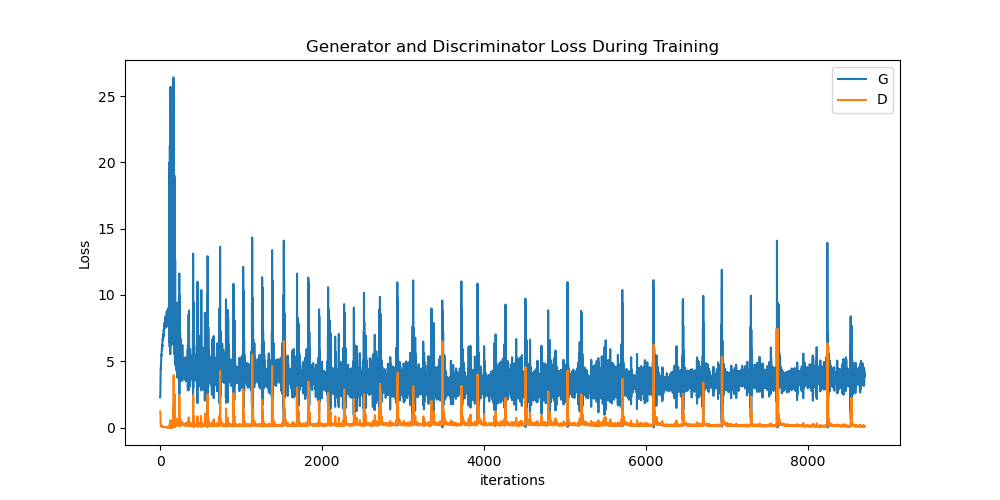
**Fig. 3.** Classifier 的架构. 在经过了两次卷积层+池化层提取特征后, 通过全连接层输出对应分类的概率. 激活函数用到了 ReLU.

如 Fig.3. 所示, 我构建的这个分类器就是利用一些常规的卷积操作将64\*64大小的图像展开为多维特征, 最后线性输出.

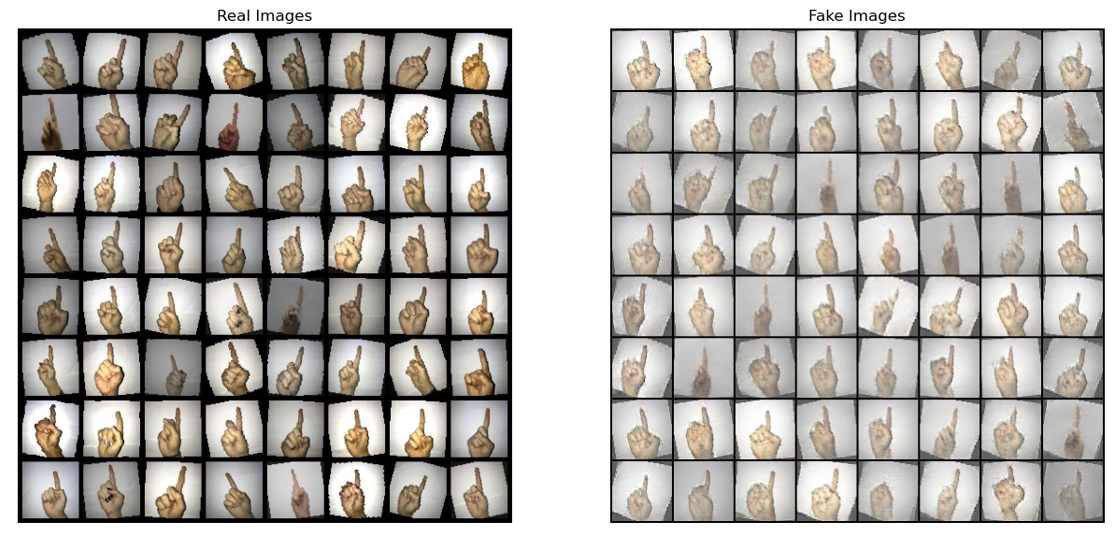
# Experiment 实验

## Fake Images Generation 虚假图像的生成

我将数据集中不同手势的图像分别传入 GAN 进行模型训练, 以下以数字 1 为例. 进行 大约9000 次迭代, Discriminator 和 Generator的 Loss 如 Fig. 4. 所示.



**Fig. 4.** GAN 训练过程中 Generator 和 Discriminator 的 Loss. 这里迭代了大于8000次. 实际上在迭代6000次后已经能够取得不错的效果.

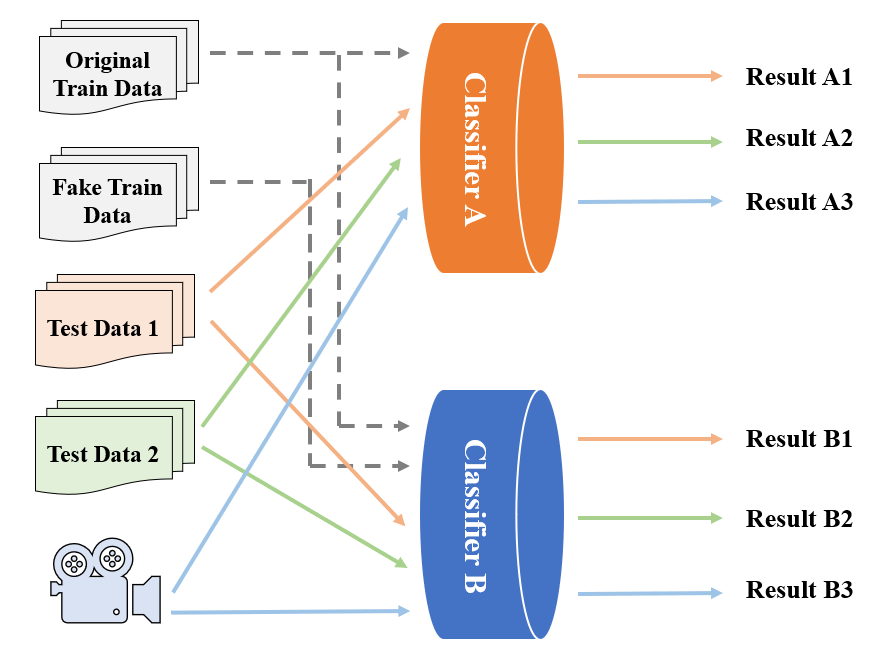


**Fig. 5.** GAN 的生成结果展示. 左图为训练集中的一部分图像, 右图为测试集中的一些图像. 这里的 GAN 仅由数字 1 的手势训练, 对于不同的手势, 需要训练不同的 GAN.

Fig. 5. 展示了一些原始图像和生成图像, 可以发现大部分生成的图像情况质量还行, 但是还是出现了一些质量不佳的情况. 因此在实际利用 Generator 模型生成图像时, 需要额外利用 Discriminator 进行判别, 即不选择置信度小于 0.5 的生成图像.

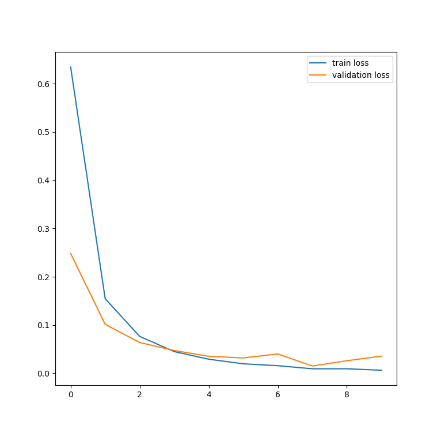
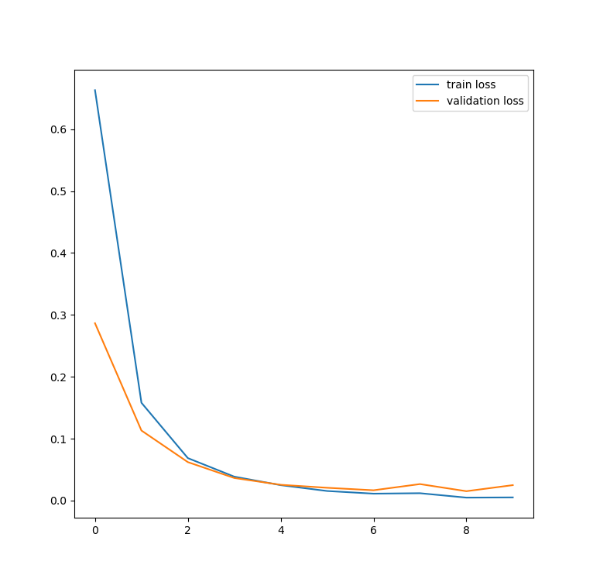
我额外提交了一份训练过程中图像在不同迭代次数下的变化视频作为附件.

## Gesture Classify 手势分类



**Fig. 6.** Classifier 的训练和测试流程. Classifier A 由 Original Train Data, 即由百度 AI Studio data 提供的数字手势图像数据集[5] (后称原数据集) 拆分出的 80% 训练以及原数据集 10% 验证. Classifier B 由 Original Train Data 和 Fake Train Data, 即由原数据集拆分出的 80% 和 GAN 生成的手势图像训练以及原数据集 10% 验证. Test Data 1 包含原数据集 10%; Test Data 2 包含数字手势识别 app 项目[6]提供的数据集; 最后还有摄像头实时检测手势作为额外测试. 综上, 有 6 种不同情况下的输出结果, 分别为 Result A1, Result A2, Result A3, Result B1, Result B2, Result B3.

Fig. 6. 展示了我对于分类器的训练和测试流程. 后文将用 Classifier A 和 Classifier B 表示不同的训练方式得到的模型.



**Fig. 7.** 左图为 Classifier A 在训练时的 train loss 和 validation loss; 右图为 Classifier B 在训练时的 train loss 和 validation loss.

由 Fig. 7. 可以看出, Classifier A 在训练时拟合程度较好, 在 epoch 5 之后略微出现过拟合现象, 因此选择本次训练 epoch 5 时得到的模型为结果模型; 同理, Classifier B 选择在 epoch 7 得到的模型. 测试结果如 Table II 所示.

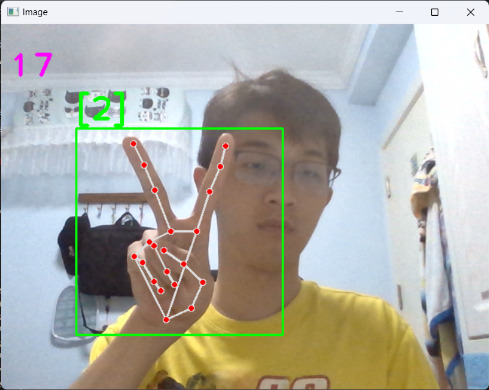
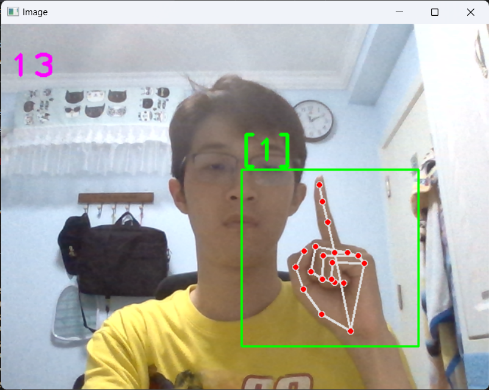
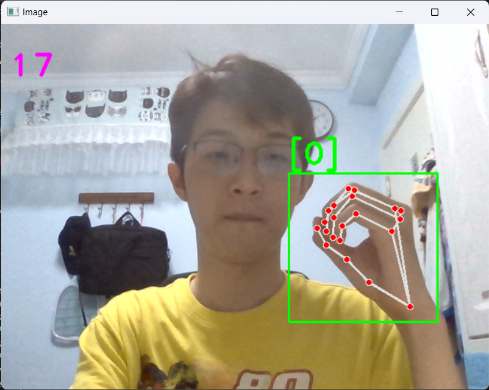
**Table II**

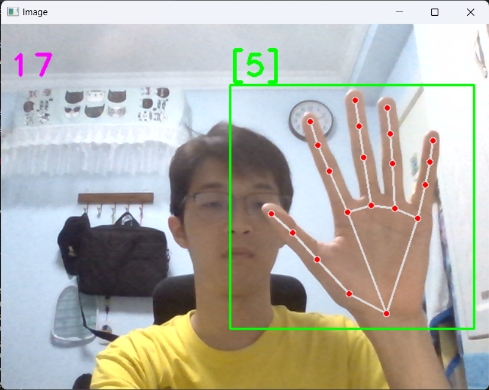
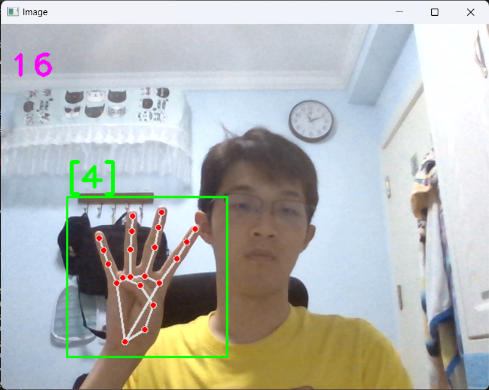
Classifier A 和 Classifier B 在不同测试集下的准确率

|  |  |  |
| --- | --- | --- |
|  | **Test Data 1 Accuracy** | **Test Data 2 Accuracy** |
| **Classifier A** | 0.991 | 0.383 |
| **Classifier B** | 0.993 | 0.415 |

如 Fig. 6. 所示, Classifier B 是我所构建的利用 GAN 生成图像训练的模型. 由于 Test Data 2 取自自制数据集, 图像质量不高, 而两个模型都是利用 Test Data 1 所在的数据集训练的; 因此模型的预测准确度较 Test Data 1 更低, 但是我的模型在这种情况下展现出更好的鲁棒性, 因为 GAN 生成的图像扩充了训练集的特征.

最后, 我利用 OpenCV 调取了电脑的摄像头, 进行实时的检测. 在对摄像头捕捉到图像进行分类时, 需要先捕捉到手部的位置. 这里我调用了 mediapipe 库中的手部捕捉方法, 得到手部的 21 个关键点的坐标后, 计算出了包含整个手的方形框; 然后将方形框单独像传入模型进行预测.





**Fig. 8.** 对于视频中手势的实时检测. 左上角粉色数字为帧率. 检测时, 计算出绿色框的位置和大小, 将框内图像传入 Classifier 2 进行分类, 分类结果将标在绿色框上方.

Fig. 8. 展示了部分识别时的截图. 需要声明的是, 分类工作完全由我的模型完成, 库函数做的仅有识别出手的位置.

# Limitations 缺点

经过测试, 我发现我的模型, 即 Classifier 2 对于数字手势 0 和 1 的分类效果不是特别理想. 我认为这又两个原因导致:

1. 训练集中的 0 和 1 手势样本不够丰富, 没有涵盖所有的形态和特征.
2. GAN 生成的 0 和 1 手势图像质量不够高.

我认为可以通过增加训练集中 0 和 1 的手势图像以及增加 GAN 的训练轮数, 微调超参数解决.

# Conclusion 总结

在本学期的课程中, 我们学习了多种图像处理和识别方式, 例如: PCA+KNN, HOG+SVM, CNN, R-CNN, YOLO, … 其中最吸引我的是最基础的 CNN, 因为在这学期之前我一直觉得神经网络和图像识别是 “触不可及” 的东西; 但是经过本学期的学习和实践后, 我从原理和实现两方面了解了当下最热门的算法的冰山一角.

在课程最后的项目中, 我通过自己设计搭建神经网络模型, 达到了预期的效果, 很有成就感. 我认为我的模型有一定实际意义, 同时我自己也得到了非常多代码能力上的锻炼, 对计算机视觉有了更深层的认识. 因此我认为自己在本学期的计算机视觉课程中得到了很多收获.

# Reference 引用

[1] Mirza, Mehdi, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014): 2672-2680.

[2] Zhang, Xi, Feifei Zhang, and Changsheng Xu. "Joint expression synthesis and representation learning for facial expression recognition." IEEE Transactions on Circuits and Systems for Video Technology 32.3 (2021): 1681-1695.

[3] Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." International conference on machine learning. PMLR, 2017.

[4] Jing (South China University of Technology) “Gesture Recognition with Pytorch and OpenCV: https://github.com/1240117300/pytorch-opencv-Gesture-Recognition”

[5] 数字0到9识别 “https://aistudio.baidu.com/aistudio/datasetdetail/29044”

[6] tz28 “Chinese Number Gesture Recognition: https://github.com/tz28/Chinese-number-gestures-recognition”