



电子科技大学

University of Electronic Science and Technology of China

计算机学院 • 人工智能应用与挑战课程

Northeastern SMILE Lab - Recognizing Faces in the Wild

项目结题报告

组名: Honkai StarRail

作者: 刘 *¹ 赵 **² 武 **³

指导老师: 文泉

¹电子科技大学计算机科学与工程学院数字媒体技术专业, 学号:2022**

²电子科技大学计算机科学与工程学院数字媒体技术专业, 学号:2022**

³电子科技大学计算机科学与工程学院计算机科学与技术专业, 学号:2022**

目录

1	前言	2
2	题目概况	2
	2.1 竞赛概况	2
	2.2 竞赛内容	2
	2.3 竞赛意义	3
	2.4 数据规模	3
	2.4.1 数据集	3
	2.4.2 提交格式	4
	2.5 现有结果	6
3	项目结题完成情况	7
	3.1 当前排名与结果	7
	3.2 结题模型设计	7
	3.2.1 开题设计	7
	3.2.2 中期设计	9
	3.2.3 结题设计	13
	3.3 结题最终结果分析	14
	3.4 结题主要问题分析	17
4	结题后拟改进措施	18
	4.1 增加新的特征提取模型	18
	4.2 优化集成学习过程	19
5	参与项目体会	19
6	意见与建议	20

1 前言

本篇报告为计算机学院 • 人工智能应用与挑战课程，Honkai StarRail 小组的项目结题报告，小组选题为“Northeastern SMILE Lab - Recognizing Faces in the Wild”，（详见网址：[小组选题](#)）。

本篇报告包括项目概况介绍、项目结题完成情况、结题后拟改进措施、参与项目体会、意见与建议五部分。

2 题目概况

2.1 竞赛概况

我们的选题名为“Northeastern SMILE Lab - Recognizing Faces in the Wild”：基于人脸识别的血缘关系鉴定模型（图：[\(1\)](#)）。本次竞赛的主办方为东北大学的微笑实验室⁴，该实验室的研究课题有“Effects of Speech Modification Strategies on Vowel Acoustics for Adolescents with Cerebral Palsy”、“Development of Auditory-Perceptual Speech Features in Preschoolers with Typical Development”、“Innovations in the Evaluation and Management of Pediatric Dysarthria”等⁵。

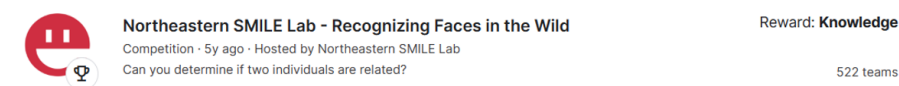


图 1: 竞赛题目

该竞赛的开始时间为 2019 年 5 月 14 日，结束时间为 2019 年 8 月 9 日，竞赛类型为 playground。虽然本次竞赛没有实物奖励，但由于该竞赛的题目较有意思，吸引了 522 支队伍、573 位选手参加该竞赛。

2.2 竞赛内容

在该竞赛中，题目要求参赛者训练一个模型图像识别模型，该模型能仅通过图像鉴定两人是否有血缘关系。由于用于亲属识别任务的现有图像数

⁴Northeastern SMILE Lab

⁵详见东北大学微笑实验室网址：<https://smilelab.sites.northeastern.edu>

据库规模不足以捕捉和反映世界各地家庭的真实数据分布，并且许多隐藏因素会影响家族面部关系，因此需要比计算机视觉算法更有区分度的模型，这也是本竞赛被提出的原因。

由于该竞赛考察的是参赛者模型的识别正确率，故该竞赛的评分标准为

$$Score = \frac{N}{M}$$

，其中 N 为正确识别血缘关系的测试集数量， M 为测试集总数。排行榜也是依据参赛者模型的识别正确率进行排名。

2.3 竞赛意义

当今社会，血缘关系鉴定有着多方面、多维度的社会意义：

- 当家庭中存在怀疑亲子关系的情况时，血缘关系鉴定有助于维护家庭的稳定与和谐。
- 当在法律上出现亲子权纠纷、抚养权争议等问题时，血缘关系鉴定有利于保障当事人的合法权益。
- 当对某些人的身份存疑时，如失踪人员寻亲、被拐卖儿童寻亲等，血缘关系鉴定可帮助确认个人身份与亲属关系。
- 在医学上可以帮助研究家族遗传病的遗传模式、基因突变的传播等，辅助控制家族遗传病的产生。

而现如今最常用的血缘鉴定方式为依据遗传学的基本原理，采用现代化的 DNA 分型检测技术来综合评判争议个体之间是否存在亲生、隔代或其他血缘关系⁶。这种血缘关系鉴定需提取被鉴定者的 DNA 样本，并通过 PCR 扩增、后 PCR 反应、毛细管测序仪检测等步骤完成血缘鉴定，鉴定过程较为繁琐。若能开发出一套基于被鉴定者的外貌特征而实现的亲子鉴定系统，则将大大提高血缘鉴定的效率，减少血缘鉴定的成本。

2.4 数据规模

2.4.1 数据集

官方给出的测试数据共包含 12 个文件（详见图 (2)），

⁶详见 <https://baike.120ask.com/art/50356>

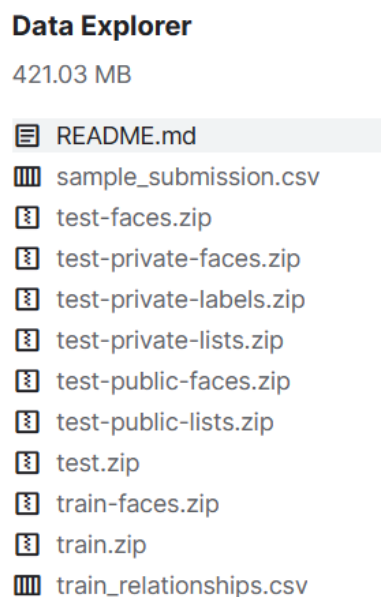


图 2: 测试文件

其中较为重要的分别是：

- train.zip，该文件为所有用于训练的人像图片。
- test.zip，该文件为所有用于测试的人像图片。
- train_relationships.csv，该文件中包含了所有含有血缘关系的图片标记，不在这个文件中的图片不含有血缘关系。
- sample_submission.csv，该文件为一个正确格式的示例提交文件。列 `img_pair` 描述了图像对，即 `abcd-efgh` 表示图像对 `abcd.jpg` 和 `efgh.jpg`。我们的目标是预测 `test-faces` 中每对图像是否相关，其中 1 表示相关，0 表示不相关。

以上数据文件总大小为 421.03MB。

2.4.2 提交格式

在该竞赛中，要求的提交格式如图 (3)：

```
img_pair,is_related
X3Nk6Hfe5x-qcZrTXsfde,0.0
X3Nk6Hfe5x-LD0pWDM8w_,0.0
X3Nk6Hfe5x-PHwuDtHyGp,0.0
X3Nk6Hfe5x-L061N_U4ot,0.0

etc.
```

图 3: 提交格式

其中 `img_pair` 为测试集中的图像对，图像对中有需要判断是否存在血缘关系的两张图片，`is_related` 则表示该图像对中的两人是否存在血缘关系，是为 1，否为 0。在官方给出的提交样例文件 `sample_submission.csv` 中，我们可以看到 `img_pair` 与 `is_related` 两个列向量的具体值，如图 (4)

	A	B
1	img_pair	is_related
2	face05508	0
3	face05750	0
4	face05820	0
5	face02104	0
6	face02428	0
7	face01219	0
8	face04262	0
9	face03697	0
10	face03524	0

图 4: 提交样例

但在我们实际的提交过程中发现，列向量 `is_related` 的数值可以不为 0 或 1，若其值为 `img_pair` 图像对具有血缘关系的可能性，这一文件也是可以被正确提交的，如图 (5)

	A	B
1	img_pair	is_related
2	face05508	0.006345
3	face05750	0.722867
4	face05820	0.536038
5	face02104	0.838642
6	face02428	0.461341
7	face01219	0.025343
8	face04262	0.282194
9	face03697	0.068675
10	face03524	0.66064

图 5: 实际提交文件

2.5 现有结果









#	△	Team	Members	Score	Entries	Last Solution
1	▲ 2	mattemilio		0.923	169	5y 
2	—	lynesyChen		0.920	173	5y
3	▲ 2	maki		0.919	40	5y
4	—	AlexeyK		0.918	64	5y 
5	▲ 10	pi-null-mezon		0.917	111	5y 

图 6: 排行榜 1-5 名

截至目前，共有 573 位参赛选手，522 支参赛队伍。由于该题在五年前已完赛，故参赛选手的最终提交时间均在五年之前。

在该题目的 leaderboard 中（见图 (6)），目前的世界第一为 mattemilio，其得分为 0.923 分，前五名得分均在 0.917 之上，前一百名得分均在 0.894 之上。他们的模型识别正确率均能达到九成左右。

3 项目结题完成情况

3.1 当前排名与结果

本小组最终提交的结果在 Private 测试集上的得分为 0.918, 排名第 5; 在 Public 测试集上的得分为 0.914, 排名第 6。






submission.csv				0.918	0.914
Complete (after deadline) · 43m ago					
4	—	AlexeyK		0.918	64
5	~ 10	pi-nuli-mezon		0.917	111
4		AlexeyK		0.916	64
5		maki		0.916	40
6		SCUT1111		0.914	115

图 7: 最终排名与结果

3.2 结题模型设计

3.2.1 开题设计

我们开题时使用的模型为 Google 开发的用于人脸识别与人脸验证的 FaceNet，它能够将输入的人脸图像转换为一个 128 维的向量，被称为人脸嵌入（face embedding），其中包含了该人脸的特征信息。FaceNet 模型的主要特点是它能够学习到高度区分性的人脸特征表示，使得不同人脸之间的距离在向量空间中能够很好地反映出它们之间的相似度。

FaceNet 模型的核心是一个卷积神经网络（CNN），具体结构框架见图 (8)，它以人脸图像作为输入，并通过多层卷积和池化操作，最终将输入图



图 8: FaceNet 结构框架

像映射到一个 128 维的向量空间中。在训练模型时采用了 triplet_loss（见图 (9)）来衡量训练过程中样本之间的距离误差，在这个损失函数中，每个

训练样本由三个人脸图像组成：锚点人脸图像（Anchor Face）、正样本人脸图像（Positive Face）和负样本人脸图像（Negative Face）。模型的目标是使得锚点人脸图像与正样本人脸图像的距离尽可能小，与负样本人脸图像的距离尽可能大，从而使得人脸嵌入在向量空间中能够很好地区分不同的人脸。简言之，在训练前或者在线学习中不断给神经网络制造“困难”，即一直在寻找与样本最不像的“自己”，同时寻找与自己最像的“他人”。通过随机梯度下降法，不断缩短自身所有样本的差距，同时尽可能拉大与其他人的差距，最终达到一个最优。通过这样一种嵌入学习（Embedding learning），能对原始的特征提取网络输出层再进一步学习，从而改善特征的表达。

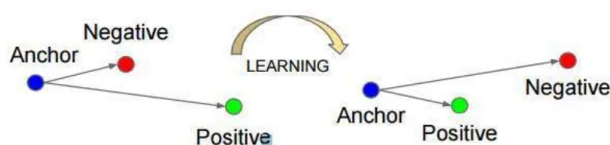


图 9: Triplet-Loss

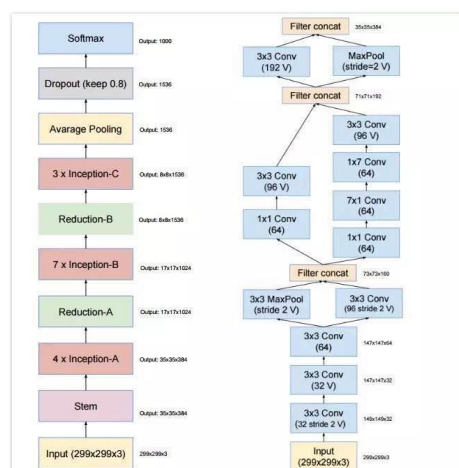


图 10: Inception-ResNet-v2 网络简化示意图

对于整个 FaceNet 结构，这里的特征提取可以当作一个黑盒子，可以采用各式各样的网络。最早的 FaceNet 采用两种深度卷积网络：经典 Zeiler&Fergus 架构和 Google 的 Inception v1。最新的 FaceNet 进行了改进，主体模型采用一个极深度网络 Inception-ResNet-v2（见图 (10)），由 3

个带有残差连接的 Inception 模块和 1 个 Inception v4 模块组成。

3.2.2 中期设计

我们中期使用的模型为基于 VGG-face 数据集预训练的 ResNet50 模型作为基础模型，并借助 Keras 深度学习框架构建了孪生神经网络结构。在模型构建过程中，我们去除了原始模型的输出层，保留了特征提取层，并设计了自定义的输出层模型，最终采用 sigmoid 层作为输出。这一设计旨在通过迁移学习，充分利用已有模型在大规模数据集上学习到的特征表达能力，从而提高模型的判断准确度。我们的目标是通过人脸图像进行血缘关系的判断，因此模型的输出层需要能够有效地区分不同人脸图像之间的相似度。采用孪生神经网络结构，可以使模型在学习过程中更好地捕捉到人脸图像之间的相似性特征，从而实现更精准的血缘关系判断。

1. ResNet50 模型

ResNet50 是 ResNet 系列中的一个特例，它有 50 层深，并且采用了残差模块的结构，克服了深度神经网络训练过程中的梯度消失和梯度爆炸等问题。通过使用残差连接，ResNet50 模型使得神经网络可以更深更广地构建，从而在图像识别等任务中取得了更好的性能。

ResNet50 模型的结构相对简单，主要由一系列的残差模块组成，其中包括了卷积层、批量归一化层和激活函数。整个网络的结构可以分为预处理层、卷积层、全局平均池化层和全连接层几个部分。具体结构见图 (11)

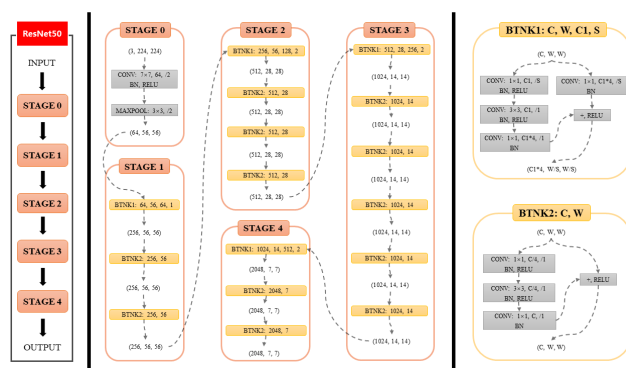


图 11: ResNet50 结构

2. 孪生神经网络

孪生神经网络 (Siamese Neural Network) 是一种特殊的神经网络架构, 最初用于解决用于度量学习 (Metric Learning) 的问题。孪生神经网络由两个完全相同的子网络组成, 它们共享相同的权重和参数。这两个子网络被称为” 孪生网络”, 因此得名” 孪生神经网络”。

孪生神经网络有两个输入 (Input1 and Input2), 将两个输入 feed 进入两个神经网络 (Network1 and Network2), 这两个神经网络分别将输入映射到新的空间, 形成输入在新的空间中的表示。通过 Loss 的计算, 评价两个输入的相似度。(详见图 (12))

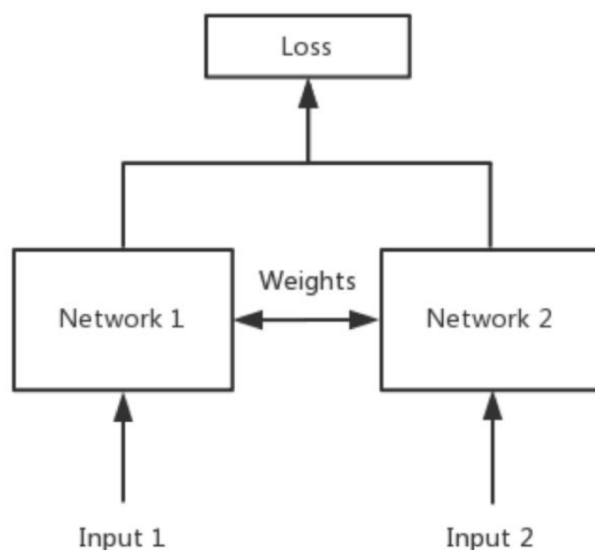


图 12: Siamese network

在训练 ResNet50 模型时, 首先, 我们划分了训练数据集 `train_images` 和验证数据集 `val_images`, (见图 (13)) 接下来我们定义一个用于读取图片的函数, 该函数需采用已经封装好的 `preprocess_input` 函数。此外, 我们还定义了一个生成器, 这个生成器的作用是生成训练样本和测试样本。并且要设定相同的正负样本数。通过使用 `yield` 语句, 能够在迭代的过程当中动态地生成数值。这种方式尤其适用于存在大量数据的情形, 它能够助力节省内存空间。

```

train_file_path = "../input/train_relationships.csv"
train_folders_path = "../input/train/"
val_families = "F09"

all_images = glob(train_folders_path + "*/*/*.jpg")

train_images = [x for x in all_images if val_families not in x]
val_images = [x for x in all_images if val_families in x]

train_person_to_images_map = defaultdict(list)

ppl = [x.split('/')[3] + "/" + x.split('/')[-2] for x in all_images]

for x in train_images:
    train_person_to_images_map[x.split('/')[3] + "/" + x.split('/')[-2]].append(x)

val_person_to_images_map = defaultdict(list)

for x in val_images:
    val_person_to_images_map[x.split('/')[3] + "/" + x.split('/')[-2]].append(x)

relationships = pd.read_csv(train_file_path)
relationships = list(zip(relationships.p1.values, relationships.p2.values))
relationships = [x for x in relationships if x[0] in ppl and x[1] in ppl]

train = [x for x in relationships if val_families not in x[0]]
val = [x for x in relationships if val_families in x[0]]

```

图 13: 划分训练数据集和验证数据集

```

# 定义神经网络模型
def baseline_model():
    input_1 = Input(shape=(224, 224, 3))
    input_2 = Input(shape=(224, 224, 3))

    for x in base_model.layers[:]:
        x.trainable = True

    x1 = base_model(input_1)
    x2 = base_model(input_2)

    x1 = Concatenate(axis=-1)([GlobalMaxPool2D()(x1), GlobalAvgPool2D()(x1)])
    x2 = Concatenate(axis=-1)([GlobalMaxPool2D()(x2), GlobalAvgPool2D()(x2)])

    x3 = Subtract()(x1, x2)
    x3 = Multiply()(x3, x3)

    x4 = Multiply()(x1, x1)
    x5 = Multiply()(x2, x2)
    x6 = Subtract()(x4, x5)

    x = Concatenate(axis=-1)([x6, x3])
    x = Dense(100, activation="relu")(x)
    x = Dropout(0.02)(x)
    out = Dense(1, activation="sigmoid")(x)
    model = Model([input_1, input_2], out)
    model.compile(loss="binary_crossentropy", metrics=['acc'], optimizer=Adam(0.0001))
    model.summary() # 输出网络结构日志
    return model

```

图 14: 模型定义

下面载入源自 VGG-face 数据集预训练的 ResNet50 模型，进而对模型予以定义。其基本思路为运用若干个 Concatante 层，之所以如此，是因为并不知晓哪种池化方法更为适宜，也不明确池化后的特征相乘后做差的效果更好，还是做差后相乘的效果更优，故而将这几种处理方式径直进行连接，交由网络自行学习。并采用二元交叉熵充当损失函数。该步骤的具体代码见

图 (14)。

然后我们定义了回调函数，用于保存模型和辅助模型训练，以及进行学习率的设置，代码见图 (15)

```
定义回调函数
file_path = "vgg_face.h5"

# 用于保存最佳模型
checkpoint = ModelCheckpoint(file_path, monitor='val_acc', verbose=1, save_best_only=True, mode='max')

# 用于跳出局部最优，使模型损失继续降低
reduce_on_plateau = ReduceLROnPlateau(monitor='val_acc', mode='max', factor=0.1, patience=20, verbose=1)

def scheduler(epoch):
    # 每隔20个epoch，学习率减小为原来的1/10
    if epoch % 20 == 0 and epoch != 0:
        lr = K.get_value(model.optimizer.lr)
        K.set_value(model.optimizer.lr, lr * 0.1)
        print("lr changed to {}".format(lr * 0.1))
    return K.get_value(model.optimizer.lr)

# 用于在每隔指定epoch后减小学习率
learningratescheduler = LearningRateScheduler(scheduler)

callbacks_list = [checkpoint, reduce_on_plateau]
```

图 15: 回调函数

最后实施训练调参操作。首先将 mode 设置为 ‘train’，每个 epoch 迭代 200 次，总计训练 100 个 epoch。待参数调好之后，再将 mode 设置为 ‘train_valid’，运用全部的数据集重新展开训练，每个 epoch 迭代 300 次，同样训练 100 个 epoch。

训练完成后，用该模型对测试集进行预测即可得到输出文件。

采用该模型，我们的得分 Private Score 为 0.868，在 Private 测试集上排名第 189，Public Score 为 0.862，在 Public 测试集上排名第 171（见图 (16)）。

Submission and Description		Private Score	Public Score	Selected	
	submission.csv Complete (after deadline) · 1h ago	0.868	0.862	<input type="checkbox"/>	
	188 38 samsudhin		0.869	4	5y
	189 25 Chinmay Vadgama		0.868	1	5y
	170 Captain Harlock		0.863	4	5y
	171 Or Com		0.862	7	5y

图 16: 中期排名与结果

3.2.3 结题设计

我们首先对中期的模型进行了修改，得到的模型识别正确率提升存在上限，但当我们浏览该比赛的讨论区时，发现排名靠前的队伍均使用了集成学习的思想。

集成学习通过结合多个基本模型的预测结果来提高模型的泛化能力和预测准确性。它通过将多个模型的预测结果进行加权平均、投票或者使用其他组合方法来做出最终的预测。集成学习的核心思想是“三个臭皮匠赛过诸葛亮”，即通过结合多个模型的优势来弥补单个模型的缺陷，从而获得更好的性能。

常见的集成学习方法包括：

1. Bagging（自举汇聚法）

通过随机采样训练数据集的子集来训练多个基学习器，然后将它们的预测结果进行平均或投票来得到最终的预测结果。常见的算法包括随机森林（Random Forest）。自举汇聚法可以降低模型的方差，提高模型的稳定性和泛化能力。还能减少过拟合的风险，因为每个基学习器都是在不同的训练子集上训练得到的，减少了模型对训练数据的过度依赖。

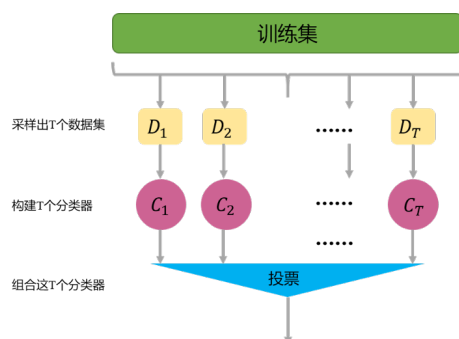


图 17: 自举汇聚法

2. Boosting（提升法）

通过串行训练多个基学习器，每个基学习器都试图修正前一个学习器的错误，从而逐步提升模型性能。常见的算法包括 AdaBoost、Gradient Boosting 等。提升法的核心是在每一轮迭代中，通过调整样本的权重

或者基学习器的权重，使得模型在前一轮中被错误分类的样本在下一轮中得到更多的关注，从而逐步减少模型的偏差。提升法在实际应用中被广泛使用，尤其在分类和回归问题中表现出色，能够处理复杂的非线性关系和高维数据集，并且具有很强的泛化能力。

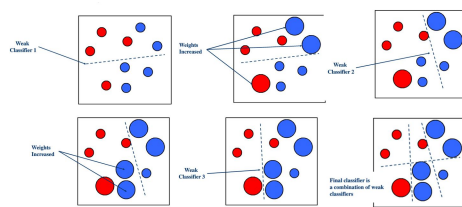


图 18: 提升法

3. Stacking（堆叠法）

将多个不同的基学习器的预测结果作为新的训练数据，然后训练一个元学习器（或称为组合模型）来预测最终的输出。堆叠法能够充分利用不同类型的基学习器的优势，提高模型的泛化能力和预测性能。也可以有效地减少过拟合的风险，因为元学习器通常是在一个新的特征空间上进行训练，而不是直接在原始数据上进行训练。但堆叠法的训练过程相对复杂，需要进行多轮的训练和预测，可能会增加模型的计算复杂度和内存消耗。

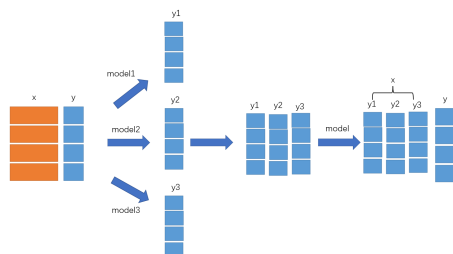


图 19: 堆叠法

3.3 结题最终结果分析

我们在中期使用的 ResNet50 模型基础上增加了 FaceNet 模型一起来提取特征，使提取出的特征更加全面，同时增加了新的特征处理方式，具

体过程如下：

- 进行图像预处理并且定义生成器函数，代码如图 (20) 所示

```
def read_img_fn(path):
    img = cv2.imread(path)
    img = cv2.resize(img, (IMG_SIZE_FN, IMG_SIZE_FN))
    img = np.array(img).astype(np.float)
    return prewhiten(img)

def read_img_vgg(path):
    img = cv2.imread(path)
    img = cv2.resize(img, (IMG_SIZE_VGG, IMG_SIZE_VGG))
    img = np.array(img).astype(np.float)
    return preprocess_input(img, version=2)

def gen(list_tuples, person_to_images_map, batch_size=16):
    ppl = list(person_to_images_map.keys())
    while True:
        batch_tuples = sample(list_tuples, batch_size // 2)
        labels = [1] * len(batch_tuples)
        while len(batch_tuples) < batch_size:
            p1 = choice(ppl)
            p2 = choice(ppl)

            if p1 != p2 and (p1, p2) not in list_tuples and (p2, p1) not in list_tuples:
                batch_tuples.append((p1, p2))
                labels.append(0)

        for x in batch_tuples:
            if not len(person_to_images_map[x[0]]):
                print(x[0])

        X1 = [choice(person_to_images_map[x[0]]) for x in batch_tuples]
        X1_FN = np.array([read_img_fn(x) for x in X1])
        X1_VGG = np.array([read_img_vgg(x) for x in X1])

        X2 = [choice(person_to_images_map[x[1]]) for x in batch_tuples]
        X2_FN = np.array([read_img_fn(x) for x in X2])
        X2_VGG = np.array([read_img_vgg(x) for x in X2])
```

图 20: 图像预处理与生成器函数

- 结合 ResNet50 模型和 FaceNet 模型的识别结果，将 ResNet50 模型得到的特征向量存储至 x_1 与 x_2 中，将 FaceNet 模型得到的特征向量存储至 x_3 与 x_4 中，并对特征向量进行 $x_1 + x_2$ 、 $x_3 + x_4$ 、 $x_1 - x_2$ 、 $x_3 - x_4$ 、 $x_1 \cdot x_2$ 、 $x_3 \cdot x_4$ 、 $x_1^2 + x_2^2$ 、 $x_3^2 + x_4^2$ 的一系列操作。代码如图 (21) 所示
- 将上述处理过后的特征向量合成为一个向量，以进行特征的融合。将合成后的向量输入至激活函数 ReLu 中，对特征进行线性变换和非线性映射，以增加模型的表达能力。然后在 Dropout 层中随机丢弃 10% 的神经元，以减少过拟合。减少密集连接层的单元数量，重复上述操作，最后在具有单个单元的密集连接层，使用 sigmoid 激活函数得到最终的位于 0,1 区间内的概率值。代码如图 (22) 所示

最终得到的结果在得分上有着显著的提升（详见图 (23)）


```

x1t = Lambda(lambda tensor : K.square(tensor))(x1)
x2t = Lambda(lambda tensor : K.square(tensor))(x2)
x3t = Lambda(lambda tensor : K.square(tensor))(x3)
x4t = Lambda(lambda tensor : K.square(tensor))(x4)

merged_add_fn = Add()([x1, x2])
merged_add_vgg = Add()([x3, x4])
merged_sub1_fn = Subtract()([x1,x2])
merged_sub1_vgg = Subtract()([x3,x4])
merged_sub2_fn = Subtract()([x2,x1])
merged_sub2_vgg = Subtract()([x4,x3])
merged_mul1_fn = Multiply()([x1,x2])
merged_mul1_vgg = Multiply()([x3,x4])
merged_sq1_fn = Add()([x1t,x2t])
merged_sq1_vgg = Add()([x3t,x4t])
merged_sqrt_fn = Lambda(lambda tensor : signed_sqrt(tensor))(merged_mul1_fn)
merged_sqrt_vgg = Lambda(lambda tensor : signed_sqrt(tensor))(merged_mul1_vgg)

```

图 21: 处理特征向量

```

merged = Concatenate(axis=-1)([Flatten()(merged_add_vgg), (merged_add_fn), Flatten()(merged_sub1_vgg), (merged_sub1_fn), Flatten()(merged_sub2_vgg), (merged_sub2_fn), Flatten()(merged_mul1_vgg), (merged_mul1_fn), Flatten()(merged_sq1_vgg), (merged_sq1_fn), Flatten()(merged_sqrt_vgg), (merged_sqrt_fn)])

merged = Dense(100, activation="relu")(merged)
merged = Dropout(0.1)(merged)
merged = Dense(25, activation="relu")(merged)
merged = Dropout(0.1)(merged)
out = Dense(1, activation="sigmoid")(merged)

model = Model([input_1, input_2, input_3, input_4], out)

model.compile(loss="binary_crossentropy", metrics=['acc'], optimizer=Adam(0.00001))

```

图 22: 神经网络的建立与结果输出

中期结果

Submissions and Rankings

Private Score

Public Score

100	0.868	0.862		
101	0.868	0.868	4	1y
102	0.868	0.868	1	1y
103	0.868	0.868	4	1y
104	0.868	0.868	7	1y

改进结果

NeuralVGG v2

0.885 0.887

104	0.885	0.887	10	1y
105	0.885	0.885	24	1y
103	0.885	0.885	24	1y
104	0.887	0.887	12	1y

Private Score: 0.868/189

Public Score: 0.862/171

Private Score 0.885, 排名为114;

Public Score 0.887, 排名为104.

图 23: 中期结果加入 FaceNet 模型后的提升

如上文所述, 我们浏览了该比赛的讨论区, 发现排名靠前的队伍均使用了集成学习的思想, 于是我们参考集成学习的思想, 重新定义了四个模型 (使用不同的特征提取模型、更改特征处理方式等), 再分别对这四个模型进行如上所述的训练, 使用训练好的模型对结果进行预测并提交, 观察该模型预测结果的得分, 通过公式 $y = \sum_{i=1}^5 w_i x_i$ 加权处理, 得分高的取较高权

重 w_i ，并再次将加权处理后的结果提交，观察得分是否变高，若变高，则将此次提交的文件替换原有公式中分数较低的那个文件，并重新分配权重 w_i ，再次得到新的预测结果，如此往复进行递归操作，当分数不再提高时停止。最终得到的结果在 Private 测试集上的得分为 0.918, 排名第 5; 在 Public 测试集上的得分为 0.914, 排名第 6。

经我们分析，能获得极大提升的原因如下：

1. **提升了模型的多样性：**以 FaceNet 模型和 ResNet50 模型为例，FaceNet 模型主要侧重于人脸特征的提取和表示，而 ResNet50 模型是一个通用的图像分类模型，这两种模型是不同的，它们具有不同的架构和特征表示能力。通过将它们集成在一起，可以增加模型的多样性，模型能够学习到更丰富的图像特征，从而提高模型的泛化能力。
2. **模型间的错误互相补偿：**由于 FaceNet 模型和 ResNet50 模型是两个不同的模型，它们可能会在不同的样本上产生不同的错误。通过将它们的预测结果进行加权平均或投票，可以减少错误的累积，从而提高模型的整体性能。
3. **降低了过拟合风险：**集成学习可以降低模型的方差，减少过拟合的风险。FaceNet 模型和 ResNet50 模型可能在不同的数据子集上表现更好，通过集成学习可以有效地减少模型的方差，提高模型的稳定性和泛化能力。
4. **提升了特征的丰富性：**FaceNet 模型和 ResNet50 模型分别学习了人脸特征和图像特征，通过将它们的特征表示进行组合，可以得到更丰富和更具区分性的特征表示，从而提高了模型的识别性能。

3.4 结题主要问题分析

在我们训练模型的过程中，主要遇见了以下问题：

1. **单一模型的识别能力存在上限：**无论我们怎么优化某一模型，其识别正确率总会存在上限，无法进一步地提高，识别正确率无法超过 90%。
2. **图像的特征提取不充分：**在我们使用 ResNet50 模型进行图像特征的提取时，受该模型的特征提取器影响，可能会遗漏图像中人脸的特征，进而导致血缘关系判断的错误。

3. **训练所用数据集质量不高：**官方给出的测试文件中的部分图片质量不高，有些存在大量噪点，有些人像的面部亮度不足，有些人像的面部被部分遮挡等等，这些因素都会导致我们所训练的模型识别正确率较低。

我们对这些问题进行了分析，并得出了相应的解决措施：

1. **采用集成学习的方式提升模型的泛化能力：**如上文所述，我们使用了五种模型得到的预测结果并对其进行加权处理，最终得到的预测结果相较单一模型有着很大的提升，不仅提高了我们的得分与排名，也使模型对图像人脸特征的提取更加充分。
2. **对训练集进行数据预处理：**我们在训练模型前先对训练集中的图像进行了预处理，对图像进行白化处理，使所有图片的亮度均衡化；对部分图像进行颜色通道对齐处理，对其该图像的 RGB 通道，提升图片质量；使用 YOLOv5 算法进行目标检测，识别脸部位置，便于人脸特征的提取。

4 结题后拟改进措施

4.1 增加新的特征提取模型

在查找提取图片特征的模型时，我们注意到一个名为 SENet50 的图片特征提取模型，它结合了 ResNet50 和 SE (Squeeze-and-Excitation) 模块这样就可以在学习特征的同时，自适应地调整特征图的权重，提高了模型的性能。在 SE 模块中，它通过对特征通道进行重新校准来提高网络的表现，使得网络更加关注对分类任务有用的特征。

SE 模块由两个步骤组成：压缩 (Squeeze) 和激励 (Excitation)：

1. **压缩：**SE 模块通过全局平均池化操作将输入的特征图压缩成一个特征向量，这个特征向量反映了每个通道的全局特征重要性。可通过公式

$$z_c = F_s q(u_c) = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W u_c(i, j) \quad (1)$$

实现全局信息的获取。

2. **激励：**特征向量经过一个全连接的神经网络，这个网络通常包括一个或多个全连接层，其输出是一个表示每个通道重要性的权重向量。这

个权重向量通过 Sigmoid 激活函数进行归一化，然后被应用到输入的特征图上。Sigmoid 激活函数如下式

$$s = F_e x(z, W) = \sigma(W_2 \delta(W_1 z))$$

$$\text{where, } W_1 \in \mathbb{R}^{\frac{C}{r} \times C}, W_2 \in \mathbb{R}^{C \times \frac{C}{r}} \quad (2)$$

通过引入 SENet50 模型，也许能够从图片中提取出更优质的特征，以用于后续的集成学习。

4.2 优化集成学习过程

由于时间有限，我们没有花费过多的时间进行集成学习，这可能导致我们最终的提交结果并未收敛，也许仍存在最优解。后续会对集成学习过程进行优化，以找到最优的权重值。

5 参与项目体会

• 刘 **:

在本次挑战课程的学习中，我实属收益颇丰。在理论课程中，我学到了近乎全面的人工智能领域理论知识，在实验课与 kaggle 项目中，我也运用了诸多人工智能的算法。

在近两年如火如荼的人工智能狂潮中，我也借此课程的机会，开始了人工智能方面的学习。通过本次 kaggle 项目-东北大学实验室提出的识别图像是否具有血缘关系项目，我们组使用了近几年较火诸如 FaceNet, VGGFace 等模型，也借此理解了神经网络等架构的思想。

在理论入门、算法复刻、数据分析、数据处理、模型构建、模型改进等诸多过程之后，我的理论知识和实践能力也随之增强着。通过团队成员所有人的鼎力合作，积极沟通，我们最终以一个优秀的成绩完成了本次 kaggle 项目，实现了我们团队在学期初给自己定下的目标。

同时，老师在项目推进中反复提到的整体与局部概念也令我们受益匪浅，相信日后该项目的经历以及思想的深入一定会对我们的学习有着更多的帮助。

• 赵 **:

本次挑战课中，我接触了机器学习的相关知识，了解了人工智能背后的专业知识，学会了如何训练自己的神经网络模型。在项目进行的过程中，我主要负责论文的编写以及 PPT 的制作，这使我掌握了 \LaTeX 这门非所见即所得的文章排版语言的使用，使用 \LaTeX 编写生成的文章排版及其工整，令人赏心悦目。除此之外，在对模型进行集成学习训练时，我也深谙“三个臭皮匠赛过诸葛亮”的道理，使我明白了团队合作的重要性。

• 武 **:

参与这次项目对我来说帮助真的很大，不同于以往别的只重理论教学的课程，这次的人工智能课程给了我一次很好的实战机会，让我真切地感受到人工智能的应用。从完全没接触过人工智能，到结课时却能较熟练地搭建一个项目，这其中的成长是显而易见的。借助我们所选的这个血缘关系识别的项目，从 eda 开始，我们不断探索不同的模型与处理，从 facenet 到 resnet 再到两者的结合以及最后集成学习的使用，我不仅学会了如何运用 Facenet 和 ResNet 等模型进行人脸识别和血缘关系判断，还深刻领悟到了集成学习的力量，虽然计算成本翻倍提高，但所带来的效果确是十分强大的。在团队合作的过程中，我也学到了如何进行团队合作，明确各方需求和意见，让团队朝着共同的目标努力。我们每个人都充满了热情和专业精神，我们在挑战中共同成长，最终取得了令人满意的成绩。这段经历将成为我宝贵的回忆，感谢我的团队成员，感谢老师。

6 意见与建议

1. **课堂讲授内容由浅入深：**人工智能与机器学习毕竟存在着一定的技术门槛，尤其对于大二编程基础不够扎实的我们来说难度与挑战很大，希望老师能在开始的几节课上加入一些示例代码的讲解，带领我们快速入门。
2. **实验指导书内容详细化：**在做实验的过程中，我们发现实验指导书中的部分内容在理解上存在困难，导致实验进展缓慢，希望实验指导书中的实验步骤部分能够更加详细，便于实验的完成。
3. **课堂展示形式多样化：**在课堂上进行小组成果展示时，没有轮到的小组

组参与度不高，课堂互动率较低，希望可以增加课堂互动形式，提高同学们的参与度。

References

- [1] Florian Schroff, Dmitry Kalenichenko, James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pages 815–823, 2015.
- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pages 770–778, 2016.
- [3] Yaniv Taigman, Ming Yang, Marc’Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pages 1701–1708, 2014.
- [4] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.
- [5] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794, 2016.
- [6] Amit Dhurandhar, Akshay Gadi Patil, Prithwjit Guha, and Ganesh Ramakrishnan. DeepMEMNet: A deep learning framework for pan-specific MHC class I peptide binding prediction. *BMC Genomics*, 20(Suppl 2):165, 2019.
- [7] Zagoruyko, S., & Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition.
- [8] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition.

- [9] Koch, G., Zemel, R., & Salakhutdinov, R. (2015). Siamese Neural Networks for One-shot Image Recognition. In Proceedings of the 32nd International Conference on Machine Learning (ICML) (Vol. 37, pp. 1774-1782).
- [10] Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 1701-1708).