Udacity 毕业项目

Udacity Capstone Project

题目 Title: **Udacity 机器学习工程师毕业项目：**

图像分类之猫狗大战

学生姓名
Student Name: 王 成 伟

时间：二〇一九年九月十六日

Date: September 16th 2019

目录

第一章 问题的定义

1.1 项目概述

计算机视觉是数据科学与计算科学的研究领域，其研究对象主要是数字设备从环境中捕获的种种图像、视频，而计算机视觉的任务则是通过对这些图像数据进行分析以达到特定的视觉任务。近年来，结合深度学习算法的计算机视觉方法使得其图像分类领域有了长足的进展，而基于深度学习算法的图像识别和分类被很好地拓展到生物医学影像识别、无人驾驶等应用领域。在学界和商业的同时催化下，Kaggle 等数据科学竞赛平台中大量的图像分类以及识别竞赛中也有诸多优秀的深度学习算法应用，例如目前正在进行的细胞影像分类比赛，而由于我目前关注的是生物技术专业，因此我对这一类图像识别的能力也比较有兴趣。因此我本次选择的 Udacity 毕业项目 Dogs vs Cats 正是这诸多图像分类竞赛之一。

由于该项目来源于 Kaggle 中 Dogs vs. Cats 的比赛，在项目中使用的数据自然是该竞赛中提供的猫和狗的图像数据，而这个数据集可以通过 Dogs vs. Cats/data 这个链接获取。该数据集主要分为测试集 test.zip 以及训练集 train.zip 两部分，而这两个 zip 文件中各自包含了 125000 张猫或者狗的摄像。我们将对图像数据进行初步的探索，并在训练之前把训练集将被分为训练部分以及验证部分，完成后续的深度学习训练之后我们再使用测试集的数据进行测试，从而验证我们的模型的有效性。在这个过程中，我们还通过 ImageDataGenerator 对图像数据进行了数据增强。

1.2 问题重述

2016 年 9 月，Kaggle 举行过一场有趣比赛：Dogs vs. Cats，这场比赛的竞赛任务主要是使用计算机对猫和狗的图像进行分类。而 3 年后的今天，机器学习领域发生了很大的变化，特别是在深度学习和图像分析方面。因此本次我将再次挑战这一个图像分类问题：输入一张包含猫或者狗的图像，然后通过计算机算法判断图片中的是猫还是狗。简而言之，这是一个关于猫和狗图像的二分类问题。在这个竞赛中，比赛提供了排名所用的评判标准，这个数学化的评估标准使得这个问题有了定量的依据而我们将会在下一个 section 中进行介绍。

为了能够快速使用和部署深度神经网络来完成项目，作者将使用迁移学习技术以及调整 Dropout 和不同层的参数等方式完成项目，具体内容可以参考后面的算法和技术

部分以及第三章的实现部分。

关于项目目标，根据 Udacity Capstone Project 中提出的要求，最后的模型效果应当能够在 Dogs vs. Cats 竞赛的 Public Leaderboard 中到达前 10% 的排名，由于参与的队伍一共有 1314 支，因此最终模型的分数需要优于其中的第 131 名，LogLoss 值需要低于 0.06127，而 LogLoss 值则是在下一段中进行介绍的 Kaggle 提供的评估标准。

1.3 评估指标

LogLoss，即对数损失函数是用于评估模型预测结果与实际的相似性而设计的一种损失函数，而对于一个模型来说，损失函数的数值越小，则说明这个模型的估计结果与实际情况更为接近，因此可以认为这个模型的效果要更好，因此在上一段中项目目标是要使得 LogLoss 低于第 131 名的数值。本次比赛的评估方法页面也提供了 LogLoss 的计算公式，具体如下：

$$LogLoss = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)]$$

其中，n 是测试集中的图像数目，$\hat{y}_i$ 是对应图像被分类为狗的图像的概率，y_i 则表示对应图像通过赋值为 0 或 1 分别表示图像是猫或狗，log() 则是自然对数。

我们需要使用训练好的模型对测试集中的图像进行测试并记录算法判断图像为狗的概率，生成 Kaggle 规定 title 的 csv 文件并提交到 Kaggle 网站中进行评分。如果评分的结果实现了上述的项目目标的话，这个项目的目标也就达成了。

第二章 分析

2.1 数据的探索

首先我们将对图像数据的情况进行初步的探索。该数据集主要分为测试集 test.zip 以及训练集 train.zip 两部分，而这两个 zip 文件中各自包含了 125000 张猫或者狗的摄像。同时我们注意到了图像中的猫和狗主要是指生物意义上的猫和狗，对于卡通图像以及图标等情况应该是不予考虑的。这两个数据集的区别主要在于是否有对图像进行标记：在测试集中图像的命名为 number.jpg，是没有进行标记的图像；而在训练集中图像的命名模式则为 category.number.jpg，在名称中的 category 可以为 cat 或 dog，从而在命名中为图像添加了一定的标记。以下是一些数据的样本图像：

(a) dog.4745 (b) dog.4746

(c) cat.0 (d) cat.1

图 2-1 **Kaggle 数据样例**

而在这个过程中，我们也发现了部分训练集中的样本图片是不太符合要求的。但是

```
In [2]: !echo "cat figures: $(find train/ -name 'cat*jpg' | wc -l)"
        !echo "dog figures: $(find train/ -name 'dog*jpg' | wc -l)"
        !echo "test figures: $(find test/ -name '*.jpg' | wc -l)"

cat figures: 12500
dog figures: 12500
test figures: 12500
```

图 2-2 **Kaggle 数据集数据量**

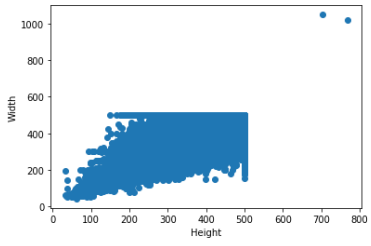
由于这些图片的数量相对于整体的数据量来说几乎是微不足道的，因此在这个认为更好的处理方式是将其作为训练中的噪音，以此检验训练模型的鲁棒性。

2.2 数据可视化

在进行了相对直观的观察之后，我们可以发现数据中图像的尺寸似乎是不一致的。但是对于深度学习模型来说，输入图像的尺寸应该是统一的。因此在对图像进行处理之前，我们首先需要对图像的整体尺寸进行了了解：

```
In [4]: # Scatter
height = [i[0] for i in shapes]
width = [i[1] for i in shapes]

plt.scatter(height, width)
plt.xlabel('Height')
plt.ylabel('Width')
plt.show()
```



从上图我们可以发现,Kaggle提供的图像大小并非完全相同的，因此我们在后续对图像的尺寸进行规范。

图 2-3 **Kaggle 数据集图像尺寸大小分布**

通过统计我们进一步确认了数据集中图像尺寸不一的情况，因此我们需要通过 ImageDataGenerator 对图像尺寸进行变换。虽然通过填充或者缩放会使得图像的细节有一定程度的丢失，但是由于本次项目的目标是通过对图像中的狗和猫等与背景环境区分较为明显的特征/生物进行识别，因此可以认为这种程度的变换对于深度学习网络来说应该不会由太大的影响；虽然有一些例外的情况，但是在图像中我们可以发现在 0-500 的长宽区间内是图像的主要分布区域，同时呈现了从 0 到 500 图像点逐渐增多，分布范围逐渐变宽的情况，因此在综合下面选择的模型的输入要求以及分布情况，统一调整后

的图像大小应该是 299×299，而通过 ImageDataGenerator 对图像进行处理之后，图像数据可以视为一个 (299, 299, 3) 的三维数组了。

同时在对数据进行浏览的过程中，我们发现了一些异常的图像，但相较于总体的图像数据量而言占比极小。一方面对于这些图像使用预训练模型进行筛查的话可能会对不少正确的图像进行误删，另一方面这些少数异常的图像噪音也可以作为模型稳定性的依据，由于目前时间仅剩 5 天左右的时间，在项目目标达成的前提下我们暂时不对图像中的异常图像进行进一步的挑除，还请各位谅解啦。

2.3 算法和技术

2.3.1 卷积神经网络基础

本次计算机视觉项目主要的基础是深度学习。而在项目中使用到的深度学习模型都离不开卷积神经网络 (CNN) 等基础知识。因此在介绍后面具体的模型和算法之前，在这一部分我们将对卷积神经网络进行简要的介绍。

卷积神经网络是一种常见于分析视觉图像和语音识别领域的，带有卷积结构的前馈深度神经网络，具有参数相对较少同时训练效果较好的优点。经典的 CNN 由输入层、卷积层、池化层、全连接层及输出层组成。

卷积层是具有自动提取输入信号的深层信息功能的隐藏层，每个特征面由多个神经元组成，它的每一个神经元通过卷积核与上一层特征面的局部区域相连。上一层的特征图被一个可学习的卷积核进行卷积，然后通过一个激活函数，就可以得到输出特征图。每个输出特征图可以组合卷积多个特征图的值。

池化层是的功能是对特征图的采样处理减少模型中的参数量的同时尽可能保留核心的特征。它通过每个神经元根据设定好的区域进行局部采样并将其统合为一个维度较低的输出，使 CNN 具有更好的适应性以及更少的模型参数/维度。

全连接层一般位于网络尾端，具有对前面逐层变换和映射提取的特征进行回归分类等处理的功能。它会将高维图像的特征图拼接为一维特征作为全连接网络的输入。全连接层的输出可通过对输入加权求和并通过激活函数的响应得到。

2.3.2 优化器部分

本次构建模型采用的优化器主要是 ADADELTA，它是一种自适应学习率的梯度下降方法，这意味着我们不需要过多地对学习率进行探索。同时 ADADELTA 通过在有限的时间窗内累积平方梯度的总和并纠正大多数基于梯度下降的算法中单位不匹配导致

的权重不匹配等方式使得梯度下降的更新方向尽可能地保持负梯度，并确保在完成大量迭代之后学习仍能继续进行。这些特性使得 ADADELTA 在计算机视觉领域的 MNIST 数字分类问题中有优秀的表现，因此在本次图像分类项目中我们也决定使用这一个优化器进行模型构建与优化。

2.3.3 深度学习技术

本项目中最主要的深度学习技术是迁移学习技术。迁移学习是指将经过训练和优化的模型参数迁移到新的模型来帮助新模型训练，基于数据任务之间的相关性程度加快并优化模型的学习效率的一种方法。而在计算机视觉的任务中，迁移学习是拓展模型识别和应用范围的优秀解决方案：在 Udacity 的课程介绍中，我们甚至可以看到迁移学习是如何将将识别不同类别物体的图像的模型应用于癌症的识别与分类中的，并且意外地具有相当不错的分类效果，而该研究也因此发表在了 Nature 杂志上。鉴于迁移学习能力在使用优化后的模型上的优越性和可拓展性以及这两次任务之间的相似性，本项目主要是通过迁移学习构建 Xception 参数调整后的对应模型来解决对猫和狗的图像进行二分类的问题。

2.3.4 模型部分

在上一个部分种我们选用的 Xception 模型是由 Inception 作为起源，并使用深度可分离卷积技术使得其参数效能更为优秀的计算机视觉 ImageNet 领域的深度学习模型，而其在 ImageNet 数据集中的能力要优于前代的 Inception 模型。因此我认为使用这个模型作为迁移学习的开端并使得其具有识别猫和狗的共功能应该是可行的。Xception 的结构如图2-4所示，其中可分离卷积核 (SeparableConv) 通过对每个通道 (channel) 进行独立卷积操作的特点使其在参数量减少的同时，提高了泛化性能。由于 Xception 的这些优良特性，相较于 ResNet 和 Inception，本次项目更倾向于使用 Xception 来更好地完成迁移学习的任务。

而由于后续基准模型中提及的原因，本次项目对 ResNet 部分的使用已经暂停了，所以如果对 ResNet 以及 ResNext 模型感兴趣的话可能需要对通过阅读原始文献进一步了解啦。

Figure 5. The Xception architecture: the data first goes through the entry flow, then through the middle flow which is repeated eight times, and finally through the exit flow. Note that all Convolution and SeparableConvolution layers are followed by batch normalization [7] (not included in the diagram). All SeparableConvolution layers use a depth multiplier of 1 (no depth expansion).

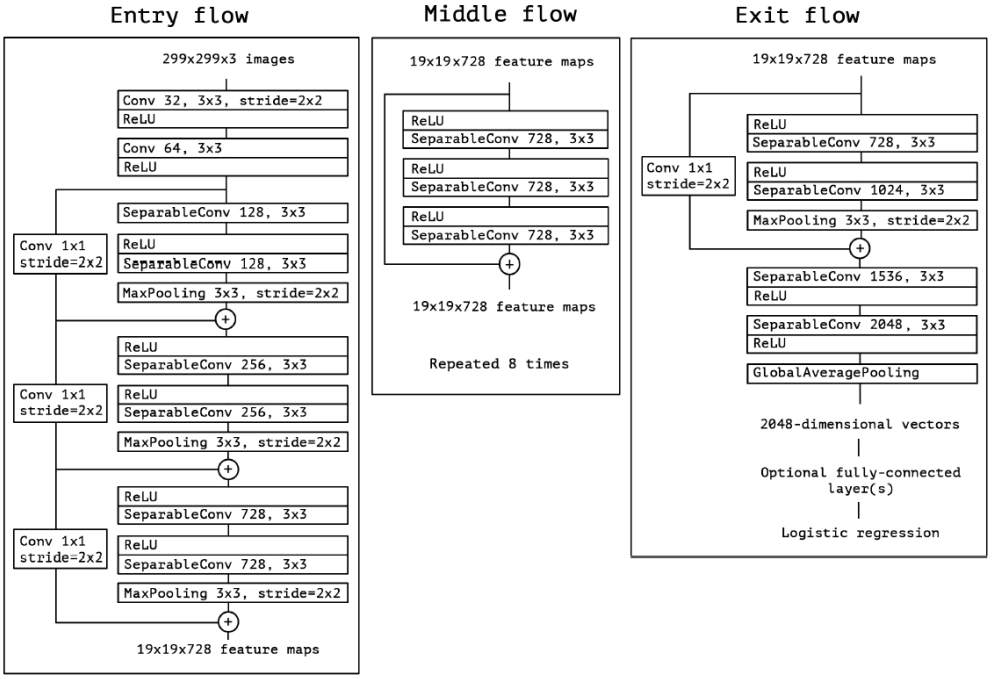


图 2-4 **Xception 模型架构**

2.3.5 模型优化方法

而在 Xception 的基础之上，我们使用了 Dropout 层以及调参的方法进一步地提供这一图像识别通用模型对于特定的猫狗图像数据识别上的准确率。Dropout 是指在深度学习网络的训练过程中，对于神经网络单元，按照一定的概率将其暂时从网络中丢弃，然后在下一次训练中再以随机的概率重新对神经元进行启动，通过 Dropout 的技巧我们可以使得神经网络的过拟合问题得到缓解，让模型的训练效果更加理想。我们将在 Xception 后添加 Dropout 层并通过对 Dropout Rate 进行探索，找到最为适合的 Dropout Rate。然后为了使得深度学习网络能够对猫和狗的图像分类和特征识别具有更有针对性的权重分配，我们需要在 Xception 原有的权值基础上开放一部分神经网络一部分的卷积层以供训练数据对其进行调整，本次项目中我们在相对有限的时间内根据 Xception 模型的层级结构，仅仅对 97 层以及 107 层之后的参数进行了相应的调整测试。由于最终的效果可以到达 Kaggle 猫狗大战竞赛的前 10 名，在完成项目目标的前提下，我们没有继续对其他调试方式进行进一步的测试，而这一点也会在今后继续学习和改进。

2.4 基准模型

本次项目选择的原基准模型是在图像识别中有优良性能的 ResNet50。ResNet50 是 2015 年由 Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun 等人提出，并在图像分类领域具有里程碑意义的模型。ResNet50 在 ImageNet 比赛的分类任务中是曾获得第一名的深度残差网络模型，鉴于 Dogs vs Cats 比赛的问题同样是图像分类问题，因此以 ResNet50 作为分类能力的基准应该是可行的。

作为与 Xception 进行比较的模型，我们将使用导入权重后 ResNet50 在猫狗大战项目中的 LogLoss 得分与 Xception 的得分进行比较，以此作为一个基准参照。同时我们也对 Xception 进行自身对照，通过比对进行 Dropout 与调参后的 Xception 模型的 LogLoss 得分我们进一步了解调整的效果是否是有成效的。

然后在参考了审阅老师的意见之后，我发现 ResNet50 由于分数表现上并没有达到 Kaggle 中 Top 10% 的水平，因此在基准模型方面我们目前参照 Xception 原模型在 Kaggle 上的评分结果，即在第三章中将会出现的 Xception-original。它在 Kaggle 中 LogLoss 的分数是 0.04180，低于了 Kaggle Top 10% 的基准，即 LogLoss = 0.06127，因此 Xception-original 作为基准模型应该大体上还是符合要求的。我们需要在 Xception 模型的基础之上进一步地通过前述的模型优化方法进行改进，使得模型的得分到达 Kaggle 中的 Top 10。而 ResNet50 模型则可以在这里作为一个额外的参考。

第三章 方法

3.1 数据预处理

在进行图像尺寸调整和归一化之前，我发现如果要使用 ImageDataGenerator 必须要将训练集中的图像进行分类，放置进两个文件夹中进行整理。然而对 1250 张图片进行复制对于 Ubuntu 系统的图形化文件系统来说是一个较为尴尬的事情，因此在对数据进行尺寸缩放和数据增强之前，我们首先使用 Symbol link 对图像进行索引。

Symbol link 根据图像中 cat 和 dog 的前缀使得训练集数据能够划分到两个命名为 cat 和 dog 文件夹中。这一部分如图3-1中 In [2] 和 In [3] 所示。

```
In [2]: train_filenames = os.listdir('train')
        train_cat = filter(lambda x:x[:3] == 'cat', train_filenames)
        train_dog = filter(lambda x:x[:3] == 'dog', train_filenames)

        set_mkdir('trainAlter')
        os.mkdir('trainAlter/cat')
        os.mkdir('trainAlter/dog')

        set_mkdir('testAlter')
        os.symlink('../test/', 'testAlter/test')

        for filename in train_cat:
            os.symlink('../../train/'+filename, 'trainAlter/cat/'+filename)

        for filename in train_dog:
            os.symlink('../../train/'+filename, 'trainAlter/dog/'+filename)

In [3]: ## Generate symbol-link
        train_data_dir, valid_data_dir, test_data_dir = symbol_link()

        100%|████████████| 25000/25000 [00:00<00:00, 56821.13it/s]
        100%|████████████| 12500/12500 [00:00<00:00, 66873.13it/s]

In [4]: write_data(Xception, (299, 299), batch_size=16, dir_train="trainAlter", dir_test="testAlter", lambda_func=xception.preprocess_input)

        WARNING:tensorflow:From /home/ninomoriaty/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:63: The name tf.get_defa
        ult_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

        WARNING:tensorflow:From /home/ninomoriaty/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:492: The name tf.placeho
        lder is deprecated. Please use tf.compat.v1.placeholder instead.

        WARNING:tensorflow:From /home/ninomoriaty/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:3630: The name tf.random
        _uniform is deprecated. Please use tf.random.uniform instead.

        WARNING:tensorflow:From /home/ninomoriaty/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:3458: The name tf.nn.max
        _pool is deprecated. Please use tf.nn.max_pool2d instead.

        WARNING:tensorflow:From /home/ninomoriaty/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:158: The name tf.get_def
        ault_session is deprecated. Please use tf.compat.v1.get_default_session instead.

        WARNING:tensorflow:From /home/ninomoriaty/anaconda3/lib/python3.7/site-packages/keras/backend/tensorflow_backend.py:163: The name tf.ConfigP
        roto is deprecated. Please use tf.compat.v1.ConfigProto instead.

        Found 25000 images belonging to 2 classes.
        Found 12500 images belonging to 1 classes.
        1563/1563 [==============================] - 282s 180ms/step
        782/782 [==============================] - 139s 178ms/step

In [4]: X_train, Y_train, X_test = export_data("data_xception.h5")
```

图 3-1 **数据预处理部分的操作**

最终我们通过这个步骤生成对应的训练集和测试集，分别存储在 testAlter 和 trainAlter 文件夹中。进而我们还需要对训练集中的数据进一步的划分：将 Kaggle 中原本的训练集以 8:2 的比例划分为训练集和验证集两个部分。

完成上述操作之后，我们需要对训练数据的图像尺寸进行调整。在这个过程中我们主要使用到的是 Keras 提供的 ImageDataGenerator 方法。我们可以通过 write_data 方

程调用 ImageDataGenerator 对据进行尺寸缩放以及数据增强，其中数据增强部分的操作包括了 rotation_range，width_shift_range, height_shift_range, shear_range, zoom_range, horizontal_flip 等方面的旋转、平移、拉伸、缩放、反转调整，应该可以更好地增强模型的适应性。

然后我们需要通过 ImageDataGenerator 的 flow_from_directory 方法以文件夹路径为参数, 生成经过数据提升/归一化后的图像数据。由于 RGB 颜色的取值范围是 [0,255], 而图像尺寸的取值范围大致分布在 [0,500]，以位置信息和颜色信息描述像素的图像数据如果直接输入，不仅会有数据度量上的差异，还会由于数值过大和度量差异影响神经网络的收敛效果。所以我们需要使用 flow_from_directory 将图像的取值范围从 [0,255] 转换为 [0,1]，从而加快网络的学习速度。

最终形成的数据集划分如图3-2所示：

```
udacity_dog_vs_cat/
├── data
│   ├── test
│   ├── train
│   └── valid
├── __pycache__
├── test
├── testAlter
│   └── test -> ../test/
├── train
├── trainAlter
│   ├── cat
│   └── dog
└── train_refined
    └── train
```

图 3-2 **数据划分情况**

3.2 执行过程

3.2.1 GPU 运行相关设定

在进行后续模型训练之前，首先需要对模型中使用的 GPU 进行设定。由于作者的显卡显存较小, 在使用的过程中出现了两种 GPU 相关的报错：CUDNN STATUS INTERNAL ERROR 以及 Resource Exhausted Error。作者主要的解决方式是通过设定 GPU 的使用率 (如图3-3) 以及对 batch_size 进行限制来解决对应的问题的。同时在运行过程中建议不双开 Jupyter notebook 以避免在运行过程中突然产生 GPU_Sync_Failed 的问题，从而避免训练结果丢失等问题。

```
## Specific setting for CUDNN_STATUS_INTERNAL_ERROR
#config = tf.ConfigProto(gpu_options=tf.GPUOptions(allow_growth=True))
config = tf.ConfigProto(gpu_options=tf.GPUOptions(allow_growth=0.95))
sess = tf.Session(config=config)
```

图 3-3 **GPU 设定**

3.2.2 Xception 原模型的构建

在完成了数据的预处理之后，我们首先需要对 Xception 模型原本的能力进行检验，因此我们首先先对 Xception 本身进行猫狗图像识别的训练和测试，并将结果提交到 Kaggle 中进行验证。具体过程如图**??**所示。

```
In [6]: ## Construct the model
input_tensor = Input(X_train.shape[1:])
x = Dropout(0.5)(input_tensor)
x = Dense(1, activation='sigmoid')(x)
model = Model(input_tensor, x)

model.compile(optimizer='adadelta',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
In [7]: ## Training
filepath="xception-tune0-best_weight.h5"
checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min',save_weights_only=True)
callbacks_list = [checkpoint]
model.fit(X_train, Y_train, batch_size=128, epochs=20, validation_split=0.2, shuffle=True,
          callbacks=callbacks_list)
```

```
In [11]: ## Testing
predict_on_model(test_data_dir, X_test, model, "pred-xception-original.csv")
```

图 3-4 **Xception-original 模型构建和结果导出**

通过 Kaggle 的验证，我们发现 Xception 的原始模型 Xception-original 的得分为 0.04180。而这离 Kaggle 大赛的前十名还有一定的距离，因此我们需要对 Xception 模型进行进一步的调整以提升模型针对猫狗图像数据的分类能力。

3.3 完善过程

3.3.1 Dropout Rate 调整

我们首先对引入的 Dropout 层进行调整，通过探索使用不同的 Dropout Rate 对应的 Xception 模型 LogLoss 的得分变化，我们可以从中选出更为合适的 Dropout Rate，从而可以更好地避免过拟合情况并提升模型的分类能力。我们对 Dropout 值为 0.1, 0.2, 0.3,

0.4, 0.5, 0.6, 0.7, 0.8, 0.9 的情况进行了探索并将结果进行了可视化，结果如图3-5以及图3-6所示。

	Dropout	Avg	Max	Min
0	0.1	0.011932	0.012197	0.011845
1	0.2	0.011889	0.012069	0.011770
2	0.3	0.011922	0.012120	0.011806
3	0.4	0.011952	0.012200	0.011790
4	0.5	0.011956	0.012080	0.011854
5	0.6	0.011907	0.012183	0.011750
6	0.7	0.012001	0.012144	0.011861
7	0.8	0.012288	0.012565	0.012081
8	0.9	0.013365	0.013602	0.013175

图 3-5 **Xception-dropout Dropout Rate 调整的数值结果**

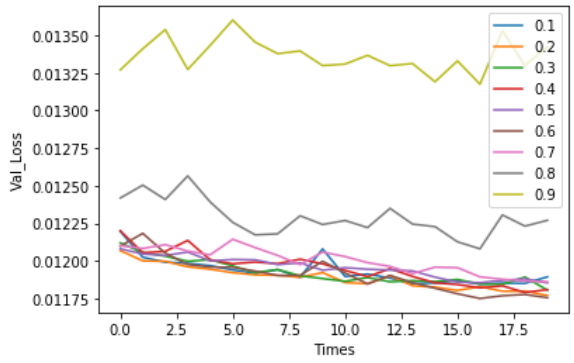


图 3-6 **Xception-dropout Dropout Rate 调整的可视化结果**

从图中的表格以及可视化的结果中可以看出，Dropout Rate 等于 0.2 的时候结果平均值和最大值是最佳的，而其最小值也是倒数第二小的。因此可以认为 Dropout 值为 0.2 时效果最佳。同时, 通过后续在 Kaggle 中对使用不同 Dropout Rate 的 Xception-dropout 模型的预测结果进行测试后，我们可以从结果对比的表格3.1中发现中 LogLoss 在 Dropout=0.2 时的结果最优, 比之前 Xception-original 的得分要低 0.001。因此我们在后续的实验过程中使用的 Dropout 层的 Dropout Rate 将设定为 0.2。

3.3.2 调参过程

在经过了对 Dropout 值的调整后，我们发现虽然模型的表现更好了，但是提升程度并不能到达前十名的范畴。而这可能是由于 Xception 模型本身的权重并没有根据猫和

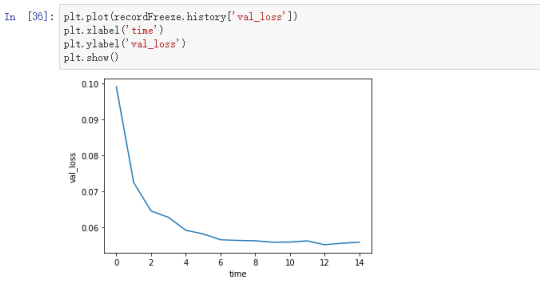
Dropout Rate	LogLoss(Kaggle)
0.2	0.04057
0.4	0.04145
0.5	0.04180
0.6	0.04196

表 3.1 Dropout Rate 的 LogLoss 得分表格

狗的图像特征数据进行特化导致的。因此，在下面的步骤中我们需要对 Xception 模型不同层的权重进行调整，而由于时间有限，在完成项目目标的前提下，根据 Xception 模型的结构我们主要对全连接层，97 层以上，107 层以上的 sepconv 进行调参。

首先我们先尝试对全连接层进行调参操作，在调参过程中使用的参数是 steps_per_epoch=625, epochs=15，而在训练到第 14 个 epoch 的时候 val_loss 的数值没有出现明显的下降了，因此使用了 15 作为 epoch 的数目。在完成训练之后，我们通过以 epoch 为 x 轴，val_loss 为 y 轴的方式对模型 LogLoss 的变化情况进行可视化 (如图3-7所示)：

Xception-freeze 全连接层调参



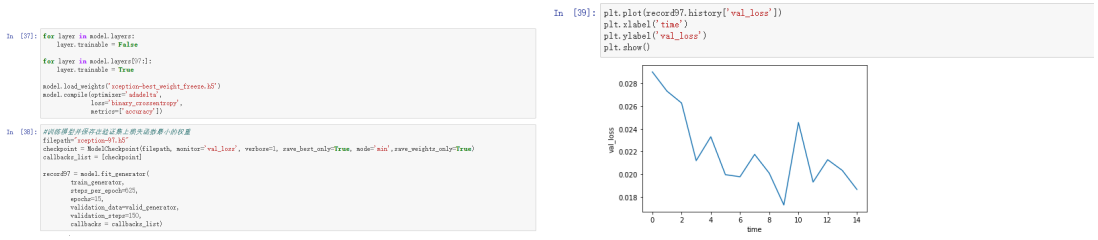
Xception-freeze 全连接层调参 LogLoss 结果

图 3-7 **Xception-freeze 全连接层调参结果**

我们可以发现 LogLoss 的数值的下降很快，并且能够到达一个不错的水准。然后我们将预测结果的 csv 文件提交到 Kaggle 中进行评分，Xception-freeze 本次的得分为:0.06179。从这一个阶段的结果可以看出训练全连接层对于结果的改变是较为显著的，但是如果仅仅是对全连接层进行训练的话效果反而会变得比较差。

因此在保存了对全连接层进行训练的权重之后，我们需要对 97 层以上的权重进行调节 (如图3-8所示)，这一次调参使用的参数是 steps_per_epoch=625, epochs=15，然而其实在 epoch=10 的时候 val_loss 的情况已经没有变化了，因此其实在训练的过程中或许使用 10 作为 epoch 的数目会更加节省时间成本，然而为了进一步探索可能存在的更有的参数，这里还是秉持探索优先的原则保持了 epochs=15 的设置：

我们可以发现 LogLoss 的数值的虽然波动，但是它的最高值很低，同时其最低值相当优秀的，并且整体的趋势依然是下降的，这是一个好现象。然后我们将预测结果的



Xception-97 97 层以上的权重调参 **Xception-97 97 层以上的权重 LogLoss 结果**

图 3-8 **Xception-97 97 层以上的权重调参结果**

csv 文件提交到 Kaggle 中进行评分，Xception-97 本次的得分为:0.03709。在 Kaggle 的排名中已经能够进入前 10 名了，因此这是一个较为成功的模型，而这也说明对 Xception 中不同层的权重进行针对特定目标的调整是有很显著的作用的。

最后我们处于保持结果相对全面的考虑对 107 层以上的权重进行调节 (如图3-9所示)：

Xception-107 107 层以上的权重调参 **Xception-107 107 层以上的权重 LogLoss 结果**

图 3-9 **Xception-107 107 层以上的权重调参结果**

我们可以发现 LogLoss 的数值波动较大并且整体的趋势呈现 U 字形，这是可能意味着模型出现了一定程度的过拟合。然后我们将预测结果的 csv 文件提交到 Kaggle 中进行评分，Xception-107 本次的得分为:0.04183。从结果来看还是比全连接层的要更优，说明调参的效果是存在的，但是还有待改进。为了比对这三种方法的效果差异，原则上来说这三者的参数设置应该是需要保持一致的，因此在这里我们对于参数的设置不再进行赘述，但正是因为这一点，其实为什么不同程度地对层次进行调节会出现这样的差异？是否有办法让神经网络判断何种调节方式是最优的？这些问题都是值得在调参过程中进一步思考和挖掘的问题，而这些问题也有待计算科学工作者进一步探究深度学习的黑箱奥秘啦。

综上所述，我们将选取 Xception-97 作为最终的模型，它在 Kaggle 中的得分为:0.03709，

排名为第 7 名，符合我们项目目标中到达 Kaggle LeaderBoard 前 10% 的预期。

第四章 结论

4.1 模型的评估与验证

本次项目中 Dropout=0.2 的模型评估结果如图4-1所示：

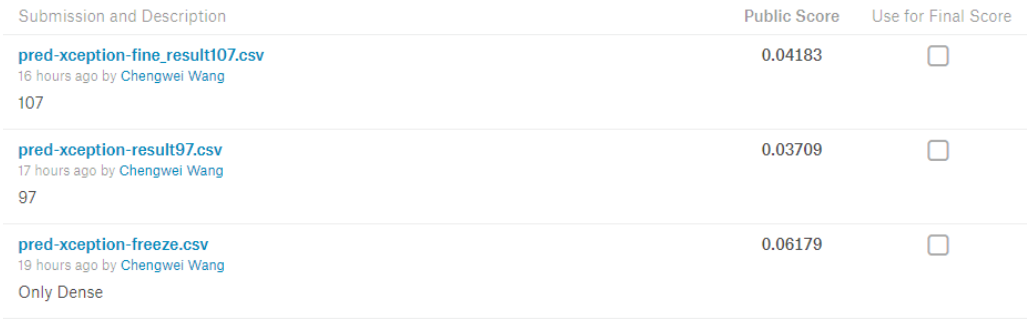


图 4-1 **Xception 优化后的结果列表**

本项目的最终模型如前所述，是基于 Dropout 调节和调参技术改进之后的 Xception 模型，相比于基准模型和 Xception 原模型，其在测试集上的得分有明显的提升。该模型最终在测试集上的得分为 0.03709，该得分符合项目目标预期并且达到 Kaggle LeaderBoard 前 0.6% 的排名水平。另一方面，虽然训练数据中有未剔除的异常图片，但是该模型具有非常强的预测性能，所以在达成稳定性要求的前提下应该可以暂时忽略这一部分图像的影响。

4.2 合理性分析

首先从项目的目标出发，最终我们得到的模型结果在 Kaggle 验证下的得分是 0.03709，在 Kaggle LeaderBoard 中的排名为第 7 名。从某种意义上说，该模型通过了其评估标准提供的检验并达成了项目目标，因此可以认为这个模型在项目的实现上的可靠的。

同时，我们也将该模型的结果与基准模型 ResNet59 进行比较，在 Kaggle 中 ResNet50 的得分为 0.07774，略低于当前的模型得分；同时，经过调整的模型的 LogLoss 值也比 Xception 原本的权重的分值低，因此可以认为在模型的完善过程中执行的操作的有效性。

然后从实际结果来看，我们抽取了部分结果，查看其对应的分类以及概率，可以发现模型对于对应结果的预测还是符合实际图片的实际请况的。因此可以认为项目大体上的是符合预期的，而其结果也是相对合理的。

第五章 项目结论

5.1 结果可视化

本次项目主要是通过迁移学习技术对 Xception 模型进行针对猫狗图像的识别优化，最终实现了一个能够在 Kaggle 中排名，LogLoss 得分为的图像分类深度学习模型。而这个项目的结构可以简要地参考下图：

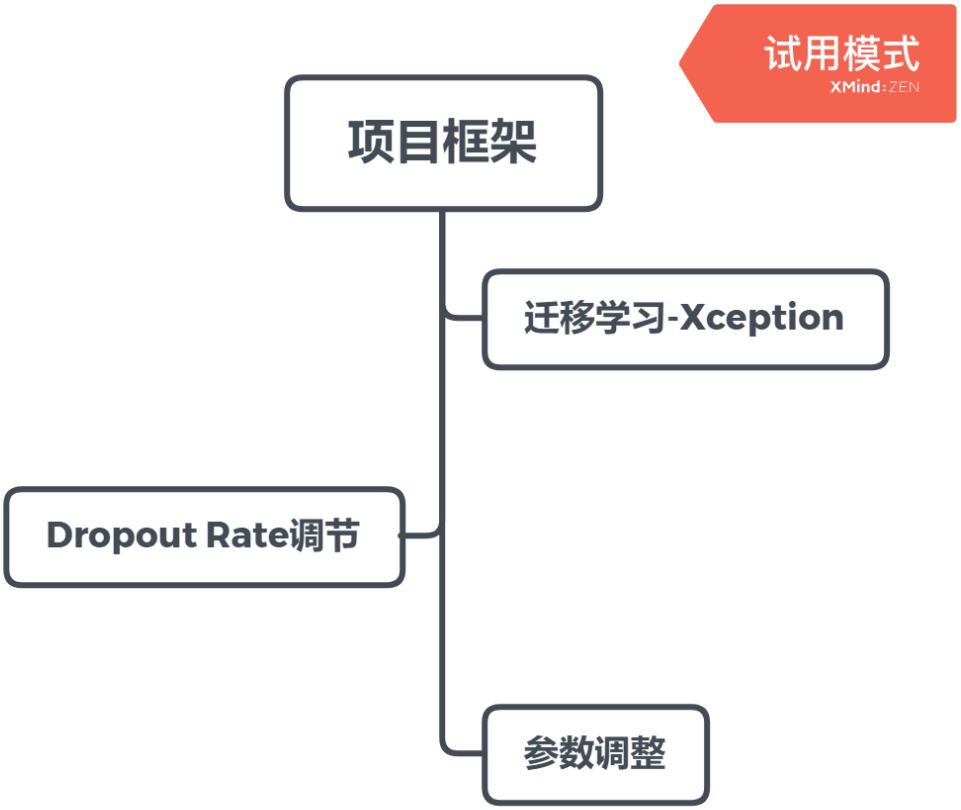


图 5-1 **项目结构可视化**

在这个项目中主要使用到的技术是迁移学习技术以及 Dropout 和不同层的参数微调，在项目的执行过程中，主要是先通过迁移学习将 Xception 模型的所有参数尝试应用到猫和狗的图像分类中，通过添加 Dropout 层和全连接层并对其中的参数进行调节，并其后通过对 Xception 中较为靠近全连接层的层的权重和参数进行调整，最终完成了完整猫狗图片分类模型。

5.2 对项目的思考

上述的解决方案以及项目结构的展示应该已经较好地对项目的框架进行了对应的总结。而在执行项目的过程中我也学习、意识到了不少新的场景。通过这一次的项目，最直接的收获就是对迁移学习的背景以及应用方法有了更加深入的了解，而这种方式由于其转化效率的优异，可以使得一个模型的应用场景不再局限于原本训练集的限制而可以在更多的领域中发挥重要的作用，而这种机制与神经科学中的学习迁移也有异曲同工之妙，可以说是值得我们对其进行进一步学习和探讨的领域。

由于我并不是科班出身的计算机研究人员，在实战的过程中我遇到过许多困难。从最开始让人尴尬的 GPU 运行环境配置到后面使用迁移学习以及对应的可视化工作，这些知识的逐渐积累和时间上的紧急都让人比较焦躁。而在这样紧急的情况下，通过不同的技术对模型进行优化，最终初步实现项目目标的过程依然是充满挑战和趣味的。

虽然项目的初步目标基本完成，但是迁移学习在图像识别领域的应用是让人感到着迷的，这一种图像技术是否可以应用于细胞分形、是否可以在三维的条件下对结构生物学的研究作出更大的推进，这是我在未来会进一步思考和关注的研究主题。

5.3 需要作出的改进

虽然这一次的项目最终能够符合预期，但是由于时间紧迫，这个项目有待改进的部分还有很多。在数据处理部分，其实尝试补充更多的来源于其他分类比赛中的数据对训练集的体积进行扩充，同时如果可能的话还可以对其中猫和狗的品种进行进一步的划分，从而使得这个项目成为一个猫科动物和犬科动物的分类器。优化算法和权重调整方面也可以进行更多的尝试来提升预测的效果。然而由于目前时间实在是仓促，因而这些部分都不能继续进行完善了，而在之后如果在实验室中有相应的项目的话，我将会尝试重新对这些内容进行进一步的完善和升级。

以上就是本次 Udacity Capstone Project 的项目报告，作者水平有限，如有问题，还请各位不吝赐教啦。

参考文献

[1] Dogs vs. Cats Redux: Kernels Edition.

[2] Recursion Cellular Image Classification.

[3] Transfer learning, September 2019. Page Version ID: 915470891.

[4] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv:1610.02357 [cs]*, October 2016. arXiv: 1610.02357.

[5] Andre Esteva, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, February 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. arXiv: 1512.03385.

[7] S. Sarkar and K. L. Boyer. Perceptual organization in computer vision: a review and a proposal for a classificatory structure. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):382–399, March 1993.

[8] Vladimir Vovk. The Fundamental Nature of the Log Loss Function. In Lev D. Beklemishev, Andreas Blass, Nachum Dershowitz, Bernd Finkbeiner, and Wolfram Schulte, editors, *Fields of Logic and Computation II: Essays Dedicated to Yuri Gurevich on the Occasion of His 75th Birthday*, Lecture Notes in Computer Science, pages 307–318. Springer International Publishing, Cham, 2015.