机器学习工程师纳米学位毕业项目

猫狗大战

陈威洋

2018 年 1 月 10 日

目　录

1.问题的定义

1.1 项目概述

首先介绍我使用的卷积神经网络（简称：CNN），它是图像传递给一系列卷积，非线性，汇聚（下采样）和完全连接的图层，并获得输出。它的灵感来源于生物学的人类视觉皮层，在具有特定任务的系统内部（视觉皮层中寻找特定特征的神经元细胞）的专门组件的想法是机器使用的，并且是 CNN 背后的基础。CNN 学习了每个卷积层上的过滤器，这些过滤器担当越来越抽象的特征检测器，这里过滤器其实是一种权重共享的思想。

1989 年 Yann LeCun 提出的一个深度学习的新模型，简称为 CNN，当时能够对字符进行分类，准确率达到 99%，打破了以前所有记录，但 CNN 是自行学习特征，在 2012 年，另外一名研究者 Alex Krizhevsky 在一年一度的 ImageNet 比赛上使用了它，准确率达到新高 85%，自此之后被 Google 用于识别搜索图片，Facebook 用它来自动添加标签等等，CNN 还是第一个解决重要商业应用的神经网络，并且仍然是当今深度学习商业应用的前沿。

此次我选择卷积神经网络中一个 VGG 神经网络，简称 VGGNet，来解决猫狗大战的分类问题。

1.2 问题陈述

项目选择的数据集是 Kaggle 竞赛提供的数据，训练集包括 12500 张被标记为猫的图片和 12500 张被标记为狗的图片，测试集包括 12500 张未标记图片。由于是现实世界的图片，每张图片的大小不同，而且拥有对应的背景，其中狗的背景还带有猫或人，这些图片不同的特征都会增加模型识别的难度。

猫狗大战是个二分类问题，可以用很多类型的卷积网络来解决问题，你可以重新建立一个的模型，但效率很低，为了提高效率，我决定使用权重迁移，它的主要作用是使用有经验的模型，加以修改交给数据训练后成为新的模型。

1.3 评价指标

Kaggle 的猫狗大赛使用的评价指标为对数损失，计算方法如下：

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

这个指标是一个输出的是连续值，以致于能够对模型进行更细致的评价，用这一指标是合理的。模型不但需要能够分类正确，而且需要分类十分肯定。

虽然正确度的输出数据相对于 LogLoss 来说是离散的，但这种离散不会大起大落，在 2013 年举行的第一次猫狗大赛中，使用的分类评价指标就是正确度。它目前是使用普遍，而且较为直观，所以本次项目使用准确度。

2.分析

2.1 数据可视化

竞赛数据集有一个训练数据集和一个测试数据集，训练数据集中图片的文件名为标签，说明了图片类别是猫还是狗。

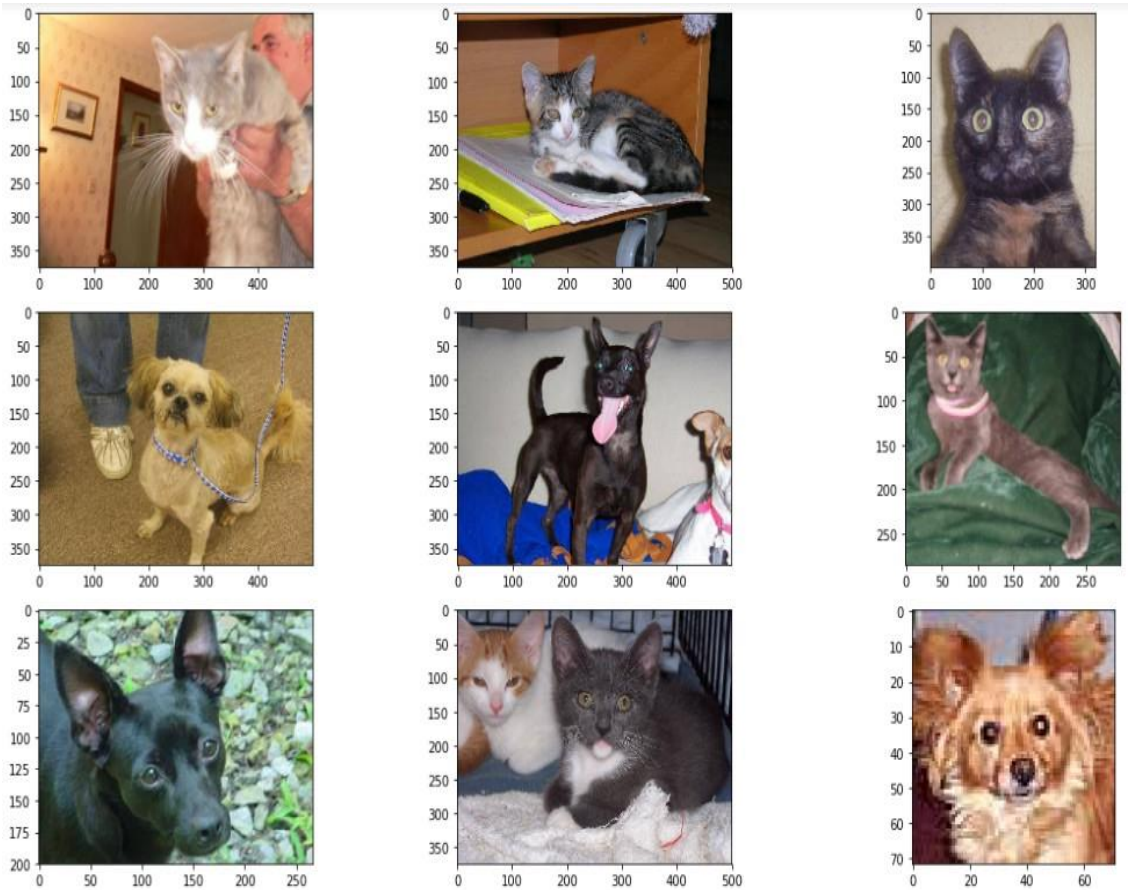


图 1 训练集图片示例

训练集共有图片 25000 张，其中猫狗各一半，测试集共有图片 12500 张。从可视化结果可以看出，每张图片的大小不同，以现实生活作为背景，大部分图片质量和分辨率良好，会有小部分图片的分辨率小而且质量差，例如：图上的右上角的那副图像，很难分辨出脸部重要特征加上它旁边的抱枕，会增加模型识别的难度。

另外训练集里面还包含了‘异常值’图片，以下是根据‘异常值’图片的尺寸和颜色的特征经过代码筛选而得出的，如图 2 所示：

图 2 ‘异常值’ 图片

这些图片总共 14 张左右，图片数量相对训练集的总数量来说是非常少的，对模型最后的预测能力的影响也是极小的，所以可以继续保留这些图片在训练集里面。

2.2 算法和技术

2.2.1 深度学习

由神经科学家麦卡洛克(W.S.McCilloch) 和数学家皮兹（W.Pitts）在《数学生物物理学公告》上发表论文《神经活动中内在思想的逻辑演算》（A Logical Calculus of the Ideas Immanent in Nervous Activity）。建立了神经网络和数学模型，称为 MCP 模型。所谓 MCP 模型，其实是按照生物神经元的结构和工作原理构造出来的一个抽象和简化了的模型，也就诞生了所谓的“模拟大脑”，人工神经网络的大门由此开启。

直到现在由神经网络之父 Geoffrey Hinton 在 1986 年发明了适用于多层感知器（MLP）的 BP（Backpropagation）算法，并采用 Sigmoid 进行非线性映射，有效解决了非线性分类和学习的问题。该方法引起了神经网络的第二次热潮。

当然神经网络在学术界又名深度学习（Deep Neural Network），它是机器学习的分支，是一种试图使用包含复杂结构或由多重非线性变换构成的多个处理层对数据进行高层抽象的算法。它是机器学习中一种基于对数据进行表征学习的算法。观测值（例如一幅图像）可以使用多种方式来表示，如每个像素强度值的向量，或者更抽象地表示成一系列边、特定形状的区域等。而使用某些特定的表示方法更容易从实例中学习任务（例如，人脸识别或面部表情识别）。深度学习的好处是用非监督式或半监督式的特征学习和分层特征提取高效算法来替代手工获取特征。

至今已有数种深度学习框架，如深度神经网络、卷积神经网络和深度置信网络和递归神经网络已被应用在计算机视觉、语音识别、自然语言处理、音频识别与生物信息学等领域并获取了极好的效果。

现神经科学被视为深度学习研究的一个重要灵感来源，但它已不再是该领域的主要导向。主要原因是没有足够的关于大脑的信息作为指导。要获得对大脑实际使用算法的深刻理解，我们需要有能力同时检测至少数千相连神经元的活动。现代科学无法做到这一点，甚至连大脑的最简单、最深入研究的部分人们都还远远没有理解。

现代术语“深度学习”超越了目前机器学习模型的神经科学观点。学习的多层次组合这一更普遍的原则更加吸引人，这可以应用于机器学习框架且不必是受神经启发的。20 世纪 80 年代，连接机制是在认知科学的背景下出现的。认知科学是理解心智，并结合多个不同层次分析的跨学科方法。连接机制的中心思想是，当网络将大量简单计算单元连接在一起时，可以实现智能行为。这种见解同样适用于与计算模型中隐藏单元作用类似的生物神经系统中的神经元。这一时期涌现出了众多的重要成果，很多一直沿用至今。

在那时，深度网络是难以训练的，其实在 20 世纪 80 年代就存在的算法能工作得非常好，但是直到 2006 年还是没有普及。这个问题只是因为计算复杂性太高，而且数据不多，加上当时可用的硬件难以进行足够的实验。事实上，这一阶段最重要的新进展是现在人们有了这些算法成功训练所需的数据资源。截至 2016 年，一个粗略的经验法则是，监督深度学习算法一般在每类给定约 5000 标注样本情况下可以实现可接受的性能，当至少

有 1000 万标注样本的数据集用于训练时将达到或超过人类表现。另外，随着数据量增加，模型的规模、复杂度和计算量也在快速地增加。

2016 年 3 月，由谷歌（Google）旗下 DeepMind 公司开发的 AlphaGo(基于深度学习)与围棋世界冠军、职业九段棋手李世石进行围棋人机大战，以 4 比 1 的总比分获胜；2016 年末 2017 年初，该程序在中国棋类网站上以“大师”（Master）为注册帐号与中日韩数十位围棋高手进行快棋对决，连续 60 局无一败绩；2017 年 5 月，在中国乌镇围棋峰会上，它与排名世界第一的世界围棋冠军柯洁对战，以 3 比 0 的总比分获胜。围棋界公认阿尔法围棋的棋力已经超过人类职业围棋顶尖水平。

2.2.2 深度卷积神经网络

卷积神经网络（Convolutional Neural Network，CNN），是 MLP 的生物启发的变体。从 Hubel 和 Wiesel 在猫的视觉皮层的早期工作中，我们知道视觉皮层包含一个复杂的细胞排列。这些细胞对视野的小部分区域敏感，被称为‘感受野’分区域平铺覆盖整个视野。这些单元作为输入空间上的局部滤波器，非常适合利用自然图像中强大的空间局部相关性。是一种专门用来处理具有类似网格结构的数据的神经网络。CNN 在诸多应用领域都表现优异。

在深度学习的大背景下，卷积神经网络的普遍深度已经达到几十层，这样的卷积神经网络被称为深度卷积神经网络（Deep Convolutional Neural Network，DCNN）。方便起见，在本文接下来的内容中，CNN 与 DCNN 等价。

卷积神经网络由几个层组成。这些图层可以有三种类型：

1. 卷积：卷积层由一个矩形的神经元网格组成。它要求前一层也是一个矩形的神经元网格。每个神经元从前一层的矩形部分输入； 这个矩形部分的权重对于卷积层中的每个神经元是相同的。因此，卷积层只是前一层的图像卷积，其中权重指定卷积滤波器。另外，每个卷积层中可能有几个网格； 每个网格从上一层的所有网格获取输入，使用潜在的不同过滤器。

2.最大池化：在每个卷积层之后，可能有一个池层。合并层从卷积层中提取小的矩形块并对其进行二次采样以从该块产生单个输出。有几种方法可以进行这种汇总，例如取平均值或最大值，或者是块中神经元的学习线性组合。我们的合并图层将始终是最大合并图层； 也就是说，他们占用了他们所在的区块的最大值。

3.完全连接：最后，在多个卷积层和最大池层之后，神经网络中的高层推理通过完全连接的层完成。一个完全连接的层将上一层中的所有神经元（不管它是完全连接的，合并的还是卷积的）连接到它所具有的每一个神经元。完全连接的图层不再在空间上定位（您可以将它们视为一维），因此完全连接的图层之后将不会有卷积图层。

卷积神经网络（CNN）由一个或多个卷积层（通常具有子采样步骤）组成，然后是一个或多个完全连接的层，如标准多层神经网络。CNN 的体系结构被设计为利用输入图像（或其它 2D 输入，例如语音信号）的 2D 结构。这是通过本地连接和绑定权重，然后通过某种形式的池化来实现的，这会导致翻译不变特征。CNN 的另一个好处是，与隐藏单元数量相同的全连接网络相比，它们更容易训练，参数也更少。

20 世纪 90 年代，LeNet 是第一个帮助推进深度学习领域的卷积神经网络之一。 自从 1988 年以来，Yann LeCun 的这个开创性的工作在许多先前的成功迭代之后被命名为 LeNet5。当时 LeNet 架构主要用于阅读邮政编码，数字等字符识别任务。其他相关网络包括：GoogLeNet（2014）、VGGNet（2014）、ResNets（2015），DenseNet（2016 年 8 月）。

CNN 在深度学习的历史中发挥了重要作用。它是将研究大脑获得的深刻理解成功用于机器学习应用的关键例子，也是第一个表现良好的深度模型之一。CNN 还是第一个解决重要商业应用的神经网络，并且仍然是当今深度学习商业应用的前沿。

它听起来像是一个奇怪的生物学和数学的组合，但是这些网络已经成为计算机视觉领域最有影响力的一些创新。2012 年是神奇网络成长的第一年，Alex Krizhevsky 用它们赢得了当年的 ImageNet 竞赛（基本上是计算机视觉年度奥运会），把分类错误记录从 26%降到了 15%，这个惊人的提高从那以后，许多公司一直在以服务为核心进行深度学习。Facebook 使用自动标记算法的神经网络，谷歌的照片搜索，亚马逊的产品推荐，Pinterest 的家庭饲料个性化和 Instagram 的搜索基础设施。

2.2.3 卷积网络的基本原理

理解卷积层的基本原理，需要先理解四个概念：1.卷积层；2.非线性；3.最大池化；4.全连接层。

1.卷积层（ConvNet）

图像是像素值的矩阵，实质上，每个图像可以表示为像素值的矩阵。如图 1 所示：

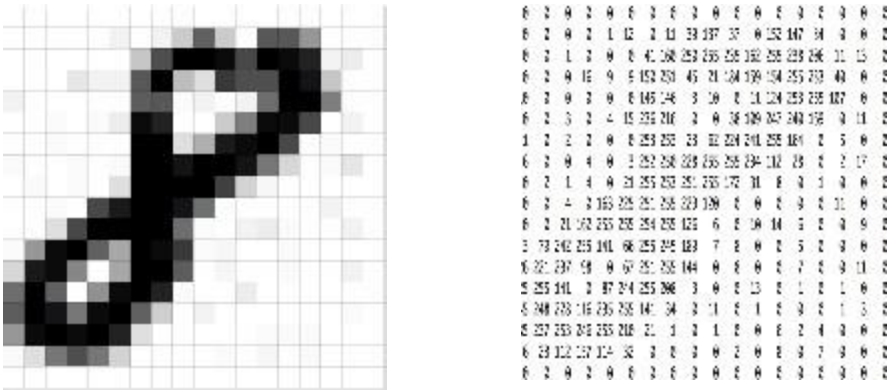
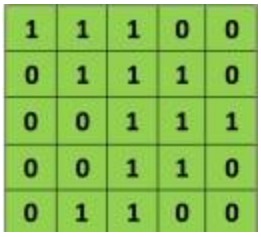


图 1

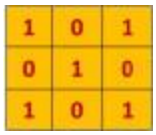
一个标准的数码相机的图像将有三个通道 - 红色，绿色和蓝色 - 你可以把它们想象成三个二维矩阵堆叠在一起（每种颜色一个），每个矩阵的像素值在 0 到 255 之间。我们将只考虑灰度图像，所以我们将有一个代表图像的 2d 矩阵。矩阵中每个像素的值将在 0 到 255 的范围内 - 零表示黑色，255 表示白色。

在 ConvNet 情况下，卷积的主要目的是从输入图像中提取特征。卷积通过使用输入数据的小方块来学习图像特征来保持像素之间的空间关系。我们不会在这里进入卷积的数学细节，但会试图了解它如何处理图像。

正如我们上面所讨论的那样，每个图像都可以被认为是像素值的矩阵。考虑像素值仅为 0 和 1 的 5×5 图像（注意，对于灰度图像，像素值范围从 0 到 255，下面的绿色矩阵是像素值仅为 0 和 1 的特殊情况）：



另外，考虑另一个 3 x 3 矩阵，如下所示：



然后，可以计算 5×5 图像和 3×3 矩阵的卷积，如下面的**图 2** 中的动画所示：

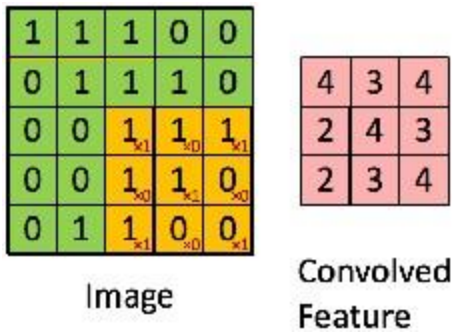
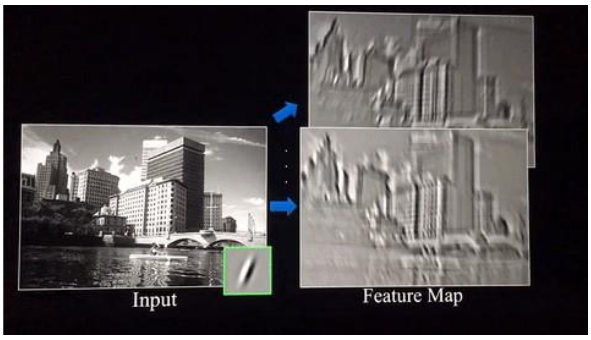


图 2

我们将原始图像（绿色）上的橙色矩阵滑动 1 个像素（也称为“步幅”），并且对于每个位置，我们计算元素之间的相乘（两个矩阵之间）并添加乘法输出以获得形成的最终整数输出矩阵的单个元素（粉红色）。请注意，3×3 矩阵仅在每个步幅中“看到”输入图像的一部分。

在 CNN 术语中，3×3 矩阵被称为“ 滤波器 ”或“内核”或“特征检测器”，并且通过将滤波器滑过图像并计算点积而形成的矩阵称为“相关特征”或“激活地图“或” 功能地图 “。重要的是要注意，滤镜作为来自原始输入图像的特征检测器。从上面的图片可以看出，滤波器矩阵的不同值将针对相同的输入图像产生不同的特征图。

下面的图 3 可以直观看出卷积层扫描到的特征图：



上面的图像和两个滤镜只是数字矩阵。实际上，CNN 在训练过程中自己学习这些滤波器的值（尽管在训练过程之前，我们仍然需要指定诸如滤波器数量，滤波器大小，网络结构等参数）。我们拥有的滤镜数量越多，提取的图像特征越多，我们的网络在识别未知图像中的图案时就越好。

特征图（Convolved Feature）的大小由我们在执行卷积步骤之前需要决定的三个参数控制：

- **深度：**深度对应于我们用于卷积运算的滤波器数量。在图 4 所示的网络中，我们使用三个不同的滤波器对原始船图像进行卷积，从而产生如图所示的三个不同的特征图。您可以将这三个特征地图视为堆叠的二维矩阵，因此特征地图的“深度”将为三。

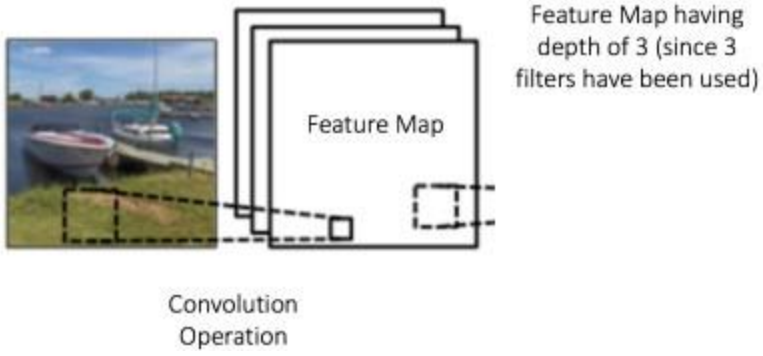


图 4

- **步幅：**我们将滤波器矩阵滑过输入矩阵的像素数。当步幅是 1 时，我们一次移动一个像素的滤镜。当步幅是 2 时，那么当我们滑动它们时，滤波器一次跳跃 2 个像素。有更大的步幅会产生更小的特征图。
- **零填充：**有时候，将输入矩阵填充到边界周围是很方便的，这样我们可以将滤波器应用到我们输入图像矩阵的边界元素上。零填充的一个很好的特性是它允许我们控制特征映射的大小。添加零填充也被称为宽卷积，而不使用零填充将是一个狭窄的卷积。这已经在中得到了明确的解释。

2.非线性操作（ReLU）

在每个卷积操作之后，需使用称为 ReLU 的附加操作。ReLU 代表整流线性单位，是一个非线性操作。其输出由下图所示：

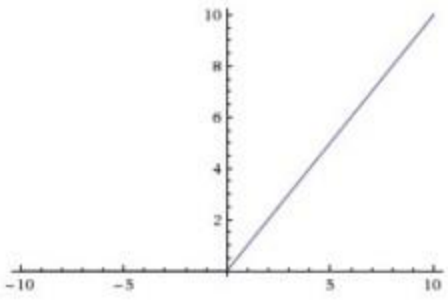


图 5

ReLU 是将特征映射中的所有负像素值替换为零。ReLU 的目的是在我们的 ConvNet 中引入非线性，因为我们希望我们的 ConvNet 学习的大部分实际数据是非线性的。从下面的图 6 可以清楚地理解 ReLU 操作。这里的输出特征映射也被称为“整理”特征映射。

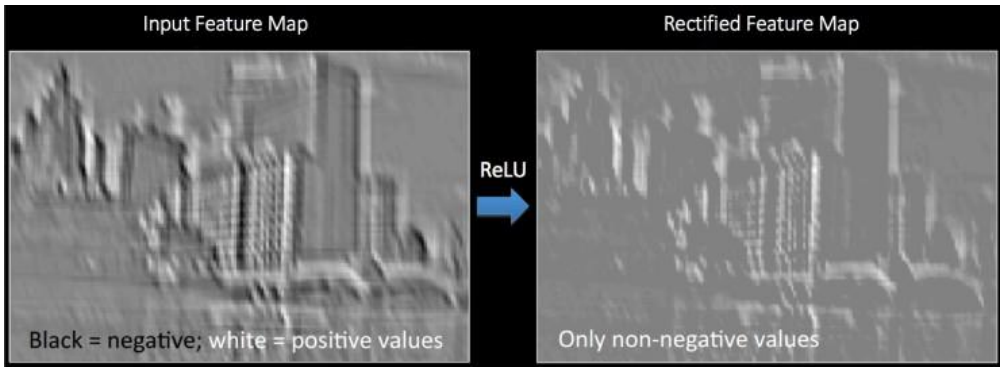


图 6

其他非线性函数，如 tanh 或 sigmoid 也可以用来代替 ReLU，但是在大多数情况下，ReLU 被发现性能更好。

3.最大池化（Max Pooling）

在最大池化的情况下，我们定义一个空间邻域（例如，一个 2×2 窗口），并从该窗口内的整形特征映射中取最大的元素。而不是采取最大的元素，我们也可以采取平均（平均合并）或该窗口中的所有元素的总和。在实践中，最大池化已被证明效果更好。

图 7 显示了通过使用 2×2 窗口在整流特征映射（在卷积+ ReLU 操作之后获得）上的最大池操作的示例。

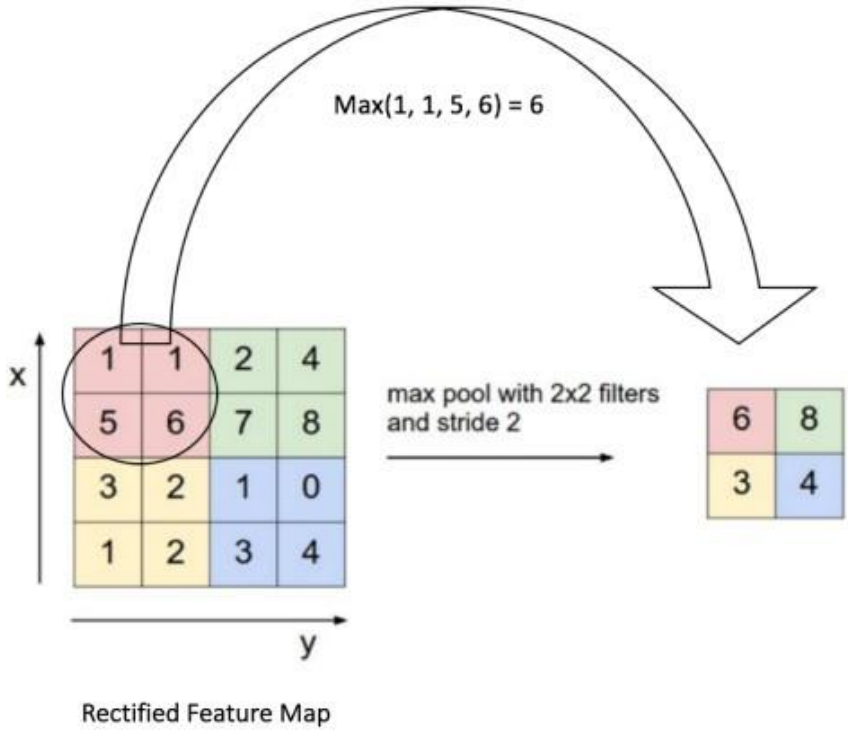


图 7

我们将 2 x 2 窗口滑动 2 个单元（也称为“步幅”），并在每个区域中取最大值。图 7 所示，这减少了我们的特征映射的维数。

图 8 显示了在上面的图 6 中的 ReLU 操作之后进行最大池化，我们收到的汇集对整流特征映射的影响。

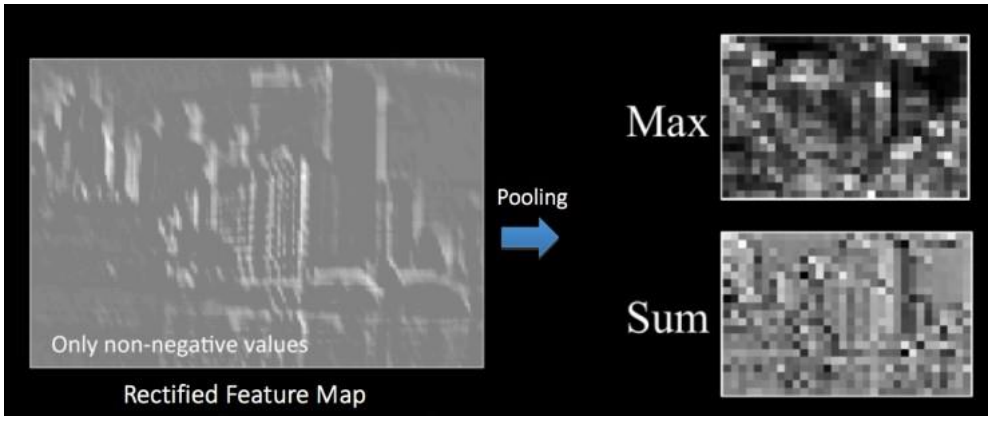


图 8

最大池化的功能是逐步减少输入表示的空间大小。特别是汇集，使输入表示（特征维度）更小，更易于管理；减少了网络中的参数和计算次数，因此，控制过拟合；使网络对输入图像中的小变换，失真和平移不变（输入中的小失真不会改变最大池化的输出 - 因为我们取本地域中的最大/平均值）；帮助我们达到几乎不变的表象形象。这是非常强大的，因为我们可以检测图像中的物体，而不管它们位于何处。

4.完全连接层（Fully Connected）

完全连接层是一个传统的多层感知器，它在输出层中使用了 softmax 激活函数。术语“完全连接”意味着前一层中的每个神经元都连接到下一层上的每个神经元。卷积层和汇聚层的输出表示输入图像的高级特征。完全连接层的目的是使用这些特征来根据训练数据集将输入图像分类成各种类别。例如，我们要执行的图像分类任务有四个可能的输出，如下面的图 9 所示：

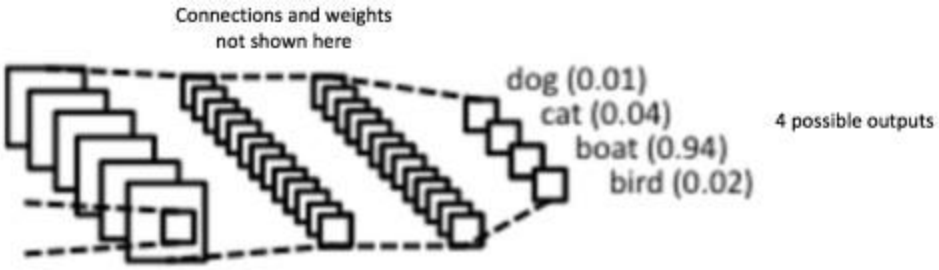


图 9

除了分类之外，添加完全连接的层也是学习这些特征的非线性组合的（通常）廉价的方式。来自卷积层和汇集层的大部分特征对于分类任务来说可能是好的，但是这些特征的组合可能更好。

完全连接层的输出概率总和为 1.这是通过使用 Softmax 作为完全连接层的输出层中的激活功能来保证的。Softmax 函数采用任意实值分数的向量，并将其压缩到 0 和 1 之间的值的向量，其总和为 1。

原理演示：

如上所述，卷积+最大池化层从输入图像充当特征提取器，而完全连接层充当分类器。构成一个基础模型，训练过程当中利用反向传播更新权值，让预测结果接近实际结果。

在下面的图中，由于输入图像是一条船，所以 Boat 类的目标概率是 1，而其他三个类的目标概率是 0,如图 10 所示：

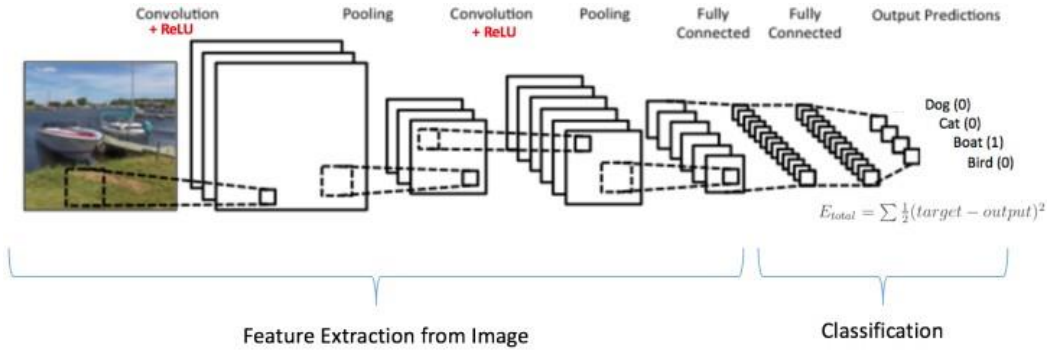


图 10

2.2.4 技术实现

实现 CNN 使用的开发语言是 Python,在 Jupyter Notebook 平台下使用 Google 开源的 TensorFlow, 项目同时使用了开源机器学习库 Keras，作为对 TensorFlow 的上层封装，以便能够使用更多的高级 API，加快开发的速度。另外，由于使用 CNN 处理大分辨率全彩色图片所需的计算量非常的大，项目使用了 GPU 进行计算加速, 需要安装 CUDA 和 cuDNN。

卷积神经网络的实现过程可概括如下：

步骤 1：我们用随机值初始化所有滤波器和参数/权重。

步骤 2：网络将训练图像作为输入，经过正向传播步骤（卷积，ReLU 和共同操作以及全连接层中的前向传播），并找出每个类别的输出概率。

可以说，上面的船形图像的输出概率是[0.2, 0.4, 0.1, 0.3]，由于权重是随机分配的第一个训练样例，输出概率也是随机的。

步骤 3：计算输出层的总误差（所有 4 个类的总和），总误差=Σ½（目标概率 – 输出概率）²

步骤 4：使用反向传播计算相对于网络中所有权重的误差梯度，并使用梯度下降更新所有滤波器值/权重和参数值，以最小化输出误差。权重根据它们对总误差的贡献进行调整。

当再次输入相同的图像时，输出概率现在可能是[0.1,0.1,0.7,0.1]，这更接近目标向量[0,0,1,0]。

这意味着网络已经学会了通过调整其权重/滤波器来正确分类这个特定的图像，从而减少输出误差。

滤波器数量，滤波器大小，网络架构等参数在步骤 1 之前都已经被固定，在训练过程中不会改变 – 只有滤波器矩阵和连接权值被更新。

步骤 5：对训练集中的所有图像重复步骤 2-4。

上述步骤训练 ConvNet – 这基本上意味着 ConvNet 的所有权重和参数现在已经被优化，以正确分类来自训练集的图像。当一个新的（看不见的）图像被输入到 ConvNet 中时，网络将经过正向传播步骤并输出每个类别的概率（对于新图像，使用已经被优化的权重来计算输出概率以正确分类所有以前的训练实例）。如果我们的训练集足够大，网络将（希望）很好地概括新图像并将其分类到正确的类别。

由于上述的大部分实现都是基于高级语言的，而即使是执行最简单的操作，高级语言也会比低级语言消耗更多的 CPU 周期，因此运算缓慢就成了高级语言的一个天然的缺陷。目前，主流深度学习框架都是利用脚本语言实现前端建模，用低级语言如 C++实现后端运行。一个显著的加速手段就是利用现成的扩展包，开发者可以根据个人喜好灵活选择。另外，一般的 BLAS 库只是针对普通的 CPU 场景进行了优化，但目前大部分的深度学习框架都已经开始采用并行 GPU 的运算模式，因此利用诸如 NVIDIA 推出的针对 GPU 优化的 cuBLAS 和 cuDNN 等更据针对性的库是更好的选择。

2.3 基准指标

关于图像分类问题上，ILSVRC 每年都会举办的著名比赛 ImageNet，它是来自工业界和学术界的团队尝试搭建，用于物体识别和定位的最佳神经网络。

到目前为止，对 ImageNet 数据集上的 1000 个类别进行分类，最好的神经网络 ResNet 在 ImageNet 上的错误率仅为 3%，这实际上比人类正常的识别率还要高。本次项目我使用的 VGGNet 来针对猫狗进行分类，在 ImageNet，它的分类错误率降到了 7%，它由牛津大学的视觉几何小组研发，VGG 结构简单和优雅，非常适合迁移学习的模型。

考虑到训练 CNN 需要大量的硬件资源与计算时间，受限于客观条件，项目设定的目标为分类正确度为 0.95。

3.具体方法

3.1 迁移学习

在实践中，很少有人从零开始训练整个卷积网络（随机初始化），因为拥有足够大小的数据集相对来说比较少见。相反，在一个非常大的数据集（例如 ImageNet，包含有 1000 个类别的 120 万个图像）在 ConvNet 是常见的，然后使用 ConvNet 作为初始化或固定特征提取器来处理感兴趣的任务。三种主要的转移学习方案如下所示：

固定特征提取器：在 ImageNet 上预先训练一个 ConvNet，删除最后一个完全连接的图层（这个图层的输出是 ImageNet 等不同任务的 1000 个分数），然后将其余的 ConvNet 作为新数据集的固定特征提取器。在一个 AlexNet 中，这将为包含紧接在分类器之前的隐藏层的激活的每个图像计算 4096-D 向量。我们将这些功能称为 **CNN 代码（卷积码）**。如果这些代码在 ImageNet 上的 ConvNet 培训期间也被限制，则这些代码被重新定义（即，阈值为零）对于性能是重要的。一旦为所有图像提取了 4096-D 代码，为新数据集训练一个线性分类器（例如 Linear SVM 或 Softmax 分类器）。

微调：第二个策略是不仅要在新数据集上的 ConvNet 之上替换和重新训练分类器，而且还要通过继续反向传播来微调预训练网络的权重。可以对 ConvNet 的所有层进行微调，或者可以保留一些先前的层（由于过度拟合的关注），并且只对网络的更高层部分进行微调。这是由 ConvNet 的早期特征包含更多通用特征（例如边缘检测器或颜色斑点检测器）的观察所驱动的，这些特征对于许多任务应该是有用的，但后来的 ConvNet 层次逐渐变得更具体到类别的细节包含在原始数据集中。例如，在包含许多狗品种的 ImageNet 中。

预训练模型：由于现代 ConvNets 需要 2-3 周的时间才能在 ImageNet 上通过多个 GPU 进行培训，因此人们常常会发布最终的 ConvNet 检查点，以利于其他可以使用网络进行微调的人员的效率。

何时以及如何进行微调？你如何确定你应该在新的数据集上执行什么类型的传输学习？这是几个因素的函数，但其中最重要的两个是新数据集（小或大）的大小，以及它与原始数据集的相似性（例如 ImageNet，就图像内容和类别而言，或非常不同，如显微镜图像）。请记住，ConvNet 特征在早期层次上更为通用，而在后面的层次中更具有原始数据集特有的特性，下面是导航 4 个主要场景的一些常见经验规则：

1.新数据集很小，与原始数据集相似。由于数据量很小，因过度安装问题而对 ConvNet 进行微调并不是一个好主意。由于数据与原始数据相似，我们预计 ConvNet 中的更高级别的特征也与此数据集相关。因此，最好的想法可能是在 CNN 代码上训练一个线性分类器。

2.新数据集很大，与原始数据集相似。由于我们有更多的数据，我们可以有更多的信心，如果我们试图通过整个网络进行微调，我们不会过度适应。

3.新数据集很小，但与原始数据集非常不同。由于数据很小，因此最好只训练一个线性分类器。由于数据集是非常不同的，所以最好不要训练网络顶部的分类器，它包含更多的数据集特定功能。相反，它可能更好地训练 SVM 分类器从早期的网络激活。

4.新的数据集很大，与原始数据集有很大的不同。由于数据集非常大，我们可以期望我们可以从头开始培训一个 ConvNet。然而，在实践中，从预训练模型的权重进行初始化通常还是有益的。在这种情况下，我们将有足够的数据和信心通过整个网络进行微调。

进行转移学习时，还需要记住一些其他的事项：

1.来自预训练模型的限制。请注意，如果您希望使用预训练网络，那么您可能会在可用于新数据集的体系结构方面受到一些限制。例如，你不能任意从预训练网络中取出 Conv 层。然而，一些变化是直接的：由于参数共享，你可以很容易地在不同空间大小的图像上运行预训练网络。这在 Conv / Pool 层的情况下是明显的，因为它们的前向函数独立于输入体积的空间大小（只要步幅“合适”）。在 FC 层的情况下，这仍然适用，因为 FC 层可以转换为卷积层：例如，在 AlexNet 中，第一个 FC 层之前的最终池化卷大小为[6x6x512]。

2.学习率。与计算新数据集的类别分数的新线性分类器的（随机初始化的）权重相比，对正在进行微调的 ConvNet 权重使用较小的学习率是很常见的。这是因为我们期望 ConvNet 权重相对较好，所以我们不希望太快和太多地扭曲它们（特别是当它们上面的新线性分类器正在从随机初始化进行训练时）。

3.1.1 模型选择

CNN 的核心是对图像特征的表示，从最基本的边缘、纹路，到较高抽象层次的对象。CNN 通过各层卷积核及其对应的权重来记录这些特征，同时，通常情况下利用最后一层的 softmax 进行概率分配，从而得出分类结果。由此可见，大部分卷积神经网络是用来做特征提取的，只有最后的很少几层是用来得出分类结果的。

从零开始训练一个卷积神经网络，需要进行细致的网络结果设计，并调整大量的参数进行优化。幸运的是，在知名计算机视觉竞赛 ImageNet 中，类别就含有猫和狗，同时，许多优秀的网络结构都提供了在 ImageNet 数据集上的训练结果。由此，利用现有的模型以及预训练出的权重，就能很好的表征猫和狗的特征，再辅以一个简单的分类器就能够获得一个相对理想的结果。

本项目使用的是 Tensorflow 库，其官方 Github 网站提供了 AlexNet、 VGG16、VGG19、InceptionV3、 ResNet50 和 Xception 这 6 个模型及其在 ImageNet 数据上的训练结果。由于本地机器配置较低，我选了入门级的 VGG16 来完成迁移学习并微调该模型成为针对解决本次项目的新的模型。

VGGNet 是非常棒的，因为它非常简单，性能卓越，在 ImageNet 竞赛中名列第二。这里的想法是，我们保留所有的卷积图层，但是用我们自己的分类器来替换最终完全连接的图层。这样，我们可以使用 VGGNet 作为我们图像的特征提取器，然后轻松地训练一个简单的分类器。我们要做的是采用 4096 个单位的第一个完全连接层，包括与 ReLUs 阈值。我们可以使用这些值作为每个图像的代码，然后在这些代码之上构建一个分类器。

3.1.2 数据预处理及模型调整

由于是基于现有模型和权重进行计算，因此数据预处理必须与想用的模型匹配。通过阅读源码可知，VGG16 模型在只读取 224x224 图片，所以需要先对训练集做图片大小统一的预处理。然后将训练集分成 80%为训练数据，20%为验证数据。

另外，训练集中存在‘异常值’图片，这些图片属于极少数，在训练模型的过程中结果的影响很小，所以我继续保留它们。

下面，我们将遍历数据集中的所有图像，并获取每个图像的代码。也就是说，我们将通过 VGGNet 卷积层运行图像，并记录第一个完全连接层的值。然后，我们可以将这些文件写入一个文件，以便以后建立我们自己的分类器。

在这里，我们正在使用该 vgg16 模块 tensorflow_vgg。网络拍摄的图像尺寸为 224×224×3 作为输入。那么它有 5 组卷积层。这里实现的网络具有这种结构（从源代码复制）：

```
self.conv1_1 = self.conv_layer(bgr, "conv1_1")

self.conv1_2 = self.conv_layer(self.conv1_1, "conv1_2")

self.pool1 = self.max_pool(self.conv1_2, 'pool1')


self.conv2_1 = self.conv_layer(self.pool1, "conv2_1")

self.conv2_2 = self.conv_layer(self.conv2_1, "conv2_2")

self.pool2 = self.max_pool(self.conv2_2, 'pool2')


self.conv3_1 = self.conv_layer(self.pool2, "conv3_1")

self.conv3_2 = self.conv_layer(self.conv3_1, "conv3_2")

self.conv3_3 = self.conv_layer(self.conv3_2, "conv3_3")

self.pool3 = self.max_pool(self.conv3_3, 'pool3')


self.conv4_1 = self.conv_layer(self.pool3, "conv4_1")

self.conv4_2 = self.conv_layer(self.conv4_1, "conv4_2")

self.conv4_3 = self.conv_layer(self.conv4_2, "conv4_3")

self.pool4 = self.max_pool(self.conv4_3, 'pool4')


self.conv5_1 = self.conv_layer(self.pool4, "conv5_1")
```

```
self.conv5_2 = self.conv_layer(self.conv5_1, "conv5_2")

self.conv5_3 = self.conv_layer(self.conv5_2, "conv5_3")

self.pool5 = self.max_pool(self.conv5_3, 'pool5')


self.fc6 = self.fc_layer(self.pool5, "fc6")

self.relu6 = tf.nn.relu(self.fc6)
```

所以我们想要的是第一个全连接层的值，在 ReLUd（self.relu6）之后。我们可以从以下图片来更直观来看，VGG16 基本的架构如图 11：

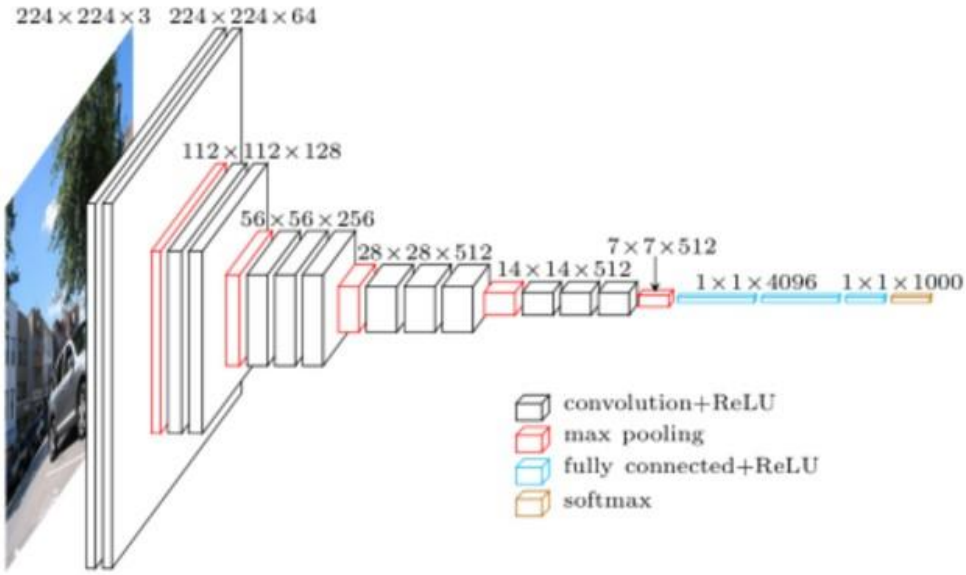


图 11

从 VGG16 基本的架构可以看到该全连接层在第 19 层，这里之前就是我们的特征提取器。经过项目提供的训练集对其进行后训练后，我们得到特征提取器的数据文件，将其保存在本地。离线式保存可以在接下来计算中只对全连接层进行训练，效率很高。

当然你可以不必要再次训练已经训练好的 VGG 模型，我之所以这么做是因为训练好的模型的特征方向不一定全符合本项目的数据特征，也许偏离较远。如果对训练好的模型再一次针对自己项目的数据进行训练，特征方向会靠拢本项目图像的特征，模型准确率会更高。为验证我的想法，我决定以这种方式来解决该分类问题。

3.2 实现

3.2.1 模型的训练

首先对训练数据进行增强变换，水平反转图像，像镜子一样，在 0.05 范围内缩放、在 0.05 范围内上下/左右平移、在 5 度范围内旋转。增强数据有两个好处：1.我们有更多数据可用于训练网络；2.我们用于训练网络的数据更加全面。

接下来对训练数进行小批量处理，每一批的大小为 32 个样本。优化方法是用的是 Adam ，epoch 为 10，初始学习率为 0.001，学习率随着训练过程进行衰减。运行后模型会自动保存准确度和权重数据，以及训练的日志。我们来看训练过程中训练损失值曲线图 12：

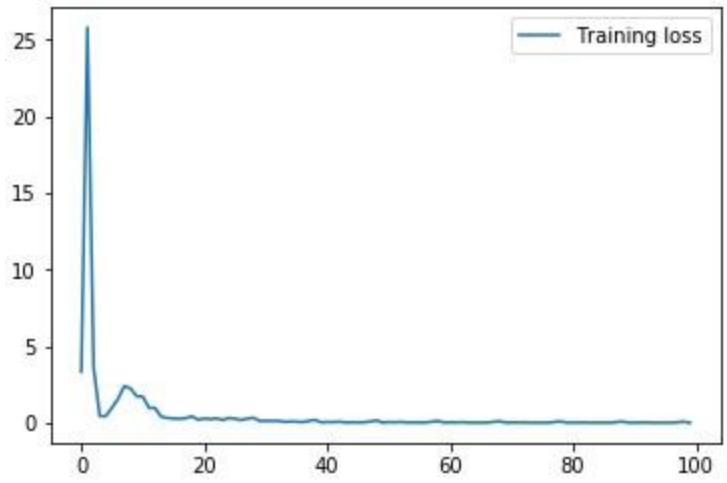


图 12 训练过程的损失值

一开始训练损失值还是较低的，但一下子就很高，最后接近零，是因为这里的反向传播起的作用，梯度下降的正确方向找到了，所以训练损失值一下低了下来。

3.2.1 结果的生成

训练数据经过批次处理后输入模型进行训练，每当迭代 5 次时就会使用验证集验证一下模型准确度，并将预测结果按照格式要求写入到 csv 文件中。并验证结果文本打印出来，图 13 为本次验证集的准确度曲线图：

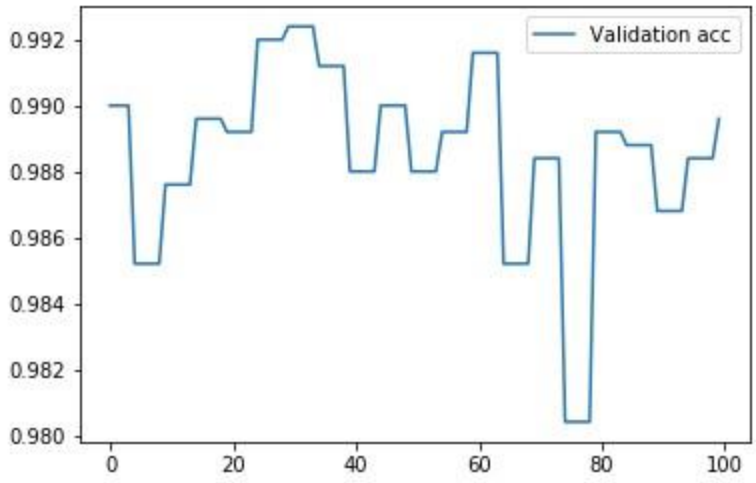


图 13

从图片可以看出模型准确度曲线不稳定，在迭代 30 次左右时为最高峰，但迭代到 80 次左右时又是准确度又是最低的，这里还需继续改进。

3.3 改进

本模型使用的 VGGNet，使用权重迁移，现在我们已经有了特征提取器，微调模型的第七全连接层（Layer7），这层使用具有 256 个单位的完全连接层级，因为第六全连接层（Layer6）输出的代码宽为 4096 个值，所以如果把 256 个单位改成 512 个单位，预测效果可能会更好。

下面我们修改模型参数后进行的训练的损失结果图 14（最后损失值为 0.01619）：

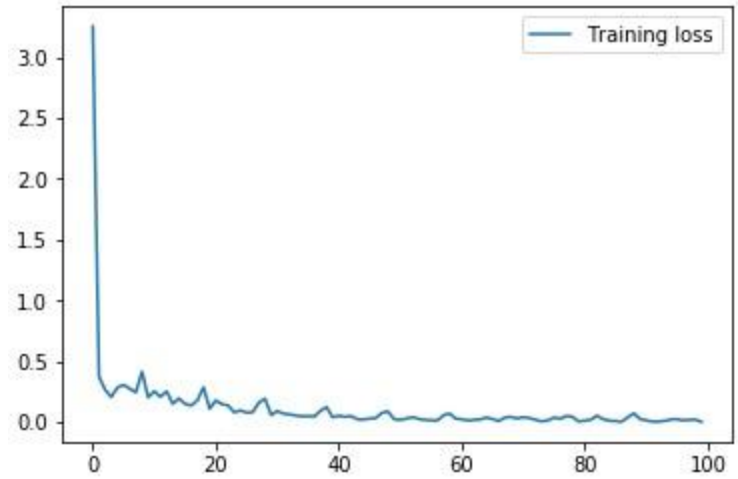


图 14 训练过程损失值

验证准确度结果，如图 15：

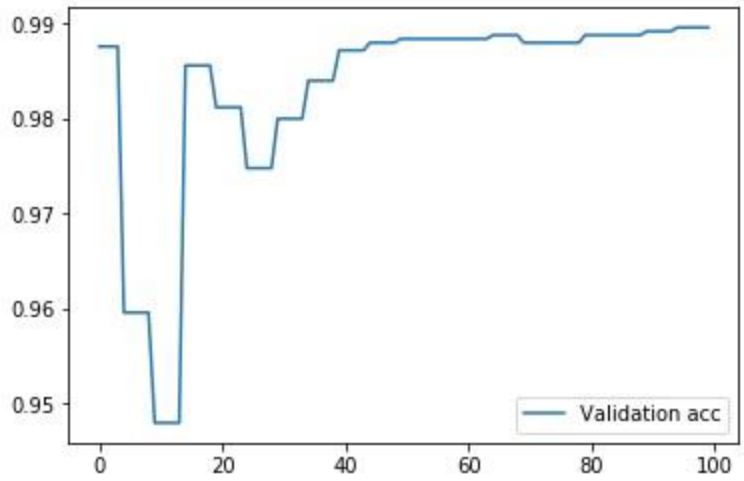


图 15 验证集准确度

从两个图可以看到，经过修改后的模型训练和验证的结果越来越稳定，损失值越接近于零，准确度越靠近 0.99。（最后的准确度为 0.9896）

4.结果

4.1 模型评价与验证

本次 VGGNet 的网络深度有 8 层，经过微调后变成了 7 层，训练的 VGGNet 时所占用的内存是我机器的 80%左右，这也是我选择 VGGNet 的原因，如果网络深度再深的话，我的电脑基本上内存不足。

当然这里也有一个解决办法，把训练好的 VGGNet，GoogLeNet，ResNet 模型的特征提取文件下载下来，然后建立新的全连接层，对其进行训练，如果这些特征符合本项目的数据特征，效果是不错的，因为训练过的模型，已经有重要的特征对其识别。该方法计算量最少并节省时间。

这里我选择 VGGNet 又一个原因，我想要验证我一个想法：虽然直接下载下来提前特征也能起好的效果，但准确率不一定高，因为特征方向不一定全符合本项目的数据特征，也许偏离较远。如果对训练好的网络再一次训练数据，特征方向会靠拢本项目图像的特征，模型准确率会更高。

现在我们用测试集对模型进行测试，得出的结果是：

```
INFO:tensorflow:Restoring parameters from checkpoints\cat_dog.ckpt
Test accuracy: 0.9920
```

准确率为 0.992，这个准确率很高了，基本上能够确定我当初的想法，给 VGGNet 再次确定特征方向，模型预测能力更好。

4.2 结果分析

这次我们使用项目自带的测试集，该测试集有 12500 张图片，每张图片的特征跟训练集是基本一致，我们抽取其中的一张图片对模型进行再次验证，看看模型分类能力的表现。

如图 16 我们抽取的图片：

图 16

该图片输入模型后，我们来查看 softmax 具体输出，如图 17：

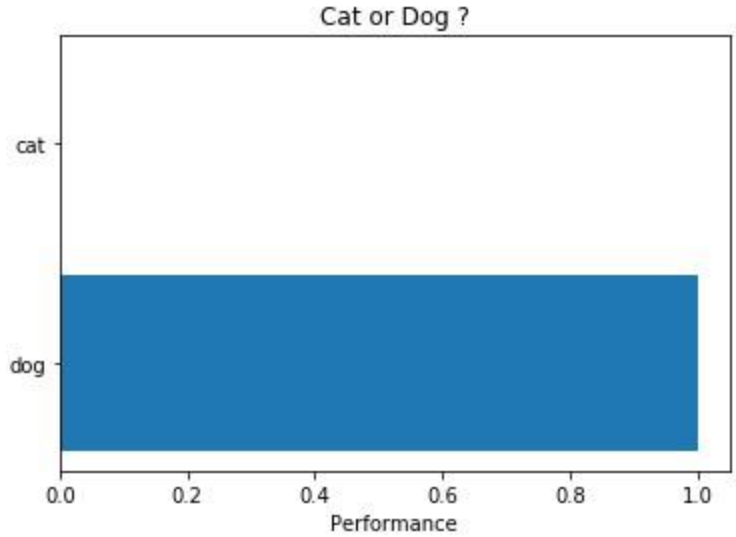


图 17

在没有狗的脸部特征，仅仅依赖狗的形状就可以正确识别这是一只狗，这说明了训练好的 VGGNet 已经具备狗的形状特征，所以能预测准确。当然，如果预测的是猫，模型也肯定会从形状、脸部重要特征等来预测。

5.结论

猫狗大战，一个充满挑战很有趣的竞赛，有很多方案可以完成该项目，选择不同的方案都有好处和坏处，但能够最后得出一个显著竞争力的模型是非常不容易的。CNN 是一个非常灵活的模型，围绕分类以及特征表示，有非常多的方法可以运用，使用 CNN 构建的计算机视觉分类模型是非常强大的。

对于竞赛而言，可以针对项目自带的测试集不断优化模型，例如，对一些识别还不太确定的猫或者狗的图片收集起来后增强数据进行训练等。这里我不多讨论自带测试集的情况，因为图像已经在自带测试集里，工程师都可以自行优化。

这里我更希望模型能够对现实中出现的一些特别情况能够进行正确处理，因为这个竞赛的目的是能够让模型适用于现实世界各种情况。

我们来看一个图片，如图 18：

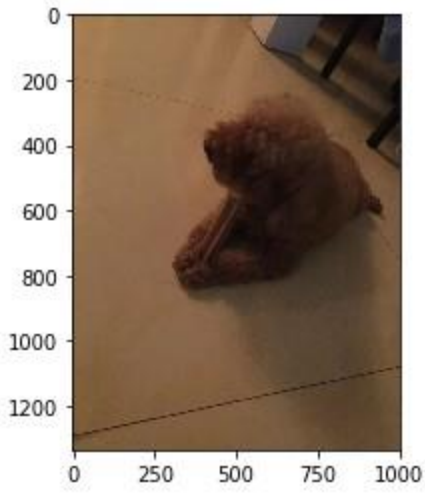


图 18

这是一张比自带测试集的图片难度更大的图片，首先我们人眼可以识别这是泰迪狗，因为泰迪狗毛形状和颜色能看出。但是对于模型来说没有了重要的脸部特征（因为被狗毛遮挡了），而且从这个角度拍摄，狗的耳朵是拍不出来的，它的体形状变小，而且四个脚形状也很模糊，这时候模型的预测会是如何？

请看图 19：

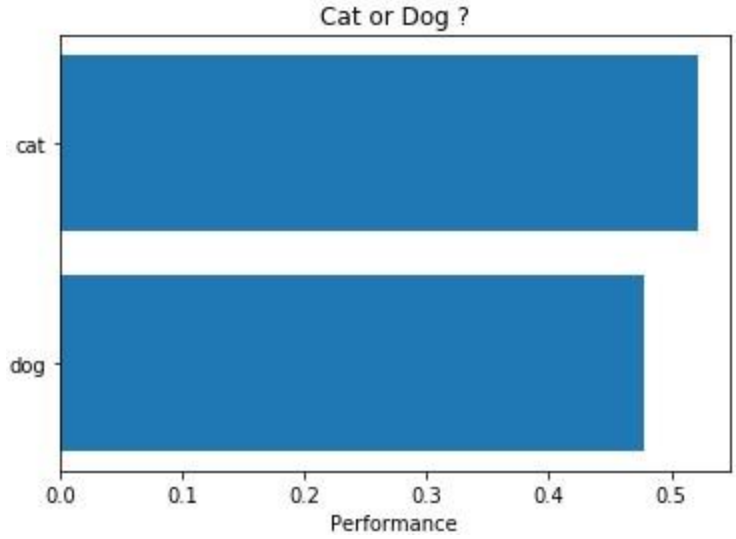


图 19

这个时候其实模型不太确定它是猫还是狗，从结果来看模型最终会认为它是猫，当然这是比较特殊的情况，遇到这种问题时候，我们应该收集泰迪狗的各种图片，然后增强数据来训练模型，再进一步加深网络就能解决问题，这里想表达的是预测能力再好的模型对特别情况其实也会出现误判。对应的无人驾驶汽车，出现在复杂的环境，也会可能出现问题。这就需要模型不断在现实生活当中得到运用，跟人一样积累经验，不断完善才能把误判减到最低。

通过这个项目，很大程度上提高了对 CNN 的理解及运用，能够更加灵活解决模型出现的各种问题，让模型表现达到最佳。

参考文献：

[1]《Neural Networks and Deep Learning》author：Michael Nielsen / Dec 2017

[2]François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. arXiv:1610.02357

[3]Shaoqing Ren, et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, 2015, arXiv:1506.01497

[4]CS231n Convolutional Neural Networks for Visual Recognition, Stanford

[5]Machine Learning is Fun! Part 3: Deep Learning and Convolutional Neural Networks

[6]Neural Networks by Rob Fergus, Machine Learning Summer School 2015

[7]A Beginner’s Guide To Understanding Convolutional Neural Networks

[8]Vincent Dumoulin, et al, “A guide to convolution arithmetic for deep learning”, 2015, arXiv:1603.07285