中南林业科技大学

毕业设计说明书

学生姓名：莫一凡　学　号：20172641

学　　院：计算机与信息工程学院

专业年级：2017 级计算机与科学技术

题　　目：基于深度学习的狗图像分类识别研究

指导教师：周慧斌　讲师

评阅教师：

2021 年 5 月

摘 要

普通计算机程序在处理人看来很简单的图像分类任务时一般效果不佳，人眼在处理图像这样拥有大量信息但是整体大于细节的任务时，一瞬间就能完成图像识别分类，而传统程序处理这样非逻辑的任务时，原先引以为傲的强大计算能力无法发挥长处。

深度学习在计算机视觉方向有了长足的发展，在领域内很多方面已经完全达到人能所做到的识别速度和准确度，甚至是超越人类。AI 计算机视觉已经在日常生活的各方各面产生了深远的影响，例如：面部识别支付，面部识别打卡，AI 换脸美图.....

针对流行的 kaggle AI 竞赛平台中的 dog-breed-Identification 竞赛，本文研究了多种流行的网络结构，在 kaggle 平台提供的训练集上进行训练，并且在测试集上验证，最终产生评分。

过程中使用了迁移学习以提高特征提取的能力，针对原数据集规模较小的问题采用了补充额外训练数据集的方式，引入了斯坦福大学的狗图片数据集，采用 GPU 进行训练，加大训练轮次。最重要的是，同时采用了 VGG16 和 resNet 两种网络模型，在网络复杂度方面设置对照比较，目的在于探索网络深度和复杂度对分类器的提升效果。

关键词：深度学习；图像分类；卷积神经网络；kaggle；狗

Title Dog image classification and recognition based on deep learning

Abstract:

Ordinary computer programs usually have poor results in dealing with the image classification task that people think is very simple. Human eyes can complete the image recognition and classification in an instant when they deal with the task that has a lot of information but the whole is larger than the details. However, when traditional programs deal with such non logical tasks, the powerful computing power that they were proud of cannot play its advantages.

Deep learning has made great progress in the direction of computer vision. In many aspects of the field, it has reached the recognition speed and accuracy that people can do, even surpassing human beings. AI computer vision has had a profound impact on all aspects of daily life, such as: facial recognition payment, facial recognition punch, AI face changing

In the process, transfer learning is used to improve the ability of feature extraction. In view of the small scale of the original data set, the way of adding additional training data set is adopted. The dog image data set of Stanford University is introduced, and GPU is used for training to increase the training rounds. Most importantly, vgg16 and RESNET network models are used at the same time to compare the network complexity. The purpose is to explore the improvement effect of network depth and complexity on the classifier.

Keywords: Deep learning; Image classification; Convolutional Neural Network; kaggle; dog

目　录

1. 引言

1.1.深度学习背景

在 1943 年由神经科学家麦卡洛克(W.S.McCilloch) 、数学家皮兹（W.Pitts）提出建立了神经网络最初的原型以及它的数学模型，最早称之为 MCP 模型，这是仿生生物神经元结构设计的模型，在当时只是一个理论的雏形，但神经网络的繁荣发展由此开端。

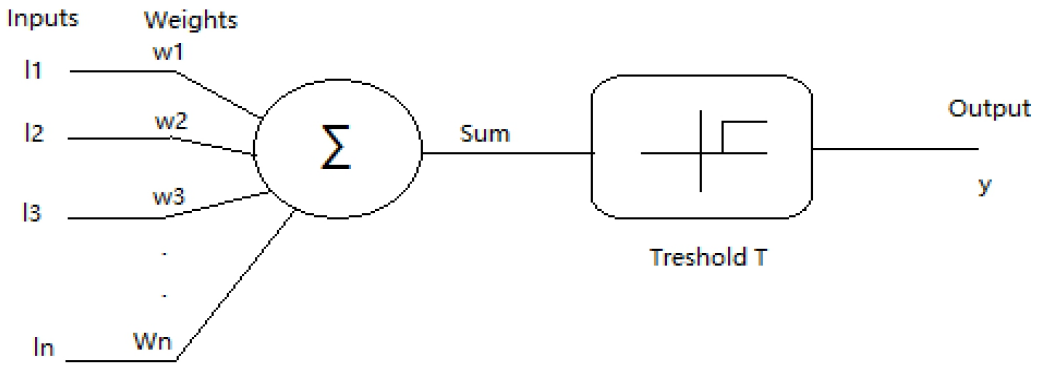

图 1.1

后来的科学家罗森布拉特首次提出了一种由两层多个神经单元组成的网络，这就是著名的多层感知器。并且将 MCP 用在了机器学习的分类算法。使用了梯度下降算法，从训练样本中自动的学习更新 weight，该方法被证实在数学层面是可以收敛的，这引起了第一次的神经网络领域的火热发展。

再后来被人反驳说神经网络只是一种线性模型，只能处理简单的线性分类问题，并且被证实无法解决非线性问题，甚至是简单的异或问题都不能正确分类，感知器跌落神坛，此后 20 年时间神经网络没有得到有效的发展。

在 1986 年神经网络之父 Geffrey Hinton 提出了能够训练多层感知器的 backpropagation 算法，简称 BP 算法，BP 算法利用感知机反向梯度下降，能够自训练调整模型参数。并且采用 Sigmoid 函数作为激活函数，能够将感知机的输入输出映射为非线性的关系。这破解了感知器不能进行非线性问题的分类。这一重大突破让神经网络得到第二次风靡。

此后 BP 算法遇到了瓶颈，在一些场景下会出现梯度消失问题，导致模型不可训练，这主要是由于 sigmoid 函数的特性导致的，在较大值和较小值时梯度变

化极小，传导到前层的误差梯度几乎为 0，无法进行参数的调整。再加上 90 年代时机器学习其他方面百花齐放，诞生了很多优秀的算法，投身深度学习的力量开始变弱。

到了近十几年时间，又是深度学习之父 Geoffrey Hinton 站出来，和他的学生发表了文章，找到了一种能够解决反向传播过程中梯度消失导致训练缓慢，收敛时间长问题的方案：无监督预训练设置权值的初始值，让权值初始时处于可训练的范围。然后进行有监督的训练微调。再一次刺激了学术界对深度学习的关注。

其后提出的 ReLU 函数能够有效的替代 sigmoid 函数，抑制梯度下降问题。而且在 2011 年后，各大公司的研究人员，尝试在多个领域问题中使用神经网络进行研究，最终在包括语音识别，CV，NLP 等多个传统计算机方案没有良好效果的领域取得了惊人的效果。直至如今深度学习方案已经成为这些领域的至关重要技术。

神经网络的发展也体现了深度学习发展的一波三折，每一次的沉积之后都有一次厚积薄发的突破。正是有像 Geoffrey Hinton 这样深耕在领域内从不放弃的先驱者，才有了今天深度学习领域的喷薄产出。

1.2. 神经网络在图像识别领域的发展

图像识别是一个发展历史很悠久方面，在早期，图像识别领域依赖于数字信号处理的发展，早期图像的质量和采集设备较差，对任务的影响很大，诞生了很多预处理算法，比如图像增强算法，手工的边缘检测算子等等，这一系列的方法都可以统称为数字图像处理以及手工特征提取。这一系列的知识对现在来说已经过时了，手工设计的各种方法针对性强，泛化性差，基于人的先验知识设计，在很多领域效果不好。但深远的影响到了现在很多深度学习图像处理，例如卷积器和池化器。

在深度学习兴起的时代，图像识别不再是需要先验知识专家处理的问题了，通过特征提取和分类器的联合训练，用数据自驱动的训练出最适合相关任务的特征提取模型。同时分类器也能最好的适配特征的相性。图像识别已经是深度学习一家独大的年代，当然这也带了一些弊端，相比于之前的处理方法，深度学习在处理这方面问题是要更加简答粗暴，缺少理论和解释性。技术总是在不断地自我迭代当中，深度学习用于图像识别确实已经非常的成熟而且做到了工业化。但是

想要取得进一步突破可能还得回过头看看。

2. 神经网络和训练

2.1 多层感知机

2.1.1 前向传播

多层感知机是最早期的神经网络，包含有输入层、输出层、以及中间的多个隐层，下图是包含一个隐层的多层感知机，一共是三层结构。

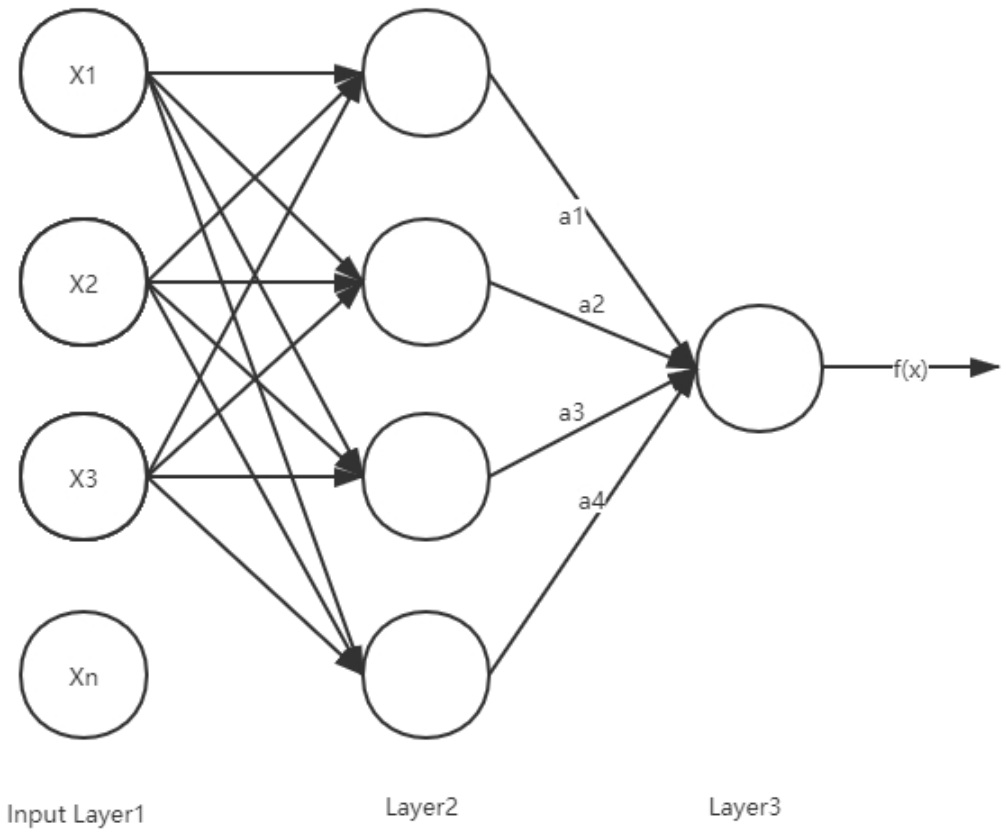

图 2.1 多层感知机

多层感知机的层之间采用全连接方式，我们用角标表示不同层，下标表示同层内的不同神经元，输入是$x_j^{(i)}$，输出是$y_j^{(i)}$，参数是$w_j^{(i)}$，常数项$b^{(i)}$。

对单个神经元内的计算公式如下：

$$y_j^{(i)} = w_j^{(i)^{\mathrm{T}}} x_j^{(i)} + b^{(i)} \tag{2-1}$$

求得单个神经元的输出，对多个神经元的计算采用矩阵化计算，充分利用计算单元针对矩阵的计算加速，以下是对多个神经元的计算的矩阵形式：

$$Z^{(i)} = W^{(i)} X + b^{(i)} \tag{2-2}$$

2.1.2 激活函数

多层感知机计算单元直观简洁，其实际数学意义是拟合各种线性函数，由于多层之间连接都是线性计算，因此不管隐层有多少层，都无法完全的拟合非线性函数。因此针对这一弱点，提出了使用非线性的激活函数来进行映射，激活函数一般是无法连续可导，通过在神经元的输出之后插入激活函数，多层感知机拥有了拟合非线性函数的能力。最初的激活函数是Sigmoid 函数：

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2-3}$$

其函数曲线如下：

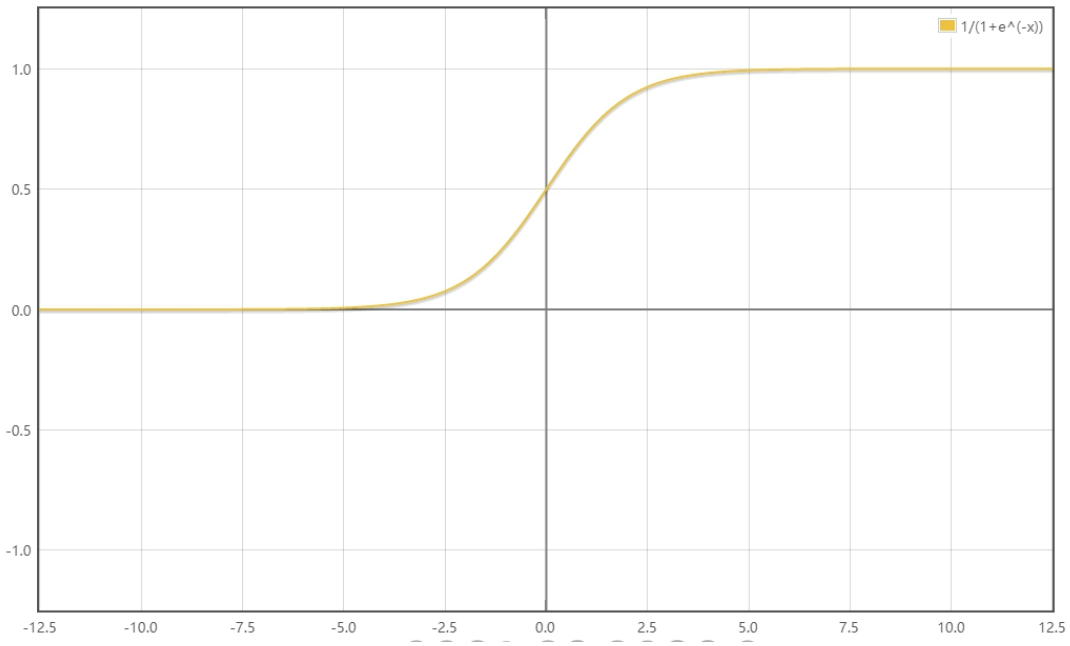

图 2.2 sigmoid 函数

sigmoid 函数也叫 Logistic 函数，它将一个实数的输入映射到(0,1)之间的任意值，可以证明sigmoid 函数是非线性函数，在 x=0 时不可求导。而且 sigmoid 函数的导数值为：

$$\sigma'(x)=\sigma(x)(1-\sigma(x)) \tag{2-4}$$

其值可以由前馈值简单的计算出来，这些特性使得sigmoid 被用于做激活函数。

但是sigmoid 函数具有缺点：

- 计算涉及除法，对计算机来说除法计算成本高；
- 在函数的大部分区间，导数值接近于 0，导致反向传播梯度时梯度消失；

尽管如此，这也不能抹灭sigmoid 函数对神经网络发展带来的深远影响。

2.1.3 反向传播

介绍了前向传播之后，重要的就是神经网络的训练过程——反向传播，也叫做 BP 算法。通过前向传播可以计算出一个预测值，而预测值与实际样本之间可以计算出误差值来，误差值越大我们的模型离正确的模型就越远，需要调整模型中各个参数。因此我们需要合适方式来计算出误差值，再使用梯度下降算法改变参数，由于这个过程和前向传播的计算刚好是一个逆过程，因此被称为反向传播，从最后的层修正参数，直到梯度计算传播到第一层，一就完成了一轮训练。

误差损失函数以及交叉熵函数：

$$\begin{aligned} J(W^{(i)},b^{(i)}) &= \frac{1}{m}\sum_{i=1}^{m}\zeta(\hat{y},y), \\ \zeta(\hat{y},y) &= -y\log\hat{y}-(1-y)\log(1-\hat{y}) \end{aligned} \tag{2-5}$$

通过对误差损失函数的求导计算得到：

$$
\begin{aligned}
&dZ^{(i)} = a^{(i)} - y, \\
&dW^{(i)} = dZ^{(i)} a^{(i-1)^T}, \\
&db^{(i)} = dZ^{(i)}
\end{aligned}
\tag{2-5}
$$

得到W,b梯度值，反过来更新原来的W,b：

$$
\begin{aligned}
&W^{(i)} = W^{(i)} - dW^{(i)}, \\
&b^{(i)} = b^{(i)} - \mathrm{d}b^{(i)}
\end{aligned}
\tag{2-6}
$$

2.1.4 学习率

这一个更新权重和偏移量的过程就是反向传播计算，权重和偏移量的改变方向朝着使误差损失值为 0 的趋势。如果把误差损失函数绘制成为在一个超维度空间中，那么我们权重和偏移量的改变就是为了让误差值走向最小值，但是由于部分样本的梯度下降方向不能代表着全体样本空间的下降方向，而在计算机训练的时候我们也无法一次使用全体样本空间的值进行一轮训练（数据有时候大到无法一次性存入内存或者是显存），所以我们要按一定比率采样的去更新权重和偏差值，这个比率就是学习率。如果学习率过大（接近 1），那么每一轮训练后，权重和偏移量会有大幅度的改变，可能会在误差损失韩函数超维度空间的最小值附近大幅度的摆动。学习率很小的情况将会非常的接近真实的最小值，但也永远不能达到。过小的学习率将会导致梯度下降的异常缓慢，学习效率很低。

$$
\begin{aligned}
&W^{(i)} = W^{(i)} - \alpha dW^{(i)}, \\
&b^{(i)} = b^{(i)} - \alpha \mathrm{d}b^{(i)}
\end{aligned}
\tag{2-7}
$$

2.2. 优化激活函数

前面提到激活函数，激活函数在神经网络中不可或缺，并且激活函数是影响神经网络训练和计算性能的关键因素。由于sigmoid函数具有一些影响很大的缺点，出现了类似tanh函数以及ReLU函数，其优越的性能使得它们几乎完全替代了sigmoid函数。

2.2.1 tanh 函数

tanh 函数又叫做双曲正切函数：

$$\tanh(x)=\frac{\sinh(x)}{\cosh(y)}=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \tag{2-8}$$

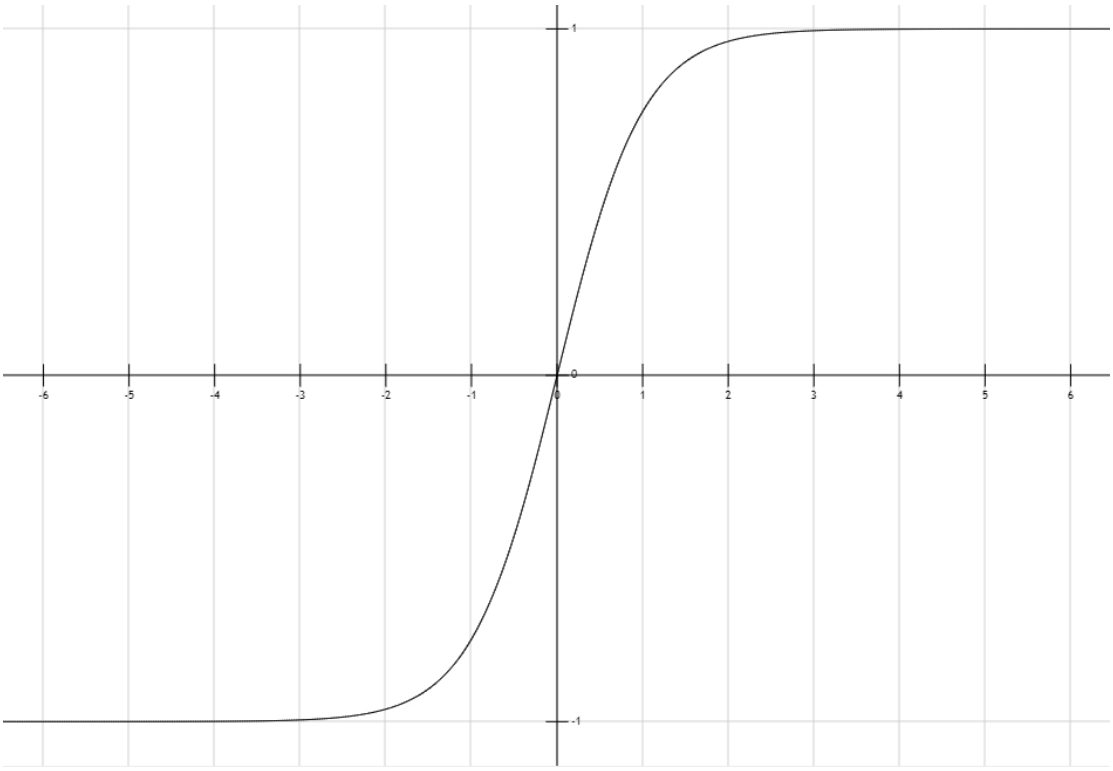

图 2.3 tanh 函数

tanh 函数是非线性函数，值域在(-1,1)。可以将任意实数映射到(-1,1)范围内。相比 sigmoid 具有以下优势：

- 可以从函数图像看出来，tanh 函数更加的陡峭，因此收敛速度更快；
- 最值相差更大，分类输出时对比更加的明显。

但是 tanh 函数依然在很大的范围存在梯度消失问题，因此没有彻底解决问题。

2.2.2 ReLU 函数

ReLU 是当前最热门的激活函数，又被称为修正线性单元：

$$f(x)=\max(0,x) \tag{2-9}$$

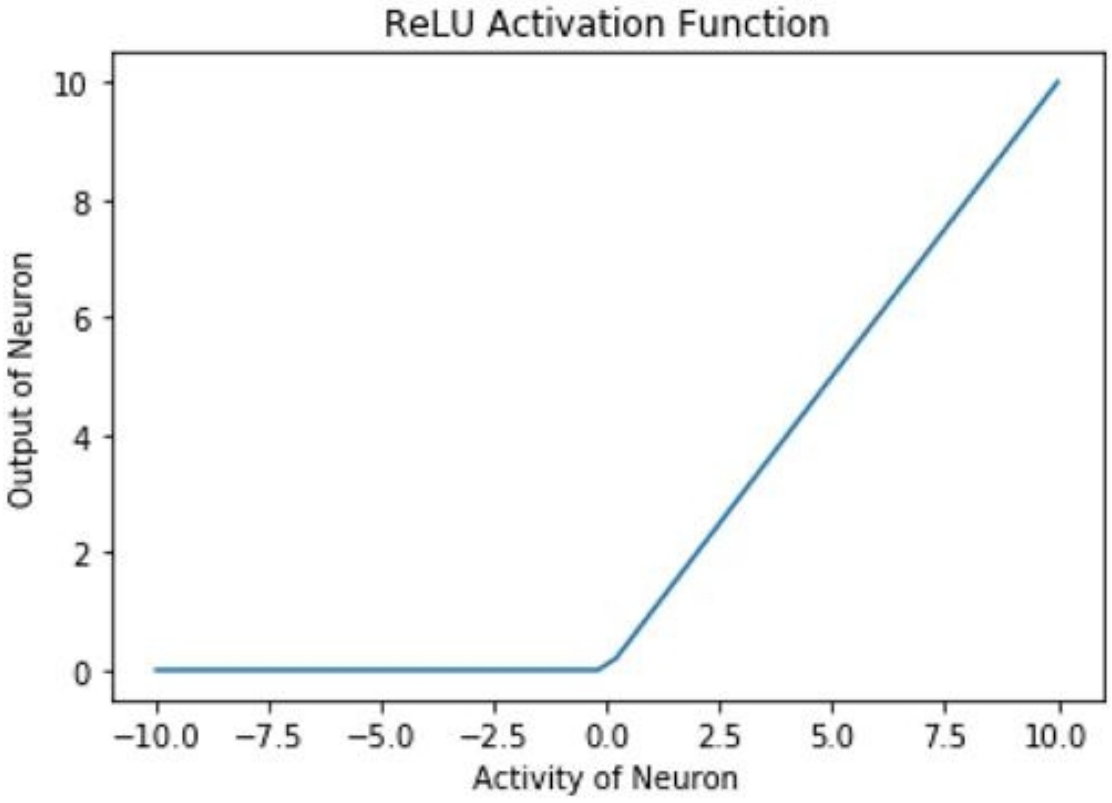

图 2.4 ReLU 函数

ReLU 是一个分段函数，在 x=0 处不可导，因此是一个非线性函数。ReLU 解决了 sigmoid 函数的各个缺点：

- 函数在 x>0 时，导数值恒定为 1，不存在梯度消失问题；
- 函数求导简单；
- 增大神经网络稀疏性。由于 ReLU 在输入负数时输出 0，这相当于将某个神经元关闭，不会对最后的结果带来任何的影响或是提升。所以说网络稀疏度提高了，提高网络稀疏度，并且是随机的关闭某一些神经元，能够提升网络的泛化能力，降低网络结构之间的依赖关系。

ReLU 经过广泛的验证，被证实性能十分的优秀，能够显著提高反向传播的速度。

同样 ReLU 在使用的过程中也发现了一些以前不曾关注的缺点：

- 神经元坏死。前面优点提到了网络稀疏度，这一可以从负面来理解，网络稀疏度提高也意味着网络的容量降低，拟合能力下降。

- 中心值不为 0。ReLU 函数值无负值，是非零化的输出，对下一层的神经网络带了偏置偏移，将会影响到下一层的梯度下降效率。

2.3 误差损失函数的选择

误差损失函数对反向传播计算梯度下降有很大的影响。选择误差损失函数应该具有计算较简单，求导方便，梯度变化明显，梯度具有稳定连续性。

2.3.1 均方误差函数

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{2-10}$$

下面是均方根误差函数值的曲线分布图，最小值即为目标值的位置：

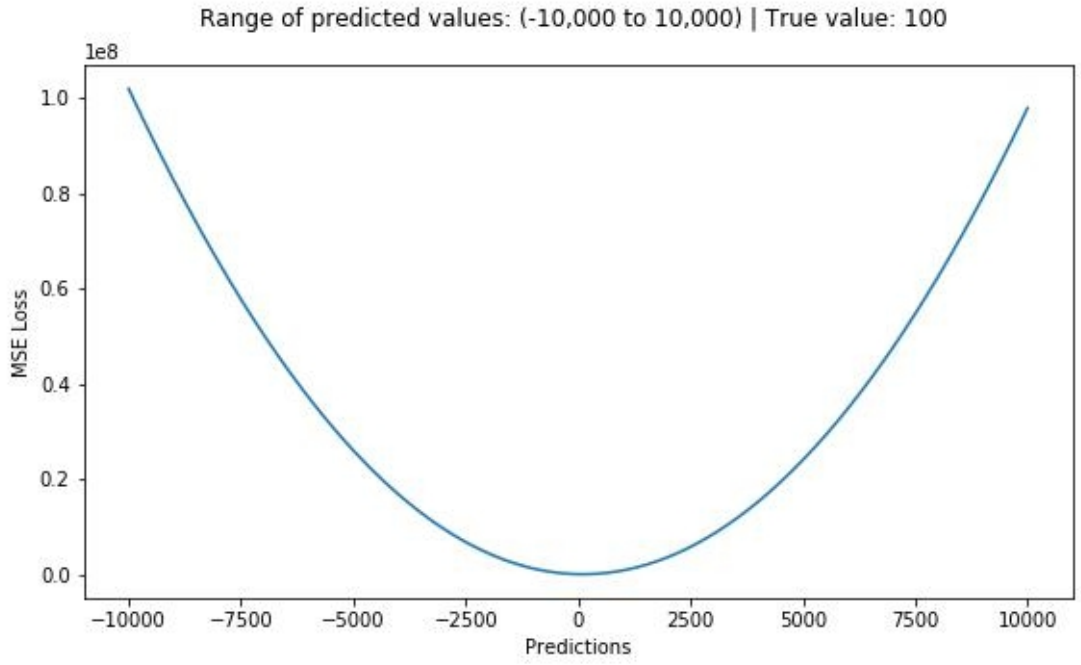

图 2.5 MSE 函数

均方绝对误差函数具有优点：函数曲线连续可导，具有稳定的解。

缺点：函数值两端梯度变化剧烈，使用梯度下降算法求解时会出现梯度爆炸。

2.3.2 平均绝对误差

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{2-11}$$

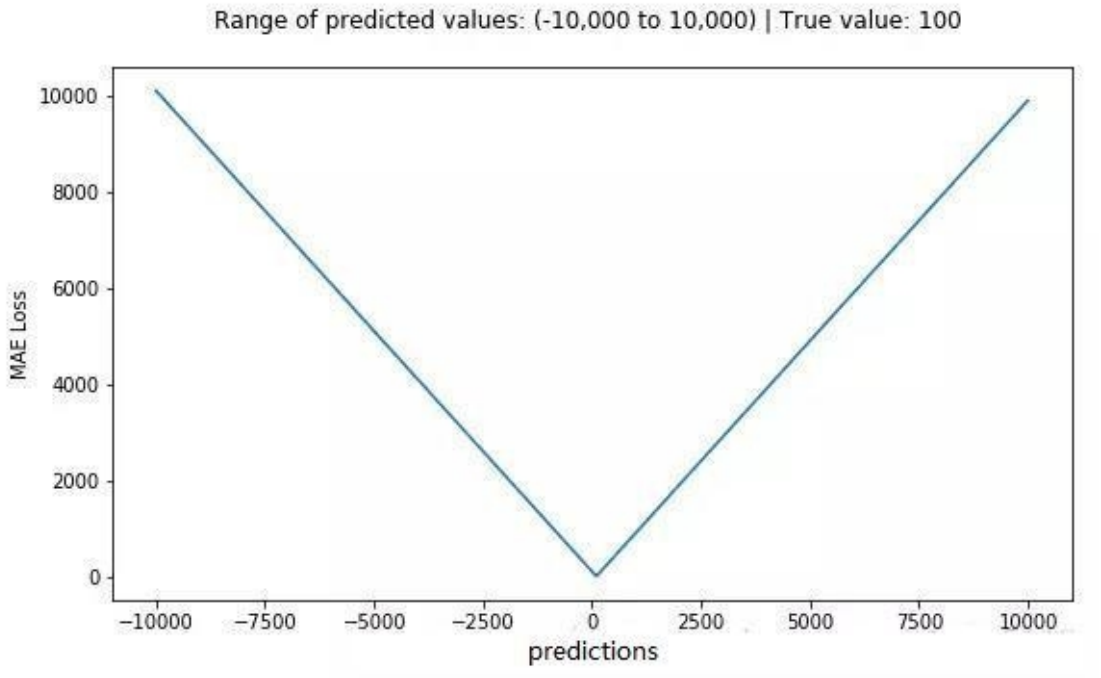

图 2.6 MAE 函数

平均绝对误差是一种不太常用的回归损失函数，它计算的是目标值和预测值之差绝对值的和，计算的值表示的是预测值平均误差振幅，但是平均绝对误差丢失了误差的方向信息。

平均绝对误差函数优点是：梯度稳定存在并且没有梯度爆炸的风险。

缺点：函数曲线并不连续可导，求导不方便。

2.3.3 smooth L1 误差损失函数

前面提到的平均绝对误差函数的缺点是不连续可导，而 smooth L1 函数就解决了这个缺点，在拐角处变得连续：

$$smooth_{L1}(x)=\begin{bmatrix} 0.5x^2, if\left|x\right|<1 \\ \left|x\right|-0.5, otherwire \end{bmatrix} \tag{2-12}$$

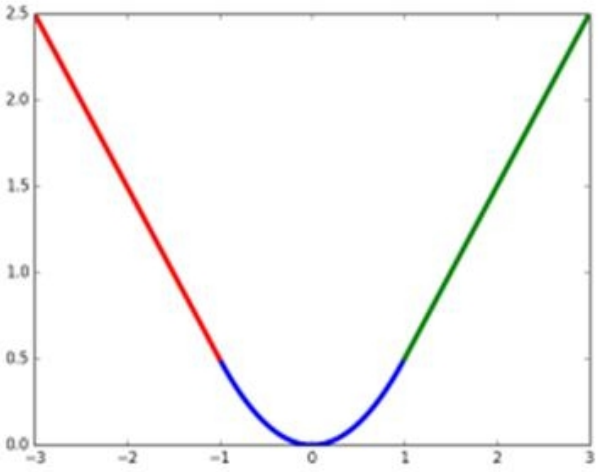

图 2.7 Smooth L1 函数

Smooth L1 针对上面提到两种误差损失函数的优缺点进行了结合。整体连续可导，梯度下降值大小可控不易出现梯度爆炸。

2.3.4 Cross entropy 误差损失函数

Cross entropy 又称交叉熵损失函数，是一种非常优秀的损失函数。主要用到信息论中的熵的概念：现有有关样本集合的真实概率分布 p，以及非真实分布 q。按照真实分布 p 描述识别一个样本所需要的编码长度的期望值为：

$$H(\mathrm{p}) = -\sum_{k=1}^{N} p_k \log p_k \tag{2-13}$$

这就是真实分布所带来的全部信息量的期望值，称之为信息熵。

要描述真实分布 p 与非真实分布 q 之间的差异，我们用 KL 散度也就是相对熵来表述：

$$\mathrm{D}_{KL}(p \| q) = \sum_{k=1}^{n} p_k \log(\frac{p_k}{q_k}) \tag{2-14}$$

相对熵越小，代表着非真实分布 q 离真实分布 p 越接近。

KL 散度公式可以化简得到：

$$
\begin{aligned}
&\mathrm{D}_{KL}(p \,\|\, q) = \sum_{k=1}^{n} p_k \log(\frac{p_k}{q_k}) \\
&= \sum_{k=1}^{n} p_k \log p_k - \sum_{k=1}^{n} p_k \log q_k \\
&= -H(p) + [-\sum_{k=1}^{n} p_k \log q_k]
\end{aligned}
\tag{2-15}
$$

当我们对随即变量进行预测是，真实分布 p 是不变的，那么信息熵 $H(p)$ 也是固定的，需要最小化的部分是 DK 散度公式化简的另外一部分，称之为交叉熵：

$$
H(p,q) = -\sum_{k=1}^{N} p_k \log \mathrm{q}_k \tag{2-16}
$$

从信息论的角度分析了为什么使交叉熵函数取得最小值的训练方向，就是让预测的概率分布趋向真实分布的方向。

2.3.5 *Soft* max *Loss* 函数

Soft max *Loss* 误差损失函数是交叉熵损失函数的一种特殊情况，交叉熵函数计算的是整个样本的输出空间的分布期望，如果真实分布的结果是只有一种可能，举例子来说一件事情的发生结果只有一个结果，非常确定，那么 p_k 中只会有一个的值为 1，其他为 0。那么交叉熵公式就可以化为：

$$
\begin{aligned}
&H(p,q) = -\log \mathrm{q}_k \\
&= -\log \frac{e^{q_k}}{\sum_{j}^{N} e^{q_j}} \\
&= -q_k + \log \sum_{j}^{N} e^{q_j}
\end{aligned}
\tag{2-17}
$$

这里 $\frac{e^{q_k}}{\sum_{j}^{N} e^{q_j}}$ 预测概率空间中，q_k 就是预测正确的概率，q_i 即是非真实空间预

测的所有可能概率。

Soft max *Loss* 误差损失函数的特性使得它用来做多分类识别任务的最后输出层。

2.4 梯度下降方法的选择

在反向传播当中，逐层计算求得梯度，下一步是利用梯度值，按一定的方式计算修正参数 w,b 。这个计算的方式对训练的结果非常重要，直接影响到能否使损失值下降到最小值，还会影响到下降的速度。将损失函数置于全体可能空间中一般是一个超维的解空间，简化为三维空间可以表示如下图：

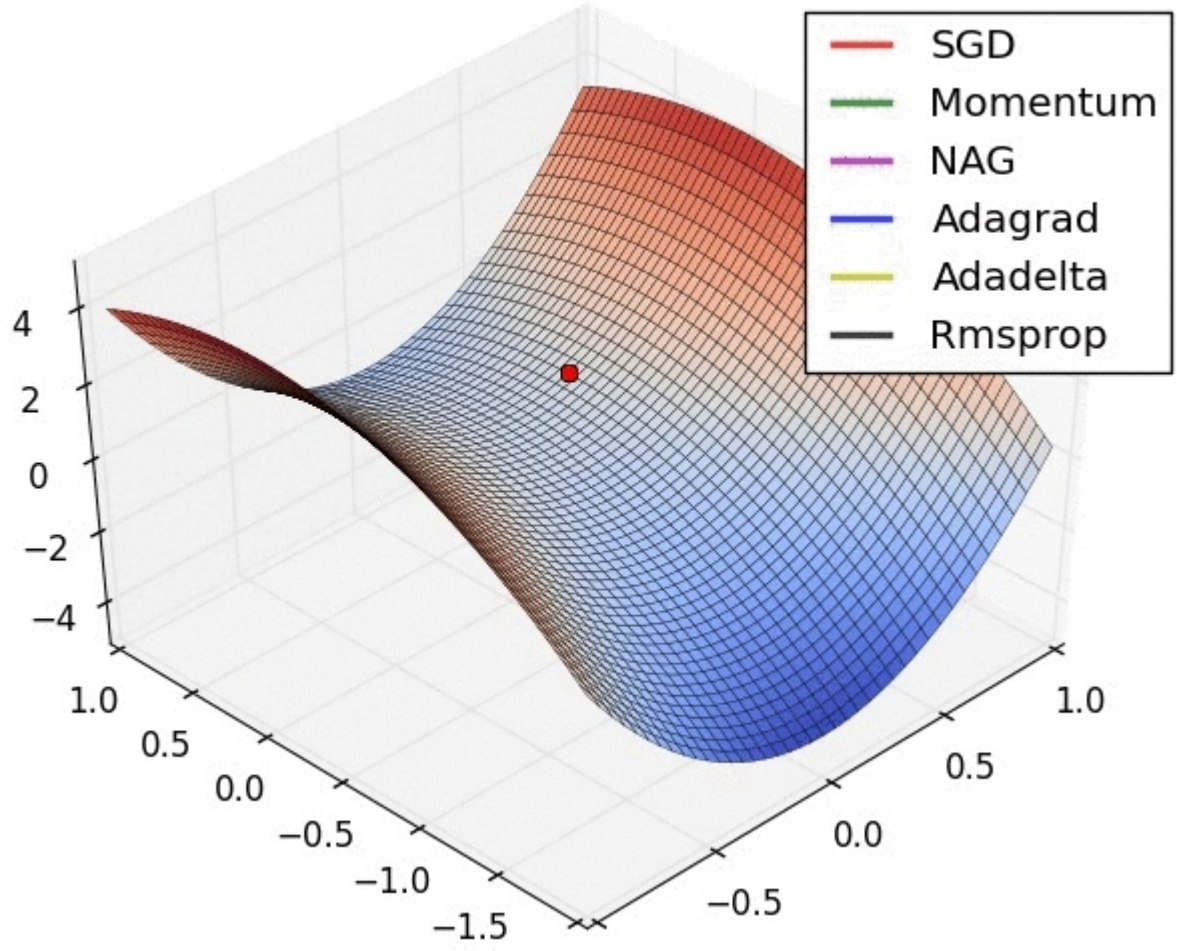

图 2.8 梯度下降马鞍面

梯度下降法的数学意义就是，梯度的方向是函数增长最快的方向。因此反方向就是函数下降最快的方向。这里约定待学习的模型隐层函数为：$f(x)=WX+b$，这里 W 是一个向量，$J(W)$ 损失计算函数，计算损失函数得到的梯度为 $\nabla J(W)$，学

习率为η。

2.4.1 随机梯度下降算法（SGD）

给定函数$f(x,y)=\frac{(x^2+y^2)}{2}$，求得梯度，$\nabla f(x,y)=(x,y)$。

给定随机初始点$(x^{(0)},y^{(0)})$，通过梯度得到迭代公式：

$$\begin{aligned} x^{(i+1)} &= x^{(i)}-\eta\frac{\partial f}{\partial x}(x^{(i)},y^{(i)})=(1-\eta)x^{(i)}, \\ y^{(i+1)} &= y^{(i)}-\eta\frac{\partial f}{\partial y}(x^{(i)},y^{(i)})=(1-\eta)y^{(i)} \end{aligned} \tag{2-18}$$

求极限可得(0,0)，因此求得函数的最小值是(0,0)。

随机梯度下降算法是很常规的，一次取一个样本计算得到梯度，减去梯度和学习率的乘积。由于单个样本的下降方向不能代表真实整体的下降方向，也就是存在噪声，梯度的下降方向有时候是真实放下降方向，有时候甚至是相反方向，但由于样本的整体能够反映出大致下降方向，所以在样本足够，训练时间够长的时候，总能够到达最低点，因此称为随机梯度下降算法。

随机梯度下降算法由于对样本中噪声的反馈很大，有改进 SGD。

基于 Mini-batch 的随机梯度下降算法。一次取一个小批量样本，求得的损失函数梯度值是各个样本的平均值，这样得到的下降方向不容易受单个样本噪声的影响。

3. 卷积神经网络

3.1 卷积层

卷积是一种数学计算，在数学中应该叫做互相关函数。常常用于提取相关特征。在神经网络中卷积核是卷积神经网络的核心工具，也是卷积计算的参与者。

卷积核在图片特征提取当中常常是作为边缘滤波器，对图片中关键特征提取，去除冗余信息，能够减少数据矩阵的尺寸。

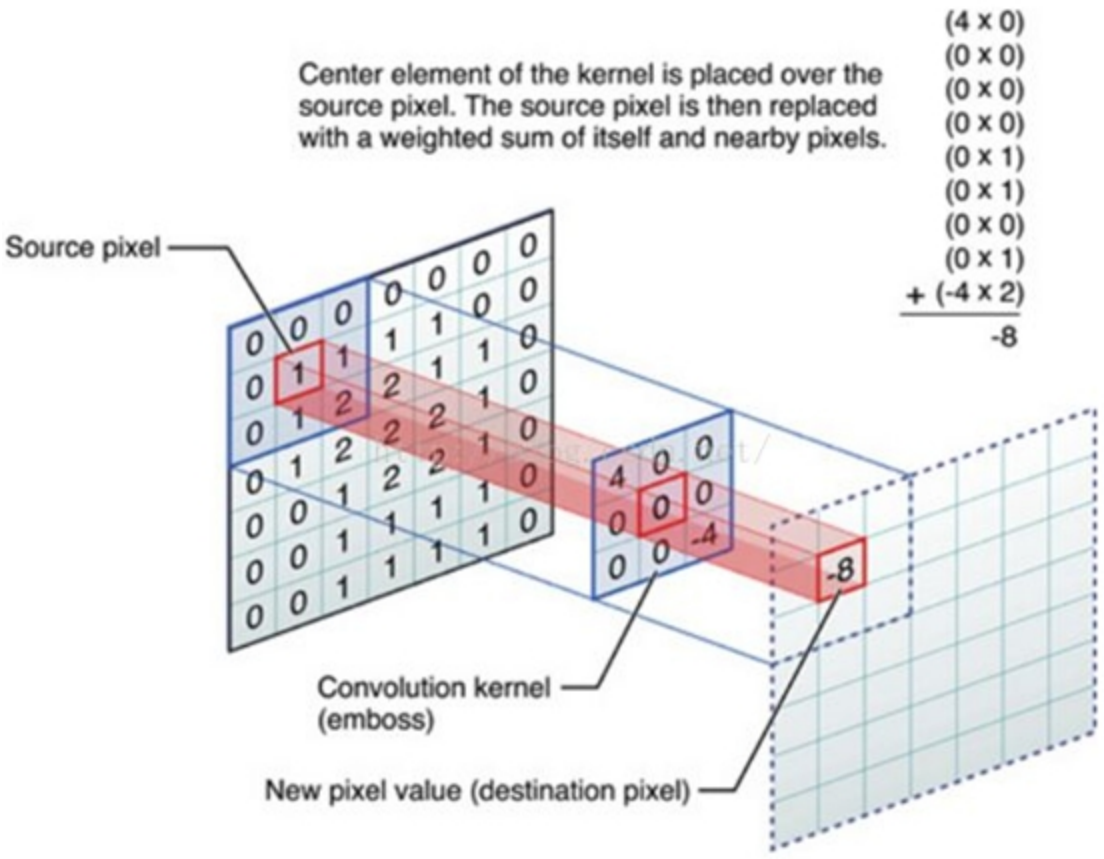

图 3.1 卷积计算

卷积层在神经网络中的作用：

（一） 局部连接：

局部连接，卷积层的输出都是输入经过卷积计算得到，只有部分的节点和前一层之间有联系，因此大量减弱了节点的耦合性。

图像处理的特点是，特征往往是相互靠近的像素点组成的，相距比较远的像素点可能关系比较弱。特征与特征之间可能存在关联，但是也有很多不存在关联性，提取某一特征的卷积层节点应该有关联，而不同特征之间的应该尽量保持独立，减弱相关，那么最好的方式就是部分连接。

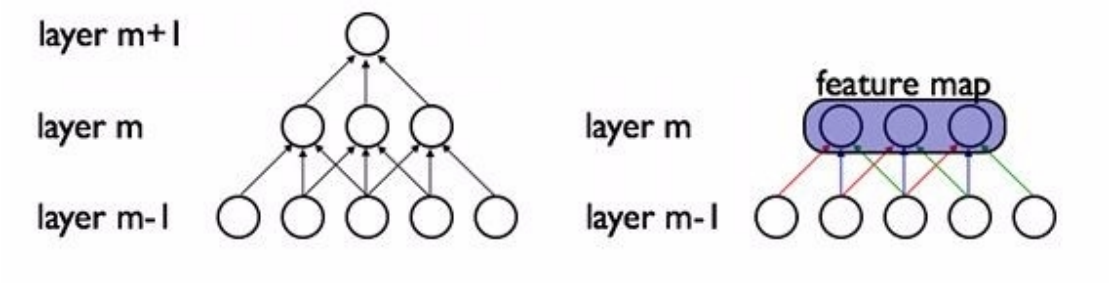

图 3.2 局部连接

（二） 参数共享

参数共享也叫做权值共享，在全连接层中，每两个连接的上下层节点都有独立的参数，而在卷积层中，训练得到参数仅仅是卷积核，而一个卷积核的大小是远远节点间参数。

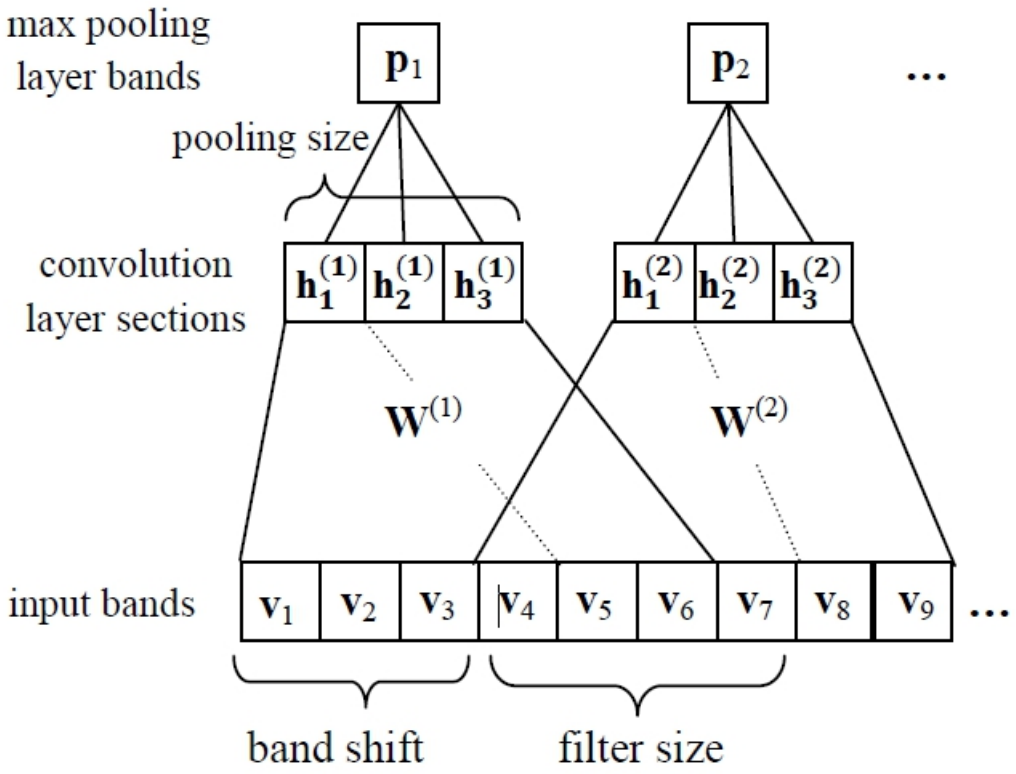

图 3.3 参数共享

3.2 池化层

池化层常常应用在连续卷积层之中，其作用是压缩数据尺寸减少需要训练参数的数量，能够减少过拟合。在图像处理当中，池化层的最直观的作用是可以起到压缩图像的作用。

特征不变性：图像处理当中对图片尺寸的变化是不敏感的，当图片下采样时，图片最基本的特征依然得到了保留。在人脑处理图像时，不会因为图片变模糊或是尺寸变小了一半而导致不认识图片了。去掉了很多的重复性特征，不会严重影响到对图片的识别准确率，池化层在神经网络的中作用类似。

特征降维：图片经过卷积核提取到特征后，相邻的输出中很可能检测到了相同的特征，还有可能提取到了更深但是无意义的特征，池化层通过下采样，去掉了重复特征和无意义的特征提取。

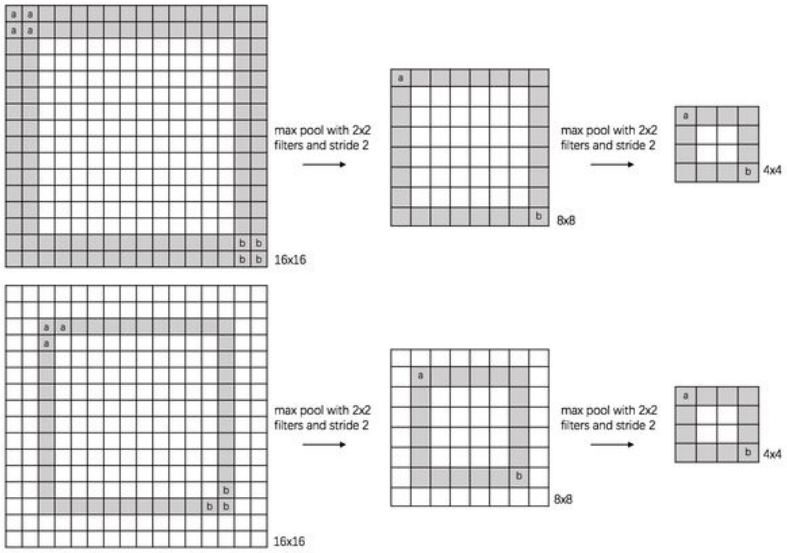

图 3.4 池化层

3.3 LeNet

LeNet 最早诞生于 1994 年，应用于手写数字识别。是最早期的有实用价值的卷积神经网络。LeNet 的诞生有其历史背景，当时计算机计算速度不足，无法达到现如今的水平，网络规模受到限制，但是网络规模小意味着网络的容纳量小。因此在必须提升网络规模前提下，通过采用卷积层以及池化层，减少了很多的全连接层。减少了计算的复杂性。从而使得网络在当时的硬件水平下依然可以训练。

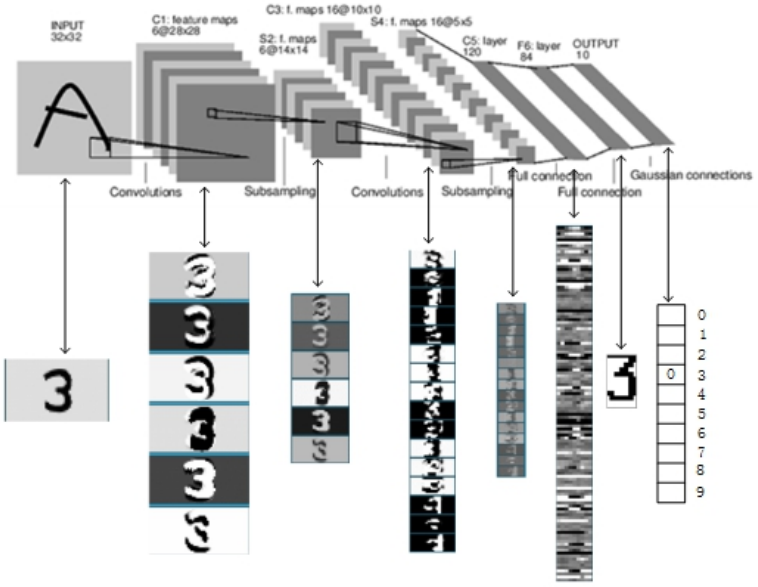

图 3.5 LeNet 网络结构

3.4 AlexNet

AlexNet 的网络模型，包含了 5 个卷积层，3 个全连接层，网络的总层数是 8 层。

网络分为相对比较独立的上下两部分，网络作者介绍道这两部分是分别存在与不同的两块 GPU 当中，通过双 GPU 的方式在当时有限的硬件条件下，又提高了网络能够设计的最大深度。充分利用了两块 GPU 的计算能力，因为当时单个 GPU 只能够存放下一半的网络模型参数。从 LeNet 到 AlexNet 可以看到硬件计算能力对网络的拟合能力有至关重要的限制。

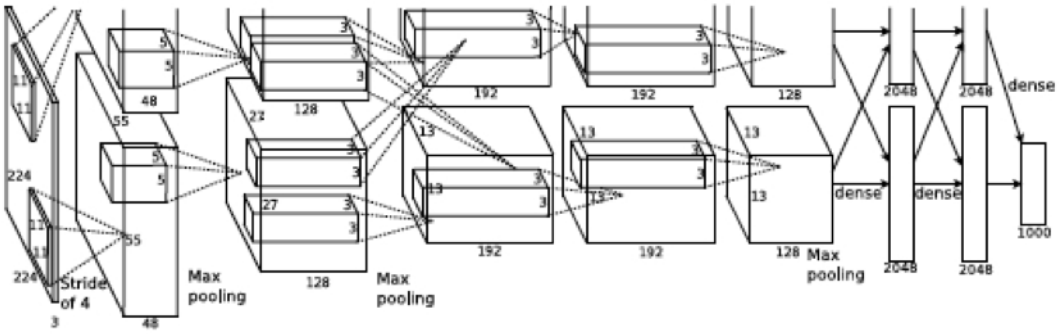

图 3.6 AlexNet

3.5 GoogLeNet

深度学习以及神经网络快速发展，硬件方面带来的提升固然很大，但是研究方向变得更加宽广，网络结构的设计变得越来越巧妙。

当时在 AlexNet 的基础上提出了 GoogLeNet 网络，其解决的问题是：在保持网络结构的稀疏性的同时，还能利用到硬件计算密集矩阵时的高速性能。

GoogLeNet 中突破性的提出了一种结构：Inception。在以往的网络中，某一层的操作是确定的，究竟是卷积层、池化层还是全连接层。但是这个操作的选择在不同的任务下效果是不一样的，因此网络的结构也应该能够适应性的去调整。在这种情况下，网络的结构也在一定程度下能够训练，Inception 能够满足上述的要求，究竟是选择什么样的卷积核，还是选择池化层。交给训练过程的自适应选择。

Inception 还使用了 1x1 卷积核，能够大幅度减少连接之间的参数值，经过 1x1

卷积核的计算，数据尺寸将减少很多。

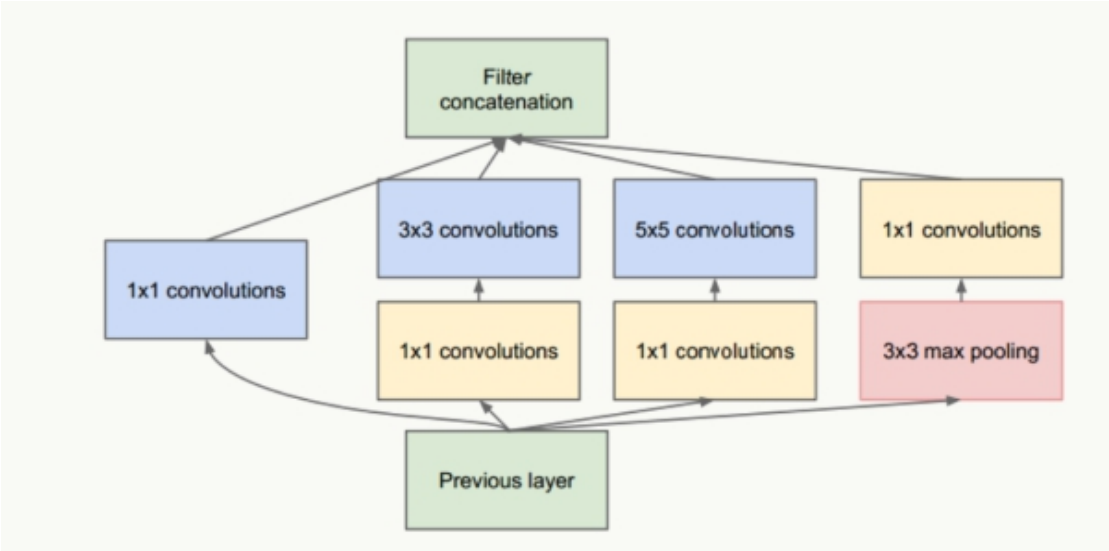

图 3.7 Inception

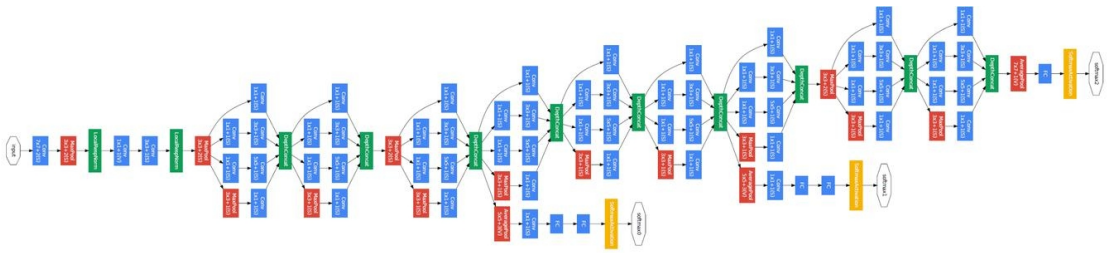

图 3.7 GoogLeNet

3.6 ResNet

ResNet 是当前应用最广泛的基于 CNN 的特征提取网络。随着网络层次的加深，更高层次的网络不能够提高分类性能，而且在深层的网络中会出现无法收敛的情况，测试数据集上也不能得到很好的训练效果。这一系列问题是由于网络过深，梯度传播过程很长，经过多层计算后，梯度渐渐消失，导致收敛异常缓慢。

若将假设输入设为 X，将网络模型中某层的参数设为 H，则以 X 为输入的此层的输出将为 H(X)。常规的网络学习到的就是从 X -> H(X)的映射关系。

而残差学习则于是使用多个不同的带参网络层学习，输出网络层之间的残差。残差块的公式表达为：H(X)+F(X,W)。

残差块分成两部分直接映射部分和残差部分。 H(X)是直接映射，F(X,W)是残差部分，残差部分一般由卷积操作计算得到。残差块引入 Identity mapping 的

概念，在输入和输出建立了两条通道，一个是直接连接学习的直接映射，一个是多部卷积操作组成的残差。模型要学习是两部分之和。如果两部分的通道数不相同，残差块引入了 1x1 卷积核，补齐两者的差距。从而是的两部分的结果能够求和。

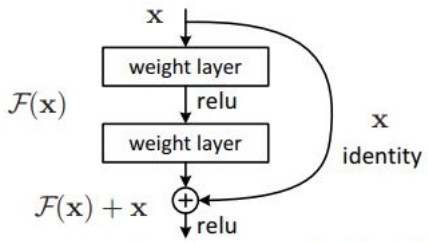

Figure 2. Residual learning: a building block.

图 3.8 残差块

4. 提升泛化能力

过拟合，是指模型在训练集上有很高的识别准确度，但是在测试验证集上效果较差，模型缺乏对未知数据的预测能力，这种情况就称为模型过拟合。下图 3.9 中，第一种是欠拟合状态，模型对训练数据拟合能力不足；第二种情况是理想的状态；最后的是过拟合状态，模型过于复杂，预测的结果虽然很正确，但是学习到了不必要的复杂特征，导致在未知数据上的预测正确率不如训练集。

图 3.9 不同拟合情况

4.1 早停（Early stopping）

Early stopping 便是一种迭代次数截断的方法来防止过拟合的方法，即在模型对训练数据集迭代收敛之前停止迭代来防止过拟合。训练随着轮次的加多，在训

练集上的正确率随之增加，但是由于样本不完全代表着真正要学习的真正模型，样本是存在误差和噪声的，因此不要求在训练集上达到100%的正确率，为了避免这种情况，可以在训练过程中随机的暂停训练，得到中间过程的模型。

4.2 正则化

指的是在目标函数后面添加一个正则化项，一般有L1正则化与L2正则化。L1正则是基于L1范数，即在目标函数后面加上参数的L1范数和项，即参数绝对值和与参数的积项。

$$C = C_0 + \frac{\lambda}{n}\sum_w |w| \tag{4-1}$$

L2正则是基于L2范数，即在目标函数后面加上参数的L2范数和项，即参数的平方和与参数的积项。

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2 \tag{4-2}$$

正则化对过拟合问题之所以有比较好的帮助，是因为：

（1）正则化是基于参数满足正态分布，而中心极限定理表明，大多数规律分布满足正态分布，因此正则化比较适合；

（2）将模型的参数限制在比较小的范围，模型参数越大，其模型也越复杂，拟合情况也就越特殊；

（3）正则化使得模型对输入参数的微小扰动不敏感，特例或者是噪声数据对模型干扰效果降低，因此模型抗过拟合能力得到提升。

（4）参数实际上模型的一阶导数值，代表了模型的平滑程度，而平滑的函数相较于复杂的函数更适合表达真正的样本解空间。

5. 狗分类任务

本文介绍在 kaggle 平台过往比赛 Dog Breed Identification 中采用上述各类深度学习方法，使用比赛提供的数据集，训练出一个网络模型。在提供的测试集上进行测试任务，最终提交到比赛平台，获得评分，得到对网络模型的评估。

5.1 任务介绍

5.1.1 数据集

需要分类的犬图片包含了 120 种不同类型的犬。平均每一类犬有 83 张图片。数据集总共包含了 1 万多张图片。数据集的质量清晰度比较高，因此即使每种犬图片数量比较少的情况下还是能够完成训练任务。

5.1.2 标签预处理

从官方下载到的数据集包含如下：

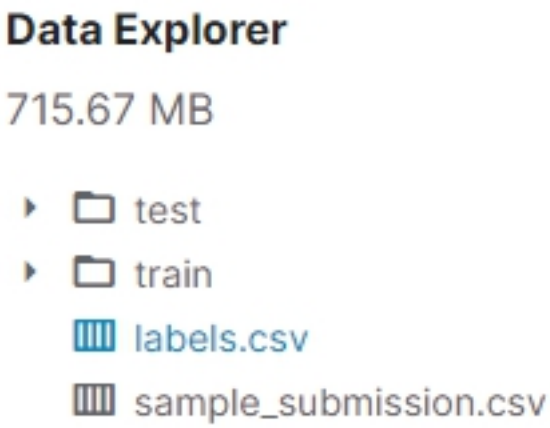

图 5.1 数据集

图片文件不包含标签，所有的图片文件名称与标签的映射包含在 labels.csv 文件中。为了方便模型加载和训练，需要根据标签对图片重新划分文件夹。通过给定随机数的方式，按照 2：8 的比例划分训练集和验证集。由于测试集上没有任何标签，因此测试集不能用于训练。

表 5.1 数据集划分

```
def divide_dataset():
    row_list = []
    with open('labels.csv', 'r') as f:
        reader = csv.reader(f)
```

```
        for row in reader:
            row_list.append(row)
    row_list = row_list[1:]
    type_list = set([i[1] for i in row_list])
    train_dir_list = ['./train/'+ i for i in type_list]
    val_dir_list = ['./val/'+ i for i in type_list]
    for each in train_dir_list:
        if os.path.exists(each) == False:
            os.makedirs(each)
    for each in val_dir_list:
        if os.path.exists(each) == False:
            os.makedirs(each)
    prefix = ['./train/', './val/']
    random.seed(0)
    f = lambda x:prefix[1] if x>0.8 else prefix[0]
     file_list = [['./train/' +i[0]+'.jpg',f(random.random())+i[1]+'/'+i[0]+'.jpg'] for i in
row_list]
    for each in file_list:
        shutil.copy(each[0],each[1])
```

通过上述脚本，完成了两部分工作，一是将相同标签犬图放置到同名文件夹下，文件夹名称就是犬种类的英文名称；二是将整个训练集划分为两部分，训练集和验证集，留有验证集以对模型的泛化能力进行评估。

5.2 网络选择

犬分类任务经过分析是常规的图片识别分类任务，因此采用特征提取网络提取特征，再使用全连接层进行分类，具体步骤如下：

1. 预训练模型提取到特征参数
2. 特征值传入全连接层进行分类识别

3. softmax 函数将结果映射到(1,120)的分类向量中

提取特征的网络选择有很多方案，按照尝试的原则，选择一下三种网络做对比分析。

- VGG16
- ResNet34
- ResNet50

特征提取网络都有相对固定的结构，采用固定的网络便于使用迁移学习的优点，充分利用已经提取好的特征。能够调整的部分是特征提取网络后的全连接层，设计结构如下：

表 5.2　网络结构表

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 8, 8, 1536)	0
global_average_pooling2d_1	((None, 1536)	0
dense_1 (Dense)	(None, 256)	393472
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 120)	30840

Total params: 424,312

Trainable params: 424,312

Non-trainable params: 0

5.3 搭建网络

5.3.1 读取数据集

由于数据集包含了上万张图片，占用内存比较大，不适合一次性全部读入，因此采用了 pytorch 中的数据加载器。按照 mini-batch 的设置值，在训练时再读入内存或是显存当中。

表 5.3　数据集加载器

```
datasetsdir = 'datasets'
traindir = os.path.join(datasetsdir, 'train')
valdir = os.path.join(datasetsdir, 'val')
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],std=[0.229, 0.224, 0.2
25])
train_loader = torch.utils.data.DataLoader(
   datasets.ImageFolder(traindir, transforms.Compose([
      transforms.RandomResizedCrop(224),
      transforms.RandomHorizontalFlip(),
      transforms.RandomRotation((-30, 30)),
      transforms.ToTensor(),
      normalize,
   ])),
   batch_size=batch_size, shuffle=True,
   num_workers=num_workers)
```

采用 pytorch 的数据加载器的同时，开启了对图片的数据增强参数。随机的对输入的图片进行尺寸放缩、裁剪、水平反转、旋转操作。并且对数据集的输入值进行正则化，这里的正则化与网络参数的正则化不同。图片输入参数为(C,W,H)。第一个值是代表颜色通道，取值范围是(0,2)。后两位代表了图片的宽高，取值范围是 rgb 色彩范围(0,255)，这样的取值范围太大，对模型参数的影响很大，容易导致模型更加复杂，因此开启正则化，将输入值映射到(0，1)之间，重新按正态分布进行约束。

5.3.2 vgg16 网络

pytorch 自带有 vgg16 特征提取模型，自带的特征提取器训练自 google 的 imagNet，是一个规模非常庞大的图片样本标签集。特征提取性能优越，由于我们每一种犬图片数量较少，如果重新训练特征提取器会有样本数量不够的问题，所以直接迁移学习使用成熟的提取器能够显著提高提取的能力。

我们关闭网络特征提取器部分的训练开关。这样在训练过程中，特征提取器部分的网络参数不会发生计算改变，因此网络需要训练计算的参数少了很多，便于提高训练速度。在特征提取器的输出部分连接一个全连接层，输出尺寸为 120，正好对应了犬图片的 120 歌类别。

表 5.4　三种网络模型

```
def vgg16(pretrained=True):
    model = models.vgg16(pretrained=pretrained)
    def set_untrainable(layer):
        for p in layer.parameters():
            p.requires_grad = False

    for layer in model.children():
        layer.apply(set_untrainable)
    model.classifier = nn.Sequential(*(list(model.classifier.children()) + [nn.ReLU(inpl
ace=True), nn.Linear(1000, 120)]))
    # model.features = torch.nn.DataParallel(model.features)
    model.cuda()
    return model
def resnet34(pretrained=True):
    model = models.resnet34(pretrained=pretrained)
    def set_untrainable(layer):
        for p in layer.parameters():
            p.requires_grad = False

    for layer in model.children():
```

```
        layer.apply(set_untrainable)
    model.fc = nn.Linear(512, 120)
    model.cuda()
    return model
def resnet50(pretrained=True):
    model = models.resnet50(pretrained=pretrained)
    def set_untrainable(layer):
        for p in layer.parameters():
            p.requires_grad = False

    for layer in model.children():
        layer.apply(set_untrainable)
    model.fc = nn.Sequential(model.fc, nn.Dropout(p=.5), nn.ReLU(inplace=True), nn.
Linear(1000, 120))
    model.cuda()
    return model
```

这里三种网络的全连接层结构后部分相似，不同的是特征提取部分，vgg16 只包含了 16 层，resnet32 包含了 32 层，resnet50 包含了 50 层。网络深度逐步加大。小的网络能够容纳的信息量虽然不如较大的网络，但是 vgg16 计算步骤少很多，训练迭代速度快，调整网络后再次训练能够及时得到训练结果和泛化情况，便于对网络的超参数进行调整，因此也有必要使用层次较浅的网络。

Resnet50 采用了 dropout 层，随机的对某些节点输入进行关闭。这是因为数据集太小的缘故，在小数据集上反复多轮次训练很容易导致过拟合，对训练集严重依赖。因此使用 dropout 减轻网络节点间的关联性，从而减少过拟合。

5.4 学习率与梯度下降

随着网络的搭建完毕，后续训练设置了误差损失函数，这里使用的是交叉熵误差损失函数。梯度下降函数选择带有动量的 Adam 函数，Adam 函数能够综合梯度的历史下降方向，而不是像随机梯度下降方法，每次下降的方向和大小与前

一次计算的结果无关，这样下降方向能够比较稳定。而带有一定的动量的好处是避免了落入局部最优的情况，使训练能够主动跳出一些局部最优解，更快的找到使损失值为最小值的参数。

学习率选择了 0.001，虽然没有采用随训练轮次变化衰减的学习率，但这是一个比较小的学习率，在训练的全阶段都比较慢速，但是能够配合 Adam 梯度西下降算法能够在最后收敛时有更精准的训练结果。

表 5.4　resnet-50 优化策略

```
model = resnet50()
loss_fn = nn.CrossEntropyLoss().cuda()
optimizer = optim.Adam([p for p in model.parameters() if p.requires_grad], lr=1e-3)
train(model, train_loader, loss_fn, optimizer, num_epochs=3)
check_accuracy(model, val_loader)
```

5.5 前向与反向传播

在 pytorch 框架下，只需要简单的调用 api 即可完成前向和反向传播计算，需要自己处理的是决定训练的循环轮次，以及过程中产生量的打印记录。

表 5.5　resnet-50 训练方案

```
def train(model, loader, loss_fn, optimizer, num_epochs = 1):
    start_time = time.time()
    for epoch in range(num_epochs):
        print('Starting epoch %d / %d' % (epoch + 1, num_epochs))
        model.train()
        batch_start = time.time()
        for t, (x, y) in enumerate(loader):
            x_var = x.cuda()
            y_var = y.cuda()

            scores = model(x_var)
```

```
        loss = loss_fn(scores, y_var)
        if (t + 1) % 100 == 0:
            print('t = %d, loss = %.4f, duration = %s' % (t + 1, loss.item(), timedelta(sec
onds=time.time() - batch_start)))
            batch_start = time.time()

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    print('Duration = %s' % timedelta(seconds=time.time() - start_time))
```

5.5.1 损失值和准确度

调用 train()函数，进行多轮次的前向与后向梯度下降计算。得到最终的模型，以及训练的效果如下表：

表 5.6 resnet-50 训练结果

```
Train on 8221 samples, validate on 2001 samples
Epoch 1/1
8221/8221 [==============================] - 6s 675us/step - loss:
0.4873 - acc: 0.8640 - val_loss: 0.3043 - val_acc: 0.9110
```

训练集有 8221 张图片，验证集有 2001 张图片。最终训练集上的准确度是 86.4%，验证集上的准确度是 91.1%。从结果可以看到验证集上准确率高于训练集，说明了泛化性能比较好。

5.6 验证集效果

利用训练好的模型对部分部分犬类图片进行识别分类，并打印图片和标签如下图：

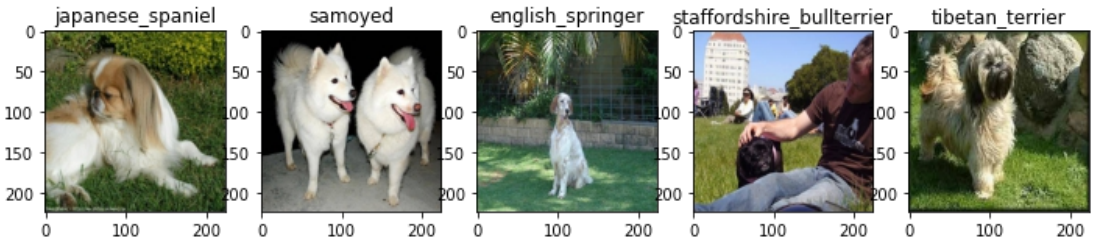

图 5.2 resnet-50 验证集部分图

Japanese spaniel 查找真实图片：

图 5.3 Japanese spaniel

English Springer 真实图片：

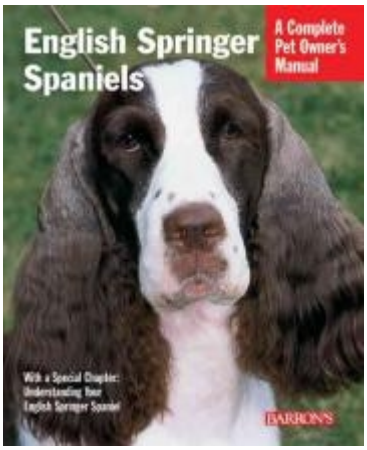

图 5.4 English Spring

Staffordshire bullterrier 真实图片：

图 5.5 Staffordshire bullterrier

tibetan terrier 真实图片：

图 5.6 tibetan terrier

可以发现，提取到的几张可视化后的预测图片都全部预测成功。

5.6 提交记录

将 test 集合所有数据经过模型计算一次得到预测结果，保存为 submission.csv 文件。提交到 kaggle 平台上进行评分得到结果如下表：

表 5.7 resnet-50 提交记录

Name	Submitted	Wait time	Execution time	Score
submission.csv	a minute ago	1 seconds	2 seconds	0.76703

以下是比赛历史排行榜部分排名：

表 5.8 resnet-50 排名

Rank	Team name	Score	Entries	Last
780	Xiuxiu Sun	0.76397	1	3y
781	BowenHe	0.76732	9	3y
782	michelia	0.77594	2	3y

通过对比可以得知本次预测成绩介于 781 名和 782 名之间，介于比赛已经结束，比赛结束后的所有提交的成绩不计入排行榜。

5.7 参照对比

除了使用 resnet50 训练提交之外，还设置了参照组：vgg16 和 resnet34，相同的

预训练过程，只替换使用的前部分网络。得到实验数据如下：

5.7.1 vgg16

替换预处理部分网络为 vgg16，vgg16 的 fc 层输出尺寸是 1000。由于训练下降效果随训练轮次影响不大，模型很早的就已经收敛，因此只训练了一轮，使用早停的方法，找到最佳的训练集效果。

表 5.9　vgg-16 训练结果

Starting epoch 1 / 1

t = 100, loss = 11.7730, duration = 0:00:14.883000

t = 200, loss = 11.0780, duration = 0:00:14.152997

t = 300, loss = 15.2570, duration = 0:00:13.270000

t = 400, loss = 17.4817, duration = 0:00:13.302001

t = 500, loss = 14.9861, duration = 0:00:13.384000

Duration = 0:02:50.576499

Checking accuracy on validation set

Got 1235 / 2038 correct (60.60)

提交 csv 文件到 kaggle 平台，在 test 数据集的验证得分如下：

表 5.10　vgg-16 得分

Name	Submitted	Wait time	Execution time	Score
sub-vgg16.csv	just now	1　seconds	2　seconds	3.07861

kaggle 相对接近的得分排名如下：

表 5.11　vgg-16 排名

Rank	Team name	Score	Entries	Last
956	Max Lawnboy	3.02639	4	3y
957	coder dr	3.03244	7	3y
958	lucadido	3.16053	15	3y

部分验证集预测结果可视化：

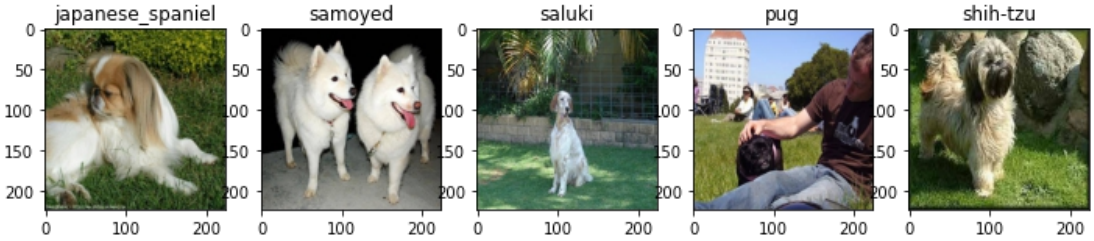

图 5.11 vgg-16 验证集部分图

与真实结果对比，预测结果只有前两张是正确的，效果不佳。

5.7.2 resnet34

调整替换 resnet34 网络模型，全连接层参数替换为 512，这是 resnet 网络 fc 层的输出尺寸。

训练三轮的过程情况 如下：

表 5.12 resnet-34 训练结果

Starting epoch 1 / 1

t = 100, loss = 20.3790, duration = 0:00:10.826066

t = 200, loss = 17.9577, duration = 0:00:09.458664

t = 300, loss = 12.7940, duration = 0:00:09.613440

t = 400, loss = 12.7710, duration = 0:00:09.517755

t = 500, loss = 13.2645, duration = 0:00:09.548642

Duration = 0:00:50.140260

Checking accuracy on validation set

Got 1110 / 2038 correct (54.47)

Duration = 0:00:20.802825

在 test 集上的得分是：

表 5.13 resnet-34 得分

Name	Submitted	Wait time	Execution time	Score
sub-resnet34.csv	just now	1 seconds	2 seconds	0.96003

成绩最后得分是 0.96003，比 resnet50 高，按照 kaggle 评分标准，resnet34 预测效果要比 resnet50 差不少。

Kaggle 近似排名如下：

表 5.14　resnet-34 排名

Rank	Team name	Score	Entries	Last
816	RobertH	0.95971	2	4y
817	doges	0.95976	29	3y
818	momo	0.98066	1	3y

查看部分图片可视化的预测情况：

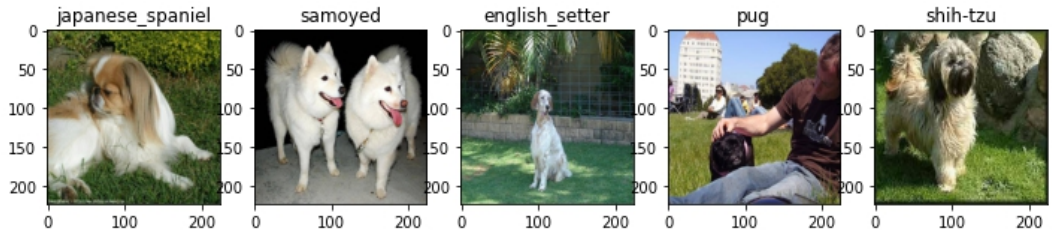

图 5.14 resnet-34 验证集部分图

前三张图片预测正确，但是后两张都预测错误。

5.8 对比结论

三种网络的预测结果对比如下：

表 5.15　三种网络效果对比

Model name	Score	Rank	View effect
Resnet-50	0.76703	712	5
Resnet-34	0.96003	818	3
Vgg-16	3.07861	958	2

Resnet-50 效果最佳，无论是从 Kaggle 得分排名还是实际的可视化预测结果，其次是网络结构相对较小的 Resnet-34，最差的是只有 16 层的 vgg-16。在其他条

件相同的情况下，预测结果的准确度能够一定程度网络的信息容纳量。很明显的可以得知，网络越复杂，能够学习到更多的先验知识。Resnet 作为一种残差网络，能够在很深层次的网络结构情况下，依然能够有效的进行训练。完美的替代了 vgg 网络。

结语

本次设计主要目的是通过深度学习完成犬图片识别分类，利用到了卷积神经网络强大的特征提取能力，在带有 GPU 的计算机上能够快速的进行识别分类任务，做到替代人工的能力，深刻的反映了技术发展对人类生产力的提升。

识别任务当中采用了多种训练手段和性能提升技术。包括了数据预处理，数据增强，多种网络结构，正则化，梯度下降算法的优化选择等。每一种方法都不是万能的，必须按照实际训练的情况来适应调整对比。奥卡利姆剃刀定律指出，如果没有必要，不应该为模型加入过于复杂的东西，简单有效的原则对卷积神经网络来说是只管重要的。因此要有充分的理解各种方法的适用场景和优劣，才能作出合理的抉择。

通过对多方面深度学习知识的学习，最后在一次识别分类任务中组合使用起来，代码与结果看起来是简洁明了的，但是这背后蕴含的数学理论是深厚的。一行简单的命令背后，是无数学者专家的经验结晶。网络最后的识别结果能够比肩升至是超过人工的识别准确度（我相信很难有人完成一万张 120 个种类的犬图片识别任务）。

同时最后识别准确度不是 100%，这是因为这一类高级的识别任务和低级的简单逻辑任务是有不一样的特点。这一类识别任务带有主观性。一切都是以人的意志为转移。也就是说狗的类别是存在于人的定义，而定义本身就是不标准的。从这一个角度来看，神经网络和数学是如出一辙的准确。但是神经网络面对的任务本身确是不准确的。有一个理论说，任何再精妙的机器学习的算法如果没有置于实际的场景之下，那么和瞎猜是没有任何区别的。

人工智能已经为生活的方方面面带来了改变，是继工业革命之后的又一次人类生产水平的飞越提升，工业革命大大的解放了人类的手，而人工智能可以说得上是再逐步的解放人类的大脑。在这方面的研究和发展日新月异，新的技术层出不穷。

致谢

行文至此，我的大学生涯也进入了倒计时，始于丁酉金秋，终于辛丑盛夏，回顾过往，皆是序曲，从最开始的懵懂无知，到现在有一个未来的方向规划，机遇与挑战并存处，还记得一句话：“时光里有年少的不羁和浪荡，有青春的颓废和迷茫，也有成熟之后的坦然和温暖”。四年匆匆而过，曾以为来日方长，而现在才知如今最好，回想大学生活依旧历历在目，是我人生之中最宝贵的经历，留下青春的回忆与人生的收获，纵有万般不舍，终要踏上下一段征程。

润育桃李，鱼渔双授。感谢一路走来给与我帮助与支持的老师们，带我走进计算机这个世界，感受数字化的魅力，将自己所学所得传授于我；感谢周慧斌老师对我学习的指导和生活的成长，亦师亦友，在我迷茫时给予我建议与帮助，不光是老师，也是我的班主任，悉心指导，督促学习，真诚待人，在此衷心感谢导师和每位老师对我的帮助与启迪！

同门挚友，四载同窗。我们来自五湖四海，也终将回到人山人海，生活学习同在一起，为做一事而共同努力，苦乐与共，各位同学陪伴我走过青春美好的时刻，让短暂的四年里感受温暖与关怀，只愿大家前程似锦，待到梦回少年时，扬鞭骏马策！

征途漫漫，惟有奋斗。自己的大学生涯即将落幕，这不是终点，而是新生活、新奋斗的起点，前行必有曙光，自己要成为追光的人，追着光，靠近光，成为光，散发光。感谢自己大学中的每次经历，让我成长，让我更加坚强，拓宽我自己的宽度。

人生虽有离别日，山水应有相逢时。再见韶山南路 498 号。我们后会有期！无论身在何处，始铭记“求是求新，树木树人”，用自己的一份力量，添一道光，发一份热！

参考文献

[1]洪奇峰,施伟斌,吴迪,罗力源.深度卷积神经网络模型发展综述[J].软件导刊,2020，19(04):84-88.

[2]陈希孺,2009.概率论与数理统计[M].中国科学技术大学出版社.

[3]陈希孺,2009.数理统计学教程[M].中国科学技术大学出版社.

MacKay D J C, 2003. Information theory, inference, and learning algorithms[M]. Cambridge University Press.

[4]Srikanth H.Requirements based Test case prioritization[C].Student Research Forum in th 12th ACM SIGSOFT International Symposium on the Foundations of Software Engineering.2004.

[5]杨真真,匡楠,范露,康彬.基于卷积神经网络的图像分类算法综述[J].信号处理,2018,34(12):1474-1489.

[6]袁文翠,孔雪,等.基于迁移学习的图像识别研究[J].微型电脑应用,2018,34(7):123-124.

[7]Dominic Masters, Carlo Luschi,Revisiting Small Batch Training for Deep Neural Networks, arXiv:1804.07612v1.

[8]Multimodal Task-Driven Dictionary Learning for Image Classification.[J] . Bahrampour Soheil,Nasrabadi Nasser M,Ray Asok,Jenkins William Kenneth. IEEE transactions on image processing : a publication of the IEEE Signal Processing Society . 2016 (1).

[9]基于深度学习和迁移学习的多任务图像分类[D]. 贺智超.华南理工大学 2017.

[10]深度学习在图像识别中的应用研究综述[J]. 郑远攀,李广阳,李晔. 计算机工程与应用. 2019(12).

[11]Error bounds for approximations with deep ReLU networks[J] . Dmitry Yarotsky. Neural Networks . 2017.