第 3 章

基于深度学习的数字图像处理

深度学习已经成为数字图像处理的主流方法。工业机器视觉系统开发过程中，经常将传统数字图像处理用作图像的数据增强、预处理和后处理等环节，将深度学习作为图像分类、目标检测、实例分割等应用的主要方法。

本章首先介绍人工智能、机器学习、强化学习、深度学习的基本概念，明确深度学习在人工智能领域中的地位和作用；然后重点对深度学习学习对象、学习过程和评价标准等进行详细描述，期望对深度学习建立一个宏观整体的认识；最后对普通神经网络、一般深度卷积神经网络及构成、典型深度卷积神经网络及应用等进行深入的讨论。

3.1 人工智能与机器学习

3.1.1 人工智能概述

人工智能（Artificial Intelligence，AI）是对模拟、延伸和扩展人类智能的理论、方法、技术进行研究的一门科学。通俗地说，人工智能就是使用人工的方法在计算机上实现的智能。从计算机科学的角度看，人工智能是计算机科学的一个分支。从学科交叉的角度看，人工智能涉及计算机科学、脑科学、认知科学、心理学、逻辑学和统计学等多个学科，如图 3-1 所示。

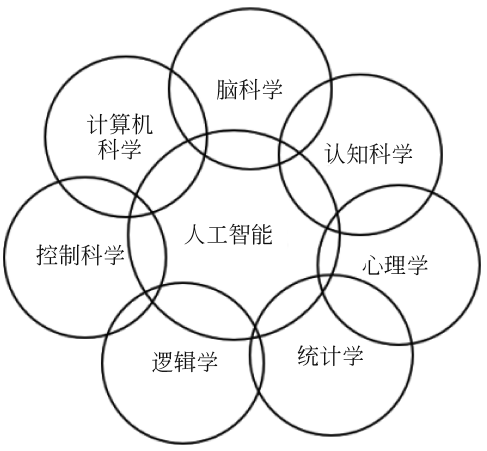


图 3-1 人工智能的相关学科

从 1956 年正式提出人工智能的概念起，人工智能的研究、发展已有 60 多年的历史。不同学科背景的学者对人工智能做出了不同的研究尝试，逐渐形成三个主流学派，分别是符号主义、连接主义和行为主义。

（1）符号主义，又称为逻辑主义、心理学派或计算机学派，该学派认为人工智能源于数学逻辑，使用符号运算模拟人的认知过程。典型的代表成果是基于产生式规则构建的专家系统。

（2）连接主义，又称为仿生学派或生理学派，该学派认为人工智能可以基于神经网络、网络间的连接机制以及学习算法实现。典型的代表成果是普通神经网络以及在此基

础上发展的深度卷积神经网络，这些网络已经广泛应用于人脸识别、字符识别、产品表面的缺陷识别等。

(3) 行为主义，又称为进化主义或控制论学派。该学派认为人工智能表现为智能体在适应环境的控制过程中的自寻优、自适应、自校正、自组织和自学习等。典型代表成果是基于强化学习的智能机器人系统。

3.1.2 机器学习概述

机器学习(Machine Learning)研究的是使用计算机模拟或实现人类学习过程的科学，是实现人工智能的重要途径。人工智能与机器学习的关系如图 3-2 所示。当前，机器学习是人工智能最重要、最具有热度的研究分支，机器学习包括传统的机器学习算法，如线性回归、决策树、支持向量机、聚类、主成分分析和人工神经网络，以及当下最受关注的深度学习、强化学习以及深度强化学习等。

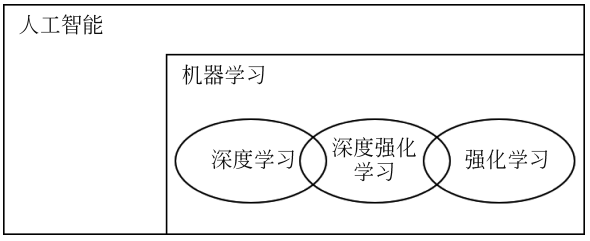


图 3-2 人工智能与机器学习的关系

按照学习方式进行分类，机器学习可以分为如下三种。

(1) **有监督学习**(Supervised Learning)，又称为有导师学习，指的是从带有标签的数据中训练模型，以对未知数据作出分类或预测。例如，从网络上收集 1000 张猫和狗的图像，并且对每张图像中猫和狗的位置及类别进行标注，然后将这些带标签的数据送入机器学习系统进行训练，模型训练达到一定的评价标准后，停止训练。当给定一个新的带有猫和狗的图像时，模型就可能指出猫和狗在图像中的位置，并以概率方式标注模型认可的分类置信度，这是一种典型的有监督学习。有监督学习的常见算法包括决策树、支持向量机、人工神经网络和深度卷积神经网络等。

(2) **无监督学习**(Unsupervised Learning)，又称为无导师学习，指的是从不带标签的数据中发现隐含的结构。例如，工厂在生产过程中，分析、收集并找到可能影响产品表面质量的 50 个因素及数据，通过学习算法找到影响产品表面质量的最重要的 10 个关键因素，这是一种典型的无监督学习过程。无监督学习的常见算法包括聚类和主成分分析等。

(3) **强化学习**(Reinforcement Learning)，指的是智能体在与环境的交互过程中通过执行一定的学习策略以达成回报最大化或实现特定目标的学习。强化学习过程示意，如图 3-3 所示。智能体基于当前环境的状态 S(State)，执行一定的动作 A(Action)，环境受到智能体动作 A 的影响会将状态从 S 更新为 S′，并获得奖励 R(Reward)，如此循环往复，直到达到某一设定值。例如，寒冷的冬天里，一只没有见过火焰的动物想取暖。这只动物可以认为是智能体。起初，该动物距离火焰较远，动物与火焰之间的距离可以定义为状

态，根据距离的远近可以定义系统不同的状态。该动物想取暖，就向火焰靠近。而当动物距离火焰非常近时，又会感到火焰的灼热，就要远离火焰。这里，靠近和远离就是智能体采取的动作。智能体获得的奖励就是取暖但不能灼伤自己。目前，强化学习已经在无人驾驶、AI 游戏、智慧物流等领域得到广泛的应用。

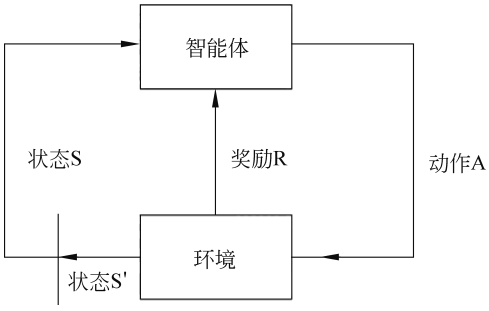


图 3-3　强化学习过程示意

3.2　深度学习的基本概念

3.2.1　深度学习概述

深度学习(Deep Learning)是机器学习的一个子研究领域。深度学习的概念源于人工神经网络。其“深度”的含义，一般指的是构成神经网络的层数较多，通常有 5 层、10 层甚至 50 层等。深度学习的本质是通过多层非线性变换，从大量数据中自动学习各种低层和高层的复杂特征，从而替代传统机器学习中需要人工设计的特征。

通过前面内容的学习，我们知道：如果要提取图像中物体的边缘，可以使用 Roberts、Sobel、Prewitt 算子或 Canny 算子；如果要计算图像的纹理特征，可以使用基于灰度共生矩阵的纹理特征、LBP 纹理特征、分形维数。这些传统的数字图像处理方法，对于图像特征的提取都有明确的数学公式的表达形式。而在深度学习中，对于图像中物体形状、边缘、纹理、亮度、对比度、颜色等特征的提取，都是通过算子或算子的组合进行级联和构造，并由神经网络自动学习而来。很多时候，深度学习抽取到的特征无法使用自然语言进行描述。

当前，深度学习应用于数字图像处理中，大多是有监督学习。例如，当使用深度卷积神经网络进行图像分类时，需要输入大量的带类别标签的图像。由于数字图像处理和机器视觉的研究热度居高不下，深度学习已经成为有监督学习中重要的研究方法之一。

近年来出现的深度 Q 网络模型(Deep Q-Network，DQN)则是将深度学习与强化学习方法相结合，采用卷积神经网络(简称 Q 网络)拟合动作价值函数，成功实现从输入图像到输出动作的端到端学习。深度学习、强化学习、深度强化学习已经成为机器学习中最引人关注的方法。

大多数应用中，深度学习作为一种有监督学习算法进行使用。深度学习作为一种重要的机器学习算法，也要关注机器学习的本质问题：如何学习？学习的目标是什么？如

何评价学习效果？学习的知识如何表达和存储？机器学习的一般过程如图 3-4 所示。

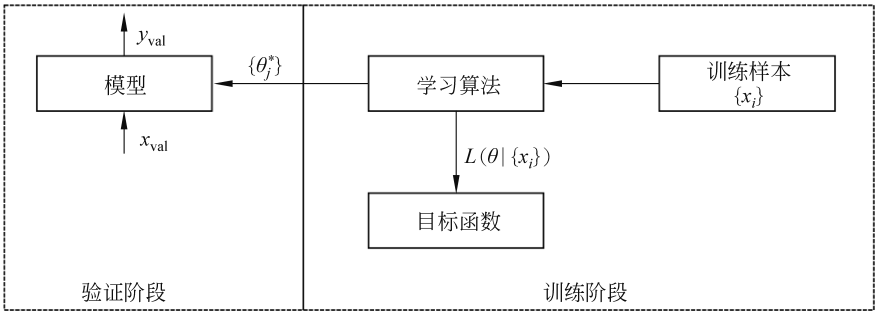


图 3-4　机器学习的一般过程

一般来说，机器学习可以分为训练阶段和验证阶段。在训练阶段，根据输入样本 $\{x_i\}$，在学习算法的驱动下，针对一定损失函数 $L(\theta|\{x_i\})$，期望通过样本的训练降低损失函数的值。这里，$\{\theta\}$是一组模型的训练参数。当训练达到一定的迭代次数或其他终止条件时，将学习到的知识以一定方式进行保存即可获得模型，模型可以理解为一组经过训练优化调整后的参数$\{\theta_j^*\}$；在验证阶段，对训练获得的模型输入新的数据 x_{val}，检验并统计输出结果 y_{val}是否满足系统精度要求。当模型达到一定的精度要求时，模型即可投入现场使用，进入现场测试和运行阶段。

下面以一个机器视觉中目标检测应用为示例，使用深度学习中的深度卷积网络详细说明模型的训练过程。炼钢生产过程中，钢包包号识别是一个典型的目标检测问题，如图 3-5 所示，不仅需要识别出钢包包号图像中 1～2 位的钢包包号(使用 1～2 位的 0～9 的数字进行标识)，还需要通过四边形的方式给出钢包包号的准确位置。

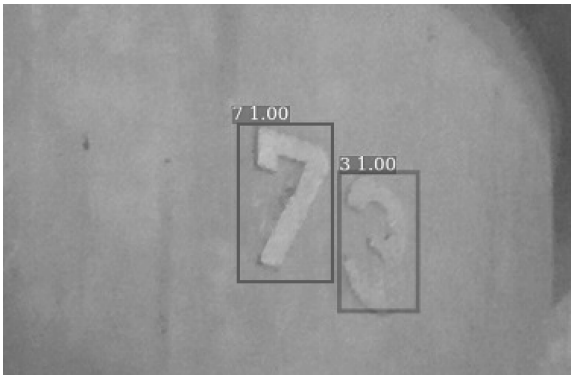


图 3-5　钢包包号的目标检测

模型训练阶段，使用大量的已经标注标签的图像及标签数据输入深度卷积神经网络。对于目标检测，一般使用 LabelImg 工具进行标注，如图 3-6 所示。显然，由于人工标注了真实的目标位置，因此这是一种有监督学习。

深度卷积神经网络接收到图像及图像中目标的标签(一般保存为 XML 文件)后，将图像作为输入，通过深度卷积神经网络中的各个算子进行层层计算，输出目标的类别和目

图 3-6　使用 LabelImg 工具标注钢包包号

标的位置，即包号中每个字符的具体值以及所在的四边形框位置。目标函数用于描述模型输出的每个字符的类别是否正确，即**分类损失**(Classification Loss)，以及每个字符的四边形位置是否精确，即**回归损失**(Regression Loss)。模型学习的目标是降低分类损失和回归损失的总和。一般情况下，对于深度学习来说，目标函数值越大，损失值越大，模型精度越低；目标函数值越小，损失值越小，模型精度越高。有关深度学习的目标函数，将在 3.2.2 节中详细介绍。

面向目标检测的深度卷积神经网络的学习过程，如图 3-7 所示。对于深度卷积神经网络来说，模型的训练过程是一个多轮反复输入图像、特征提取、计算损失和调整神经元之间连接权重的过程。对于钢包包号目标检测的深度卷积神经网络来说，训练阶段需要多次输入钢包包号图像，通过神经网络的各层算子提取有效特征并进行组合，以进行目标类别判定和精确位置的计算。当模型输出预测的目标类别和位置后，即可计算分类损失和回归损失，然后根据该损失调整神经网络中神经元之间的连接权重。这些权重就是深度卷积神经网络对于知识的表达和存储形式。有关神经网络学习算法，将在 3.3.4 节中详细介绍。

3.2.2　深度学习的损失函数

机器学习经常解决的问题有两类：**分类**(Classification)和**回归**(Regression)。分类问题的输出为有限个离散值。回归问题的输出为无限个连续值。例如，预测明天是晴天、雨天还是多云，预测某个字符是 0～9 中具体哪一个，这些都是典型的分类问题；预测明天天气的气温是多少度，预测某个字符出现在图像中的位置，这些都是典型的回归问题。

深度学习是一种典型的机器学习方法，同样也要处理分类和回归问题。对于分类的损失函数，深度学习中经常使用**均方误差损失**(Mean Squared Error，MSE)和**交叉熵损失**(Cross Entropy，CE)进行描述。

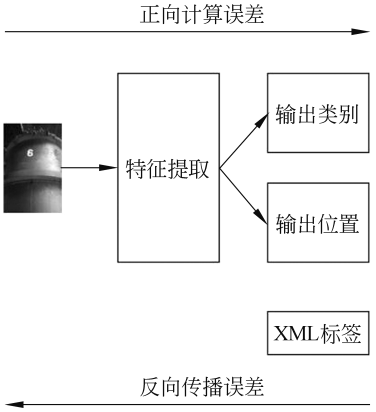


图 3-7　面向目标检测的深度卷积神经网络的学习过程

1. 均方误差损失

当模型的输出有 M 个类别，设这 M 个输出为$[\hat{y}_1,\hat{y}_2,\cdots,\hat{y}_M]$，分别是每个类别的预测输出概率。例如，当识别的字符是 0～9 的 10 个字符的其中一个字符时，如果$[\hat{y}_1,\hat{y}_2,\cdots,\hat{y}_M]=[0.1,0.8,0.05,0.02,\cdots,0.01]$，则在输出概率中，字符“1”对应的概率 0.8 为最大值，输出的分类结果为字符“1”，即识别出来的字符为“1”。

设真实的标签为$[y_1,y_2,\cdots,y_M]$。对于识别字符 0～9 的其中一个字符问题，如果真实的字符为“1”，则$[y_1,y_2,\cdots,y_M]=[0,1,0,\cdots,0]$。

均方误差损失(MSE)定义，如式(3-1)所示。

$$L=(y_1-\hat{y}_1)^2+(y_2-\hat{y}_2)^2+\cdots+(y_M-\hat{y}_M)^2 \tag{3-1}$$

如果真实标签为类别 p，$1\leqslant p\leqslant M$，则均方误差损失为

$$\begin{aligned}L&=(y_1-\hat{y}_1)^2+(y_2-\hat{y}_2)^2+\cdots+(y_M-\hat{y}_M)^2\\&=(1-\hat{y}_p)^2+(\hat{y}_1^2+\cdots+\hat{y}_{p-1}^2+\hat{y}_{p+1}^2+\cdots+\hat{y}_M^2)\end{aligned}$$

可以看出，预测类别 p 的均方误差损失，不仅与类别 p 的预测概率有关，还与其他类别的预测概率有关。

2. 交叉熵损失

交叉熵损失(*CE*)定义，如式(3-2)所示。

$$L=-(y_1\log\hat{y}_1+y_2\log\hat{y}_2+\cdots+y_M\log\hat{y}_M) \tag{3-2}$$

如果真实标签为类别 p，$1\leqslant p\leqslant M$，则交叉熵损失为

$$\begin{aligned}L&=-(y_1\log\hat{y}_1+y_2\log\hat{y}_2+\cdots+y_M\log\hat{y}_M)\\&=-y_p\log\hat{y}_p\\&=-\log\hat{y}_p\end{aligned}$$

可以看出，预测类别 p 的交叉熵损失，只与类别 p 的预测概率有关，与其他类别的预测概率无关。大多数情况下，深度学习选择交叉熵作为分类损失函数。

深度学习的训练过程就是多次反复输入图像、特征提取、计算损失和调整参数的过程，目标是降低损失函数的值，以获得高精度或误差较小的模型。

3. 模型过拟合与欠拟合

由于训练样本和验证样本不是相同的，深度学习训练出的模型在训练数据上和验证数据上的表现可能差异较大。

若模型在训练数据上获得了较小的误差损失，而在验证数据上的误差损失较大，则称为模型过拟合。过拟合的原因可能是特征维度太多、模型过于复杂、参数过多、训练数据太少等。若模型在训练数据上误差一直较大，无法找到合适的模型表达来描述数据集，则称为模型欠拟合。欠拟合的可能原因是训练不充分、训练数据太少、训练数据中噪声过多、超参数设置不合理等。模型过拟合与欠拟合，如图 3-8 所示。

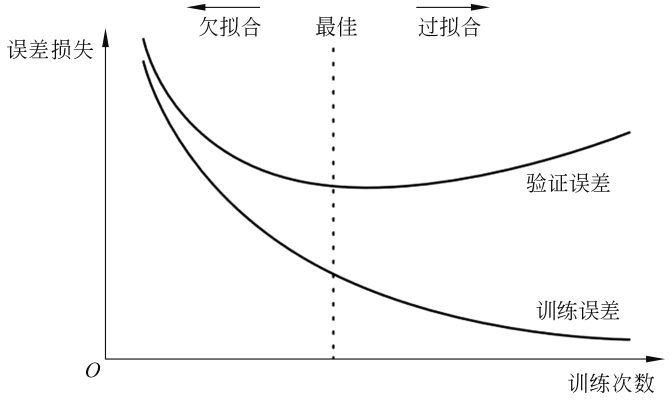


图 3-8　模型过拟合与欠拟合

有关图像处理中目标检测的回归损失函数，将在目标检测的深度卷积神经网络相关章节详细介绍。

3.2.3　深度学习的评价指标

当使用深度学习方法训练出模型后，需要系统地讨论模型在验证数据上的表现。对于分类问题，常用的评价指标大多基于混淆矩阵，并据此定义精确率(Precision)、召回率(Recall)、准确率(Accuracy)、错误率(Error Rate)以及 F1 值(F-measure)。

混淆矩阵，又称为误差矩阵。混淆矩阵的每一列代表预测值，每一行代表实际的类别，见表 3-1。

表 3-1　混淆矩阵

预测值 / 实际值	预测为正样本	预测为负样本
实际为正样本	TP	FN
实际为负样本	FP	TN

这里，TP、FP、FN 和 TN 的含义如下。

(1) TP(True Positive)：实际为正样本，预测为正样本；

(2) FP(False Positive)：实际为负样本，预测为正样本；

(3) FN(False Negative)：实际为正样本，预测为负样本；

(4) TN(True Negative)：实际为负样本，预测为负样本。

例如，一个火灾智能检测系统，模型的实际检测结果，如图 3-9 所示。方框里面是模型给出检测结果为火焰的情况，其中，8 个真实火焰被检测出来火焰，2 个灯泡的照明被错误地检测出来为火焰；方框外面是模型没有给出检测结果的情况，其中，3 个真实的火焰，1 个灯泡的照明，火灾检测模型没有给出检测结果。

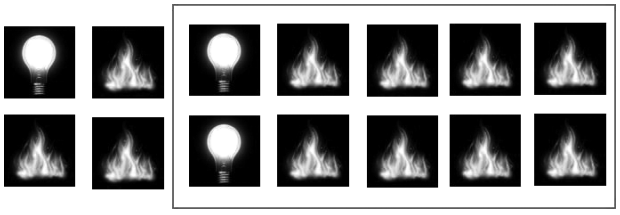


图 3-9　火灾检测结果的示意

这里，对于火灾智能检测系统，火焰是正样本，灯泡照明是负样本。方框里面的 8 个火焰被检测出来火焰，即 8 个正样本被识别为 8 个正样本，TP＝8；方框里面的 2 个灯泡照明被检测出来火焰，即 2 个负样本被识别为 2 个正样本，FP＝2；方框外面的 3 个火焰没有被识别出来火焰，即 3 个正样本被识别为 3 个负样本，FN＝3；方框外面的 1 个灯泡照明没有被识别出来，即 1 个负样本被识别为 1 个负样本，TN＝1。

下面对精确率、召回率、准确率、错误率以及 F1 值进行定义并解释说明。

(1) 精确率指的是预测为正类的全部样本中，预测正确的正类样本比例。精确率的计算，如式(3-3)所示。

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}} \tag{3-3}$$

(2) 召回率指的是在全部的正类样本中，预测正确的正类样本比例。召回率的计算，如式(3-4)所示。

$$\text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}} \tag{3-4}$$

(3) 准确率指的是在全部样本中，预测正确的正类样本和预测正确的负类样本的比例。准确率的计算，如式(3-5)所示。

$$\text{Accuracy}=\frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}} \tag{3-5}$$

(4) 错误率(Error Rate)指的是在全部样本中，预测错误的正类样本和预测错误的负类样本的比例。错误率的计算，如式(3-6)所示。

$$\text{ErrorRate}=\frac{\text{FP}+\text{FN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}} \tag{3-6}$$

(5) F1 值定义，如式(3-7)所示。

$$\text{F1}=\frac{2*\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}} \tag{3-7}$$

根据上述定义，对于该火灾智能检测系统，计算各个指标的结果如下。

$$\text{Precision}=\frac{\text{TP}}{\text{TP}+\text{FP}}=\frac{8}{8+2}=\frac{8}{10}$$

$$\text{Recall}=\frac{\text{TP}}{\text{TP}+\text{FN}}=\frac{8}{8+3}=\frac{8}{11}$$

$$\text{Accuracy}=\frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}}=\frac{8+1}{8+2+3+1}=\frac{9}{14}$$

$$\text{ErrorRate}=\frac{\text{FP}+\text{FN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}}=\frac{2+3}{8+2+3+1}=\frac{5}{14}$$

$$\text{F1}=\frac{2\times\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}=\frac{2\times\frac{8}{10}\times\frac{8}{11}}{\frac{8}{10}+\frac{8}{11}}=0.76$$

3.2.4　深度学习模型开发的一般过程

本节主要以机器视觉领域使用的深度卷积神经网络的模型开发过程为例，对深度学习模型的开发过程做一个简要介绍。这里以 X 射线底片上的焊缝缺陷检测为例，需要识别的缺陷主要包括裂纹、未熔合、未焊透、条形缺陷和圆形缺陷共 5 种。模型开发过程包括：数据标注、数据扩增、数据格式转换、模型训练、模型转换、模型调用等步骤。

1. 数据标注

数据标注指的是使用专用标注软件对输入图像数据进行分类标识和目标标注。对于 X 射线底片上的焊缝缺陷，使用 LabelImg 软件标注裂纹、未熔合、未焊透和条形等各种缺陷的具体位置，如图 3-10 所示。这里使用四边形框标注一个裂纹缺陷，标记缺陷名称为 crack，标注软件会自动记录缺陷类型和缺陷所在位置信息。位置信息一般记录为四边形左上角坐标、宽度和高度。

2. 数据扩增

数据扩增指的是不增加原始数据，仅对原始数据进行一些变换而获得更多的训练数据。数据扩增的主要目的是提高模型的鲁棒性和泛化能力。对于图像数据来说，常见的扩增方法包括：基于几何方法的扩增、基于颜色或亮度的扩增、基于噪声的扩增、基于深度学习方法的扩增。

- 基于几何方法的扩增包括图像翻转、图像旋转、图像扭曲、图像缩放和图像裁减等。
- 基于颜色或亮度的扩增包括亮度调整、色度调整、饱和度调整和对比度调整等。
- 基于噪声的扩增包括向图像中添加高斯、椒盐或泊松噪声等。

上述扩增方法，均可以使用传统方法的数字图像处理实现。

- 基于深度学习方法的扩增的典型代表是使用对抗生成网络（Generative Adversarial Networks，GAN）生成新的数据，实际是通过深度学习训练出一个新的模型来扩增数据。

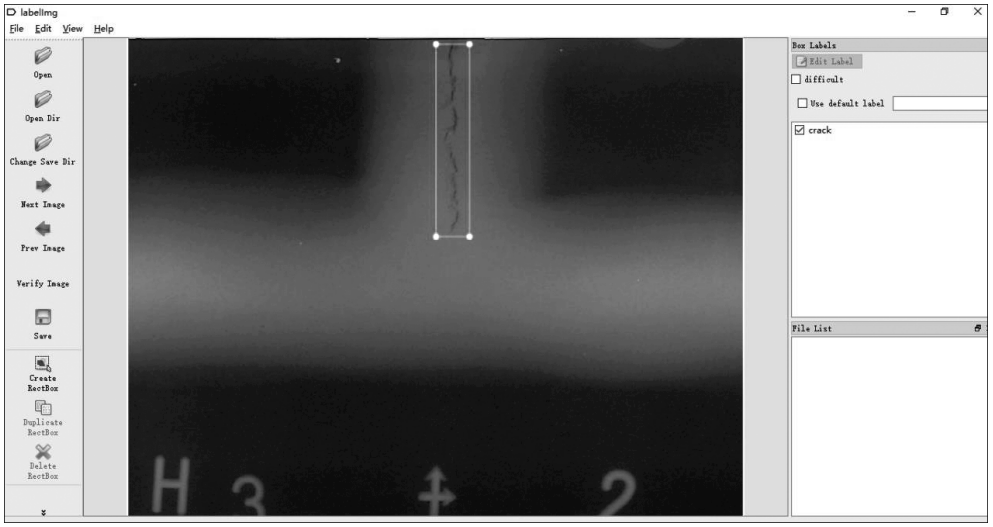
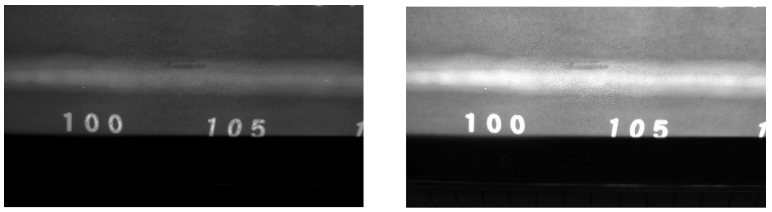


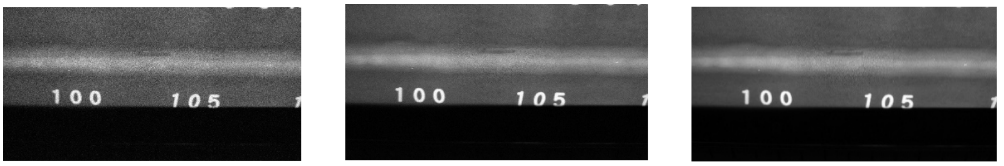
图 3-10　使用 LabelImg 标注 X 射线底片上的焊缝缺陷

增加亮度、减少亮度的数据扩增示意，如图 3-11 所示。图 3-11(a)是对原图像的亮度降低 30 的扩增图像，图 3-11(b)是对原图像的亮度提高 30 的扩增图像，两种扩增方式属于图像的线性点运算。图像中添加高斯噪声的示意，如图 3-12 所示。图 3-12(a)、图 3-12(b)和图 3-12(c)分别加入了方差为 0.01、0.001 和 0.0001 的高斯噪声，属于图像的算术运算。



(a) 亮度降低30　　(b) 亮度提高30

图 3-11　亮度扩增的示意



(a) 加入方差为0.01的高斯噪声　(b) 加入方差为0.001的高斯噪声　(c) 加入方差为0.0001的高斯噪声

图 3-12　噪声扩增的示意

3. 数据格式转换

数据格式转换指的是将图像数据及标签文件组成的训练数据转换成为模型训练要求的格式。将图像数据及标签文件转换为 TFRecord 格式的示意，如图 3-13 所示。其中，

JPEGImages 文件夹存放图像数据，Annotations 文件夹存放标签文件。TFRecord 格式是 TensorFlow 训练的标准数据格式，将数据存储为二进制文件，复制和读取效率更高，可以更加高效地进行数据的频繁访问操作。

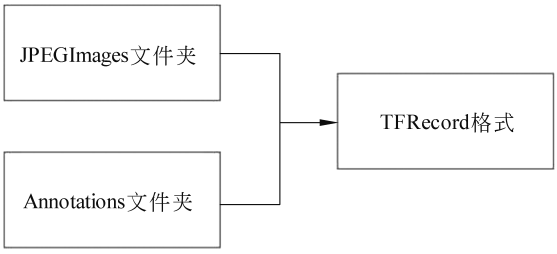


图 3-13　训练数据转换为 TFRecord 格式

4. 模型训练

模型训练指的是多次反复输入图像、进行特征提取、计算损失和调整参数的过程。模型训练过程输出的各种损失函数值，如图 3-14 所示。其中，cla_loss 为分类损失，loc_loss 为目标检测的回归损失。

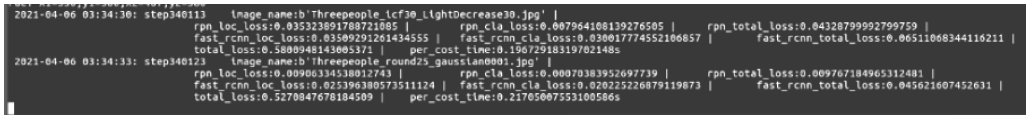


图 3-14　模型训练过程

5. 模型转换

模型训练过程得到的模型依赖于训练环境，只能在指定的框架下运行。例如，基于 TensorFlow 进行模型训练，得到的含有检查点 checkpoint 文件的 ckpt 模型就属于这种情况。因此，经常需要将训练得到的模型转换为可独立运行的文件。对于 TensorFlow，经常需要将 ckpt 模型转换为 pb 模型。pb 模型文件具有编程语言独立性，可独立运行并允许任何编程语言进行解析并运行。ckpt 模型转换为 pb 模型的示意，如图 3-15 所示。

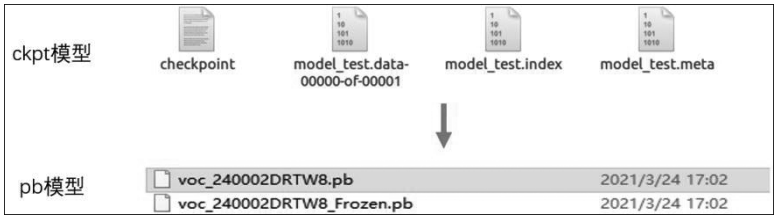


图 3-15　模型转换

6. 模型调用

一般来说，基于深度学习开发机器视觉系统，需要完成模型训练和模型调用两个阶段的工作。模型训练和模型调用可以使用不同的编程语言开发。前述的模型训练就是使用 Python 编程语言在 Ubuntu 操作系统下进行的，通过模型转换为 pb 模型，如图 3-16 所示。使用 C++ 并基于 QT 框架开发的前端界面，基于 TensorFlow C++ 编程接口可以调用 pb 模型，如图 3-17 所示。

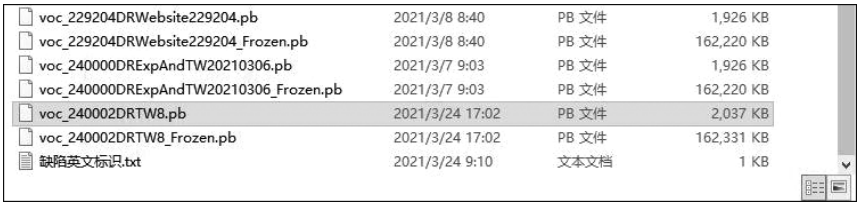


图 3-16　前端界面调用的 pb 模型

图 3-17　前端界面

3.3　普通神经网络

深度卷积神经网络是数字图像处理或机器视觉开发中广泛使用的一种深度学习方法，而深度卷积神经网络是从一般或普通神经网络发展而来的。

神经网络(Neural Network，NN)，又称为**人工神经网络**(Artificial Neural Network，ANN)，是一种模仿人类神经网络行为特征，进行分布式并行信息处理的模型。神经网络是由大量的人工神经元按照一定拓扑结构相互连接而成，通过调整人工神经元之间的连接权重或连接关系达到信息处理、知识存储进而模拟人工智能的目的。

3.3.1　人工神经元

人工神经元是对生物神经元的抽象和简化。生物神经元由细胞体、树突和轴突组成。细胞体是神经元的主体，树突是感知器官，接受来自其他神经元传递的输入信号，轴突用来传递本神经元产生的输出信号。

人工神经元是对组成生物神经元的树突、轴突和细胞体进行数学抽象而来。具体来说，人工神经元就是将树突和轴突简化并抽象为多个输入和单个输出，将细胞体抽象为一个激活函数，如图 3-18 所示。

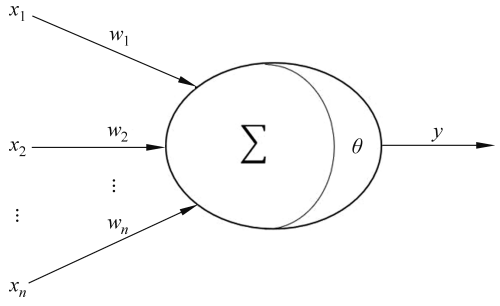


图 3-18　人工神经元模型

人工神经元的输入与输出关系，如式(3-8)所示。

$$y = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right) \tag{3-8}$$

这里，$x_1, x_2, \cdots, x_n$ 是人工神经元的 n 个输入，$w_1, w_2, \cdots, w_n$ 是 n 个输入对应的权重，θ 是神经元的阈值，输入和阈值进行线性组合 $\sum_{i=1}^{n} w_i x_i - \theta$ 作为激活函数 f 的输入，y 是神经元的输出。神经元的输出依赖于具体的激活函数。

3.3.2　神经网络的拓扑结构

神经网络由多个神经元按照一定拓扑结构连接而成。大多数神经网络的拓扑结构都是前馈神经网络，如图 3-19 所示。

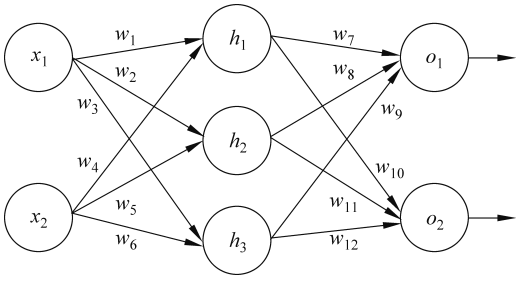


图 3-19　前馈神经网络

前馈的神经网络由多层神经元组成，包括输入层、隐藏层和输出层，各层顺序连接，数

据严格遵从从输入层经过隐藏层到输出层进行单向流动，同一层内的神经元之间没有连接。图 3-19 中，x_1 和 x_2 是前馈神经网络的输入，o_1 和 o_2 是网络的输出，h_1、h_2 和 h_3 组成神经网络的一个隐藏层。前馈神经网络的信息处理能力主要由隐藏层层数、拓扑结构、激活函数等决定。神经元之间的连接权重就是模型的参数，模型训练过程就是不断地调整神经元之间的连接权重来降低损失函数的值，训练好的模型实际就是一个调整优化后的神经元连接参数及网络拓扑结构。这种前馈神经网络的典型代表是反向传播(Back Propagation，BP)神经网络。

3.3.3 激活函数

激活函数指的是将神经元的输入映射为输出的函数。引入激活函数可以增强神经网络的拟合能力。常见的激活函数包括 Sigmoid、tanh、ReLU 和 PReLU。

1. Sigmoid 函数

Sigmoid 函数的表达式，如式(3-9)所示。

$$f(x)=\frac{1}{1+\mathrm{e}^{-x}} \tag{3-9}$$

Sigmoid 函数图像，如图 3-20 所示。

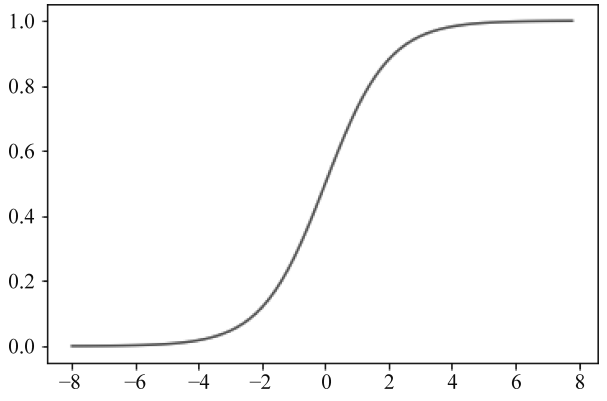


图 3-20 Sigmoid 函数图像

Sigmoid 函数，又称为逻辑回归(Logistic Regression，LR)函数，作为一种指数函数模型广泛应用于面向分类和回归问题的机器学习。Sigmoid 函数的优点是：输出映射在(0，1)之间，且单调连续，输出范围有限且稳定，输出范围(0，1)之间的值还可以作为分类问题中类别的输出概率，可直接用作输出层；其缺点是：由于其双端饱和的特性(函数两端的梯度趋近于零)，训练过程中容易导致梯度消失，模型训练失败或训练不充分，模型欠拟合，此外，Sigmoid 函数输出并不是以 0 为中心。

2. tanh 函数

tan$h$ 函数的表达式，如式(3-10)所示。

$$f(x)=\frac{\mathrm{e}^{x}-e^{-x}}{\mathrm{e}^{x}+\mathrm{e}^{-x}} \tag{3-10}$$

tan*h* 函数图像，如图 3-21 所示。

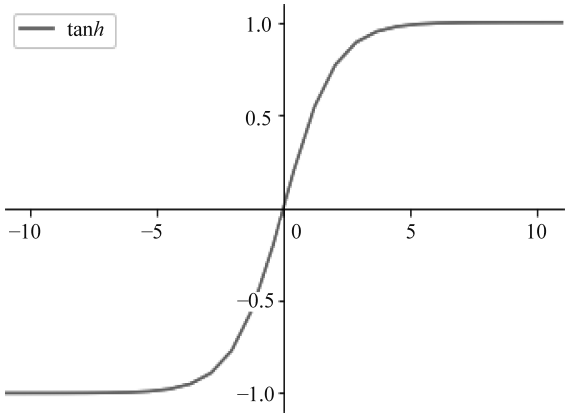


图 3-21　tan*h* 函数图像

tan*h* 函数，又称为双曲正切函数。tan*h* 函数的优点是：解决了 Sigmoid 的输出不是以 0 为中心的问题；其缺点是：函数依然具有双端饱和的特性，可能在模型训练中导致梯度消失。

3. ReLU 函数

ReLU 函数，又称为线性整流函数（Rectified Linear Unit，ReLU）。ReLU 函数的表达式，如式(3-11)所示。

$$f(x)=\begin{cases}x, & x\geqslant 0\\ 0, & x<0\end{cases} \tag{3-11}$$

ReLU 函数图像，如图 3-22 所示。

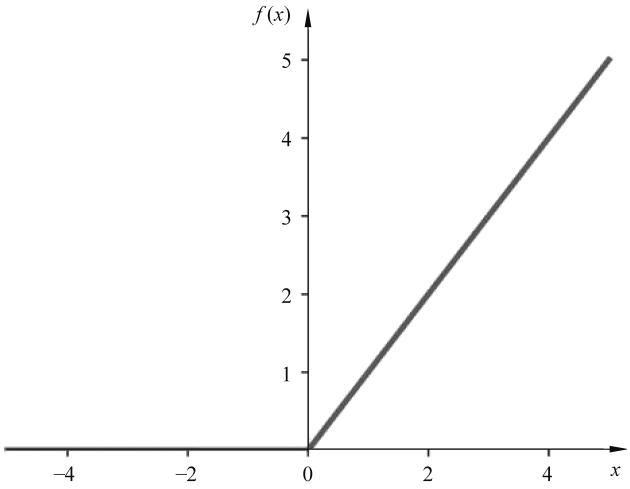


图 3-22　ReLU 函数图像

ReLU 函数的优点是：解决了梯度消失的问题，在 x 为正数的区间中，神经元不会饱

和，模型训练过程中能够快速收敛；其缺点是：输出不是以 0 为中心，而且当输入为负时，神经元不被激活。因此，在训练过程中可能出现神经元“死亡”，神经元之间的连接权重无法更新的情况。

4. PReLU 函数

PReLU 函数的表达式，如式(3-12)所示。

$$f(x)=\max(0,x)+a*\min(0,x) \tag{3-12}$$

PReLU 函数图像，如图 3-23 所示。

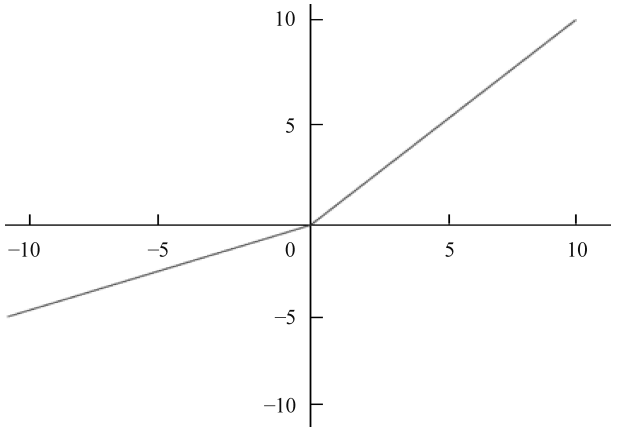


图 3-23　PReLU 函数图像

PReLU 函数中的 a 为可调整的参数，一般在(0,1)之间。当 a=0 时，PReLU 函数退化为 ReLU 函数；当 a 为非 0 实数时，PReLU 给所有的负值输入赋予一个非 0 的较小的斜率，从而改进 ReLU 函数在输入为负的情况下完全不被激活的情况。

3.3.4　BP 算法

本节主要介绍典型的前馈神经网络的学习算法，通过学习调整神经元之间的连接权重，形成知识表达、模拟人工智能。

前馈神经网络的学习过程总体上分为两个阶段：正向计算和反向传播。正向计算指的是输入数据从输入层依次经过隐藏层的各层神经元进行逐层计算，并最终通过输出层进行输出。在输出层，综合神经网络的输出和给定的标签，计算损失函数值；反向传播指的是将损失函数所表达的误差按原来正向计算的路径进行反向传递，并对沿途各个神经元之间的连接权重进行调整，期望下次进行正向计算时由损失函数计算的误差减小。

1. 正向计算

下面以一个含由两个输入、三个神经元组成的隐藏层和两个输出的简单前馈神经网络为例，详细说明 BP 算法的工作过程，如图 3-19 所示。

对于隐藏层的神经元 h_1，输入来自 x_1 和 x_2，输出给 o_1 和 o_2。输入和输出分别如式(3-13)和式(3-14)所示。

$$\mathrm{In}_{h_1}=w_1*x_1+w_4*x_2 \tag{3-13}$$

$$h_1 = \mathrm{Out}_{h_1} = \mathrm{Sigmoid}(In_{h_1}) \tag{3-14}$$

神经元 h_1 的输入与输出如图 3-24 所示。

隐藏层的神经元 h_2，输入同样来自 x_1 和 x_2，输出给 o_1 和 o_2。输入和输出分别如式(3-15)和式(3-16)所示。

$$\mathrm{In}_{h_2} = w_2 * x_1 + w_5 * x_2 \tag{3-15}$$

$$h_2 = \mathrm{Out}_{h_2} = \mathrm{Sigmoid}(\mathrm{In}_{h_2}) \tag{3-16}$$

隐藏层的神经元 h_3，输入同样来自 x_1 和 x_2，输出给 o_1 和 o_2。输入和输出分别如式(3-17)和式(3-18)所示。

$$\mathrm{In}_{h_3} = w_3 * x_1 + w_6 * x_2 \tag{3-17}$$

$$h_3 = \mathrm{Out}_{h_3} = \mathrm{Sigmoid}(\mathrm{In}_{h3}) \tag{3-18}$$

输出层神经元 o_1，输入来自 h_1、h_2 和 h_3，输出即模型的一个输出。输入和输出分别如式(3-19)和式(3-20)所示。

$$\mathrm{In}_{O_1} = w_7 * h_1 + w_8 * h_2 + w_9 * h_3 \tag{3-19}$$

$$o_1 = \mathrm{Out}_{o_1} = \mathrm{Sigmoid}(\mathrm{In}_{o_1}) \tag{3-20}$$

神经元 o_1 的输入与输出，如图 3-25 所示。

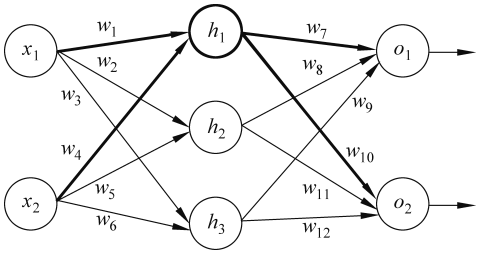


图 3-24　神经元 $\mathbf{h_1}$ 的输入与输出

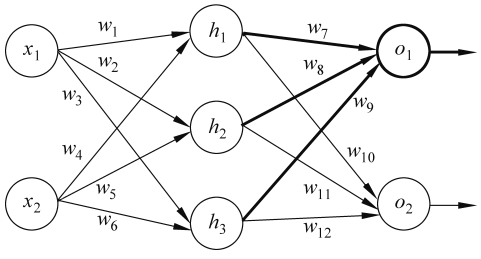


图 3-25　神经元 o_1 的输入与输出

输出层神经元 o_2，输入同样来自 h_1、h_2 和 h_3，输出即模型的另一个输出。输入和输出分别如式(3-21)和式(3-22)所示。

$$\mathrm{In}_{o_2} = w_{10} * h_1 + w_{11} * h_2 + w_{12} * h_3 \tag{3-21}$$

$$o_2 = \mathrm{Out}_{o_2} = \mathrm{Sigmoid}(\mathrm{In}_{o_2}) \tag{3-22}$$

为了计算简单，损失函数采用均方误差损失(MSE)，$\mathrm{Error} = \frac{1}{2}\sum_{i=1}^{2}(o_i - y_i)^2$，其中，$y_i$ 为真实输出(标签数据)，o_i 为神经网络的计算输出。

设输入 x_1 和 x_2 分别为 0.5 和 0.3，相应的真实输出为 0.23 和 −0.07。当各个神经元之间的连接权重确定之后，即可进行正向计算，如图 3-26 所示。一般来说，神经元之间的连接权重可初始化为(0,1)之间的小数，大多使用随机函数生成。

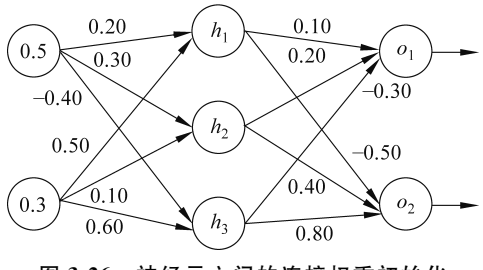


图 3-26　神经元之间的连接权重初始化

若隐藏层神经元和输出层神经元的输出分别如下。

$$h_1=\text{Sigmoid}(0.20*0.50+0.50*0.30)=0.56$$

$$h_2=\text{Sigmoid}(0.30*0.50+0.10*0.30)=0.54$$

$$h_3=\text{Sigmoid}(-0.40*0.50+0.60*0.30)=0.50$$

$$o_1=\text{Sigmoid}(0.10*0.56+0.54*0.20+(-0.30*0.50))=0.50$$

$$o_2=\text{Sigmoid}((-0.50*0.56)+0.54*0.40+0.80*0.50)=0.58$$

则均方误差损失(MSE)的计算如下。

$$\text{Error}=\frac{1}{2}(0.50-0.23)^2+\frac{1}{2}[0.58-(-0.07)]^2=0.25$$

2. 反向传播

反向传播指的是将损失函数所表达的误差按原来正向计算的路径进行反向传递。反向传播是根据微积分中的链式法则,沿着从输出层经过隐藏层到输入层的反向顺序,依次计算损失函数对权重参数的梯度,并据此进行调整。

对于图 3-25 所示的神经网络,以权重 w_7 为例,计算 MSE 函数对 w_7 的梯度,如式(3-23)所示。

$$\delta_7=\frac{\partial\text{Error}}{\partial w_7}=\frac{\partial\text{Error}}{\partial o_1}*\frac{\partial o_1}{\partial\text{In}_{o_1}}*\frac{\partial\text{In}_{o_1}}{\partial w_7} \tag{3-23}$$

其中,

$$\frac{\partial\text{Error}}{\partial o_1}=o_1-y_1$$

$$\frac{\partial o_1}{\partial\text{In}_{o_1}}=o_1*(1-o_1)$$

$$\frac{\partial\text{In}_{o_1}}{\partial w_7}=h_1$$

误差损失在权重 $w_8\sim w_{12}$ 上的计算,与δ_7的计算类似。

对于输入层与隐藏层之间的权重,以权重 w_1 为例,计算损失函数对 w_1 的梯度,计算如式(3-24)所示。

$$\begin{aligned}\delta_1&=\frac{\partial\text{Error}}{\partial w_1}=\frac{\partial\text{Error}}{\partial o_1}*\frac{\partial o_1}{\partial w_1}+\frac{\partial\text{Error}}{\partial o_2}*\frac{\partial o_2}{\partial w_1}\\&=\frac{\partial\text{Error}}{\partial o_1}*\frac{\partial o_1}{\partial\text{In}_{o_1}}*\frac{\partial\text{In}_{o_1}}{\partial h_1}*\frac{\partial\text{In}_{h_1}}{\partial\text{In}_{h_1}}*\frac{\partial\text{In}_{h_1}}{\partial w_1}+\frac{\partial\text{Error}}{\partial o_2}*\frac{\partial o_2}{\partial\text{In}_{o_2}}*\frac{\partial\text{In}_{o_2}}{\partial h_1}*\frac{\partial h_1}{\partial\text{In}_{h_1}}*\frac{\partial\text{In}_{h_1}}{\partial w_1}\\&=\left(\frac{\partial\text{Error}}{\partial o_1}*\frac{\partial o_1}{\partial\text{In}_{o_1}}*w_7+\frac{\partial\text{Error}}{\partial o_2}*\frac{\partial o_2}{\partial\text{In}_{o_2}}*w_{10}\right)*\frac{\partial h_1}{\partial\text{In}_{h_1}}*\frac{\partial\text{In}_{h_1}}{\partial w_1}\end{aligned} \tag{3-24}$$

其中,

$$\frac{\partial h_1}{\partial\text{In}_{h_1}}=h_1*(1-h_1)$$

$$\frac{\partial\text{In}_{h_1}}{\partial w_1}=x_1$$

误差损失在权重 $w_2 \sim w_6$ 上的计算，与 δ_1 的计算类似。

根据上述公式，分别计算误差损失在各个权重上的梯度，如下。

$$\delta_7 = \frac{\partial \text{Error}}{\partial w_7} = \frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * \frac{\partial \text{In}_{o_1}}{\partial w_7} = 0.27 * 0.50 * (1 - 0.50) * 0.56 = 0.04$$

$$\delta_8 = \frac{\partial \text{Error}}{\partial w_8} = \frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * \frac{\partial \text{In}_{o_1}}{\partial w_8} = 0.27 * 0.50 * (1 - 0.50) * 0.54 = 0.04$$

$$\delta_9 = \frac{\partial \text{Error}}{\partial w_9} = \frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * \frac{\partial \text{In}_{o_1}}{\partial w_9} = 0.27 * 0.50 * (1 - 0.50) * 0.50 = 0.03$$

$$\delta_{10} = \frac{\partial \text{Error}}{\partial w_{10}} = \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * \frac{\partial \text{In}_{o_2}}{\partial w_{10}} = 0.65 * 0.58 * (1 - 0.58) * 0.56 = 0.09$$

$$\delta_{11} = \frac{\partial \text{Error}}{\partial w_{11}} = \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * \frac{\partial \text{In}_{o_2}}{\partial w_{11}} = 0.65 * 0.58 * (1 - 0.58) * 0.54 = 0.09$$

$$\delta_{12} = \frac{\partial \text{Error}}{\partial w_{12}} = \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * \frac{\partial \text{In}_{o_2}}{\partial w_{12}} = 0.65 * 0.58 * (1 - 0.58) * 0.50 = 0.08$$

$$\begin{aligned}\delta_1 &= \frac{\partial \text{Error}}{\partial w_1} = \left(\frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * w_7 + \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * w_{10}\right) * \frac{\partial h_1}{\partial \text{In}_{h_1}} * \frac{\partial \text{In}_{h_1}}{\partial w_1} \\ &= (0.27 * 0.50 * (1 - 0.50) * 0.10 + 0.65 * 0.58 * (1 - 0.58) * (-0.50)) \\ &\quad * 0.56 * (1 - 0.56) * 0.50 = -0.01\end{aligned}$$

$$\begin{aligned}\delta_4 &= \frac{\partial \text{Error}}{\partial w_4} = \left(\frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * w_7 + \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * w_{10}\right) * \frac{\partial h_1}{\partial \text{In}_{h_1}} * \frac{\partial \text{In}_{h_1}}{\partial w_4} \\ &= (0.27 * 0.50 * (1 - 0.50) * 0.10 + 0.65 * 0.58 * (1 - 0.58) * (-0.50)) \\ &\quad * 0.56 * (1 - 0.56) * 0.30 = -0.01\end{aligned}$$

$$\begin{aligned}\delta_2 &= \frac{\partial \text{Error}}{\partial w_2} = \left(\frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * w_8 + \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * w_{11}\right) * \frac{\partial h_2}{\partial \text{In}_{h_2}} * \frac{\partial \text{In}_{h_2}}{\partial w_2} \\ &= (0.27 * 0.50 * (1 - 0.50) * 0.20 + 0.65 * 0.58 * (1 - 0.58) * 0.40) \\ &\quad * 0.54 * (1 - 0.54) * 0.50 = 0.01\end{aligned}$$

$$\begin{aligned}\delta_5 &= \frac{\partial \text{Error}}{\partial w_5} = \left(\frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * w_8 + \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * w_{11}\right) * \frac{\partial h_2}{\partial \text{In}_{h_2}} * \frac{\partial \text{In}_{h_2}}{\partial w_5} \\ &= (0.27 * 0.50 * (1 - 0.50) * 0.20 + 0.65 * 0.58 * (1 - 0.58) * 0.40) \\ &\quad * 0.54 * (1 - 0.54) * 0.30 = 0.01\end{aligned}$$

$$\begin{aligned}\delta_3 &= \frac{\partial \text{Error}}{\partial w_3} = \left(\frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * w_9 + \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * w_{12}\right) * \frac{\partial h_3}{\partial \text{In}_{h_3}} * \frac{\partial \text{In}_{h_3}}{\partial w_3} \\ &= (0.27 * 0.50 * (1 - 0.50) * (-0.30) + 0.65 * 0.58 * (1 - 0.58) * 0.80) \\ &\quad * 0.50 * (1 - 0.50) * 0.50 = 0.01\end{aligned}$$

$$\begin{aligned}\delta_6 &= \frac{\partial \text{Error}}{\partial w_6} = \left(\frac{\partial \text{Error}}{\partial o_1} * \frac{\partial o_1}{\partial \text{In}_{o_1}} * w_9 + \frac{\partial \text{Error}}{\partial o_2} * \frac{\partial o_2}{\partial \text{In}_{o_2}} * w_{12}\right) * \frac{\partial h_3}{\partial \text{In}_{h_3}} * \frac{\partial \text{In}_{h_3}}{\partial w_6} \\ &= (0.27 * 0.50 * (1 - 0.50) * (-0.30) + 0.65 * 0.58 * (1 - 0.58) * 0.80) \\ &\quad * 0.50 * (1 - 0.50) * 0.30 = 0.01\end{aligned}$$

根据 MSE 函数值对各个连接权重的梯度，对权重进行调整，如式(3-25)所示。

$$w'_i = w_i - \eta * \delta_i \tag{3-25}$$

这里，w'_i是新的权重，w_i 是原有权重，δ_i 是 MSE 误差在神经元连接权重上的梯度，η 是学习率，控制对梯度的学习程度。

设 $\eta=1$，调整后的权重，如图 3-27 所示。

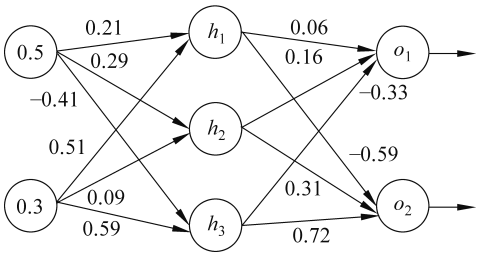


图 3-27　神经网络中调整后的权重

再次进行一次正向计算，MSE=0.22，与调整前的 0.25 比较，误差损失降低了，说明模型通过学习提高了预测或拟合精度。

多次迭代地进行正向计算→反向传播，使得误差损失值不断降低。当误差损失值基本不再变化时，认为找到了合适的模型参数，结束训练过程。

3.4　深度卷积神经网络

深度卷积神经网络(Deep Convolution Neural Network)是一种多层前馈神经网络，多通道输入的图像可以直接输入网络，经过卷积层、池化层、归一化层、全连接层等依次进行处理，输出分类或回归结果。深度卷积神经网络是机器视觉系统采用的典型深度学习方法，是在一般神经网络的基础上发展而来。与一般神经网络相比，深度卷积神经网络又有许多不同的特点，如表 3-2 所示。

表 3-2　深度卷积神经网络与一般神经网络的比较

比　　较	深度卷积神经网络	一般神经网络
网络结构	数十层及以上	一般 3 层或以内
层间连接	共享权重、部分连接	一般为全连接
目标损失函数	交叉熵(CE)损失函数	MSE 函数
激活函数	ReLU 及改进函数	Sigmoid 函数
避免过拟合	Dropout、BN 等技术	经验
参数优化	Adam	SGD

(1) 从网络结构来说，一般神经网络通常为 3 层或 3 层以内，而深度卷积神经网络大多在数十层甚至上百层。