第3章　孪生网络

第2章讲解了深度学习的基本概念，以及深度神经网络、循环神经网络、生成对抗网络、Transformer及扩散模型。本章将先介绍一种特殊的神经网络——孪生网络，它是最简单、最常用的单样本学习算法之一。介绍它是如何在样本较少的情况下开展学习的，并用于解决低数据问题(low data problem)。接着给出孪生网络的基本架构，介绍孪生网络的应用场景。最后通过一个图像识别的案例编写一个简单的孪生网络模型程序，在实践中学习孪生网络。

本章内容：

- 孪生网络简介。
- 孪生网络的架构。
- 孪生网络的衍生。
- 孪生网络的发展及应用。
- 案例：利用孪生网络进行图像识别。

3.1　孪生网络简介

孪生网络(siamese network)。siamese表示暹罗双胞胎，意为连体人。顾名思义，该网络可以理解为两个或多个神经网络在一定程度上是“连体”的。所以孪生网络又称为连体网络，网络中的连体是通过共享权重实现的。孪生网络是一种监督学习，也是一种度量学习的方法，是最简单常用的单样本学习算法之一，主要用于各类别数据点较少的应用中。一般图像分类有2种情况：第1种

情况是图片类别较少，但是每一类的数据量多。第 2 种情况是图片类别较多，但是每种类别的数量较少。对于第 1 种情况，使用深度学习网络如 CNN，或 SVM 等机器学习就可以轻易地解决。对于第 2 种情况，使用 CNN 等深度学习算法就不能达到很好的效果。现在的公司学校都在用人脸识别技术做门禁系统，要识别出一个人，就需要这个人的很多图像来训练网络，并且网络还要有良好的精度。显然人脸识别就属于第 2 种情况，即种类多而每一类的数据量少。因此，这种情况下使用孪生网络就可以很好地解决这类图像分类问题。

那么孪生网络究竟是怎样实现的呢？它又是如何判断图像是属于哪一类的呢？首先，孪生网络是用于判断两个输入值是否相似的，输入两张图片，通过对比这两张图片的相似度来判断它们是否属于同一类。孪生网络中有两个对称的神经网络，它们具有相同的权重和架构，并由损失函数连接。孪生网络有两个输入，这两个输入分别进入两个完全相同的神经网络，最后通过能量函数评价两个输入的相似度。

下面举例说明孪生网络的工作流程，如图 3-1 所示。假如要判断图片 P_1 和图片 P_2 是否相似，首先将图片 P_1 作为网络 A 的输入，将图片 P_2 作为网络 B 的输入。接着两个输入图像分别经过这两个网络后提取出各自的 feature（特征向量，又叫 embedding），由于孪生网络是权重共享的，所以网络 A 和网络 B 的结构

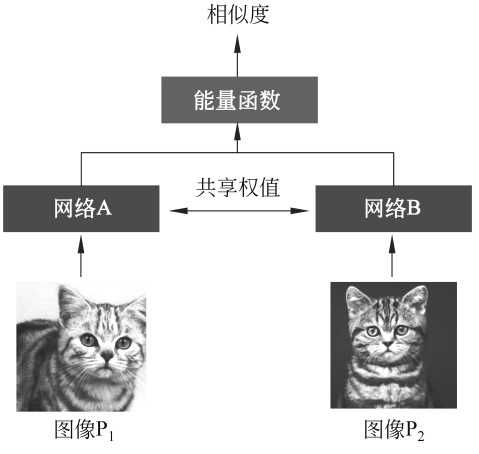

图 3-1　判断两张图片的相似度

相同，并且需要有相同的权重。之后网络 A 和网络 B 分别将输入图像 P_1 和 P_2 各自的 feature 作为能量函数的输入。最后由能量函数判断两个特征向量的距离，从而来达到判断两个输入图像是否相似的目的。这里的距离可以有很多，比如欧氏距离、余弦距离、指数距离等。最终输出两张图片的相似度。

3.2　孪生网络的架构

通过前面的学习，我们对孪生网络有了初步了解，下面详细介绍孪生网络。

孪生网络的架构如图 3-2 所示。它由两个完全相同的网络和一个能量函数组成，这两个网络具有相同的权重和架构。将 X_1 和 X_2 分别输入网络 A 和网络 B，也就是两个 $\boldsymbol{F}_w(X)$。这两个网络有着相同的权重 w，它们会分别输出 X_1 和 X_2 的特征向量，也就是 $\boldsymbol{F}_w(X_1)$ 和 $\boldsymbol{F}_w(X_2)$，并作为能量函数 E_w 的输入。能量函数 E_w 计算两个输入的距离，从而得出两个输入的相似度。能量函数 E_w 的表达式如下：

$$E_w(X_1,X_2)=\|\boldsymbol{F}_w(X_1)-\boldsymbol{F}_w(X_2)\| \tag{3-1}$$

这里使用 L_2 距离（欧氏距离）作为能量函数，当 E_w 值较小时，说明 X_1 和 X_2 相似，反之说明 x_1 和 x_2 不相似。

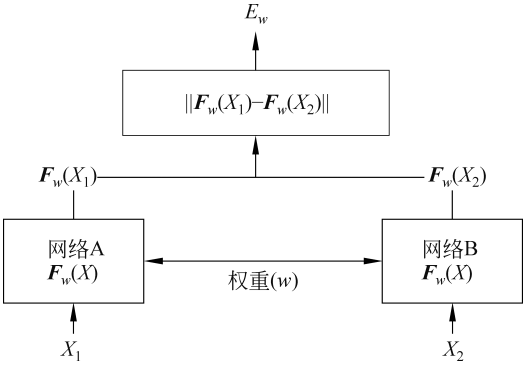

图 3-2　孪生网络的架构

上面介绍，孪生网络有两个输入，它们是成对出现的，它们的二元标签（binary label）$Y\in\{0,1\}$，代表输入对是正样本对（相似）还是负样本对（不相似）。

例如表 3-1 所示的图片样本对，第一行是正样本对(标签为 1)，第二行是负样本对(标签为 0)。

表 3-1　输入图片样本对

图片样本对		标　签
		1
		0

孪生网络的损失函数采用的是对比损失函数(contrastive loss)，目的是判断两个输入之间的相似性。这种损失函数的原理是：原本相似的样本，经过特征提取后，在特征空间内两个样本依旧相似；原本不相似的样本，经过特征提取后，在特征空间的两个样本依旧不相似。对比损失函数的表达式如下：

$$\text{Loss}=\frac{1}{2N}\sum_{n=1}^{N}[Y(E_w)^2+(1-Y)\max(0,m-E_w)^2] \tag{3-2}$$

其中，N 表示样本数量，Y 表示输入标签，当两个输入值相似时为 1；不相似时为 0。E_w 表示能量函数，可以是任何距离度量。m 表示输入样本对不相似时的距离阈值，即这两个输入值不相似时，它们的距离范围为$[0,m]$，当距离超过 m 时，这两个输入值的不相似性可看作 0，就不会导致损失。

3.3　孪生网络的衍生

孪生网络已经在图像处理领域大放光彩。随着科学技术的发展，不少专家学者在原有的网络结构基础上对其进行改进，这里简单介绍几种。

3.3.1　伪孪生网络

孪生网络是由权重和架构完全相同的神经网络组成的，如果两个网络不共享权重，或者两个网络是不同的神经网络，这种网络就叫作伪孪生网络（pseudo siamese network）。伪孪生网络的两个神经网络可以是结构相同但权重不同，也可以是完全不同结构的两个网络。如图 3-3 所示，该伪孪生网络就包含一个 LSTM 网络、一个 CNN 网络。这种伪孪生网络可以用来比对不同数据类型的信息（形式上多模态的信息）所表达的内容的相似性，如一段文字和一张图像，判断文字内容是否符合图片。

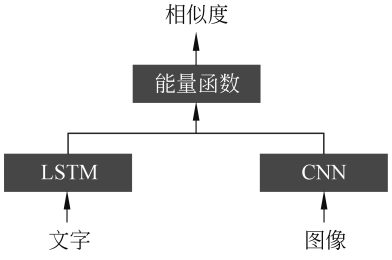

图 3-3　伪孪生网络

伪孪生网络有很多用处。如 Lloyd H. Hughes 等人利用伪孪生网络解决在非常高分辨率的光学和合成孔径雷达遥感图像中识别相应斑块的任务[1]，提出了一种具有两个独立但相同的卷积流的伪孪生网络架构，用于处理遥感图像补丁和光学补丁。损失函数上使用的是二元交叉熵损失。当然，伪孪生网络的应用还有很多，读者可以查看相关资料。

3.3.2　三胞胎连体网络

孪生网络以及伪孪生网络都是由两个网络组成的，如果换成三个网络可行吗？答案当然是可行的。Elad Hoffer 和 Nir Ailon 在论文中就提出了三胞胎网络[2]（triplet network），其结构如图 3-4 所示。

从图中可以看出，三胞胎网络有 3 个输入，分别对应 3 个网络，这 3 个输入可以是一个正样本对两个负样本对，或一个负样本对两个正样本对。三胞胎网

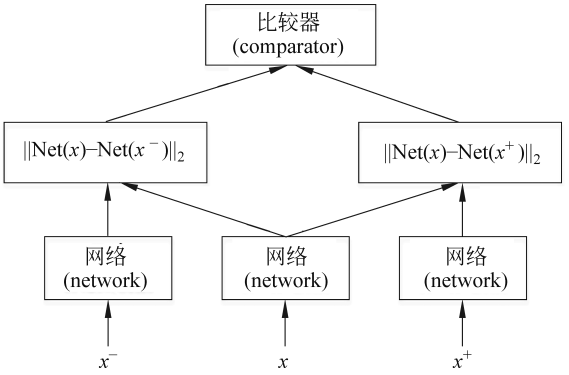

图 3-4　三胞胎网络结构

络中的这 3 个网络和孪生网络的形式比较类似，是由 3 个结构完全相同、权值共享的网络组成的。网络训练的目标是使同类别间的距离尽可能地小，不同类别间的距离尽可能地大。根据作者的经验，该网络在 MNIST 数据集上有着较优的表现，也可以作为无监督学习框架，具体内容读者可以阅读相关论文。

同样，三胞胎网络也有许多用处。比如 Yishu Liu 和 Chao Huang 就利用三胞胎孪生网络进行场景分类任务[3]。该网络由三个相同架构和相同权值的卷积神经网络组成，每个输入对应一个网络。其中两个输入是正样本，第三个是负样本。它们构造了 4 个新的损失函数来提高分类精度。感兴趣的读者可以阅读相关论文。

3.3.3　三胞胎伪孪生网络

孪生网络就是这么神奇，不同的权值、更多的子网络都会使孪生网络变得更加强大，如果把这些改变都加入其中，会变成什么样呢？李光正等人在论文[4]中使用了三胞胎伪孪生网络来检测不同的焊接缺陷或动作，如图 3-5 所示。这个网络由三个不同的网络组成。该网络有 3 个输入，分别输入图像、声音以及电流电压 3 种不同的数据。3 种子网络使用的是 3 种改进的卷积神经网络，使用了跨模态注意机制（cross-modal attention，CMA）来完成图像、声音和电流电压之间的交互。该网络详细构造非常复杂，读者可以阅读论文学习，这里不再介绍。

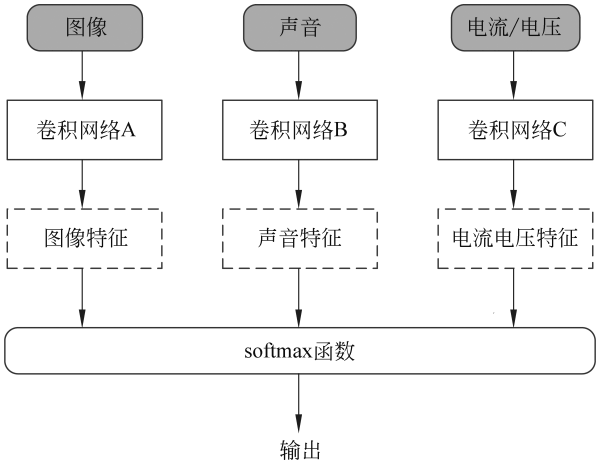

图 3-5　三胞胎伪孪生网络结构示意图

3.4　孪生网络的发展及应用

以上介绍表明，孪生网络通过寻找两个输入值之间的相似性来学习。因此该网络主要应用于需要对比两个输入之间相似性的任务中去。这样的应用很多，并且许多领域都有涉及。

早在 1993 年，Jane Bromley 等人在论文[5]中提出了孪生网络，用于验证支票上的签名与银行预留签名是否一致。作者收集了 5990 个签名数据用于识别签名的真实性，数据分为正样本签名和负样本签名，用于训练孪生网络。使用时间延迟网络（time delay neural network，TDNN）作为孪生网络的两个子网络。在训练过程中，两个子网络从两个签名中提取特征，然后计算两个特征向量夹角的余弦值作为距离值。要识别一个新签名时，提取这个签名的特征向量与存储的签名者特征向量比较，如果距离小于设定阈值，则说明这个签名是真实的。

之后由于当时技术条件的限制，孪生网络的发展几乎停滞不前。直到 2010 年，Hinton 在 ICML 上发表了 *Rectified Linear Units Improve Restricted Boltzmann Machines*[6]。他使用孪生网络做人脸识别，判断两张人脸图像是否

相似。采用两个 Noisy Rectified Linear Unit (NReLU)作为两个子网络,能量函数选用的是余弦距离。2015 年,Sergey Zagoruyko 在 *Learning to Compare Image Patches via Convolutional Neural Networks*[7]一文中介绍了几种改进的孪生网络,并做了对比。他在以 CNN 为子网络的孪生网络的基础上借鉴了双通道(2-channel)网络、空间金字塔池化(SPP)网络以及双流网络(two-stream)的结构,对孪生网络进行改进,用于图片相似度对比,并做了对比试验。在图像匹配上,Iaroslav Melekhov 等人使用孪生网络对世界各地的地标图片进行图像匹配[8]。他们使用两个卷积神经网络作为孪生网络的子网络,使用欧氏距离作为能量函数,损失函数采用的是基于边际的对比损失函数,用于计算图片之间的相似度。

随着孪生网络发展的日益成熟,孪生网络也开始慢慢应用在计算机视觉、目标跟踪领域和自然语言处理等领域。Luca Bertinetto 等人提出了一种新的全卷积孪生网络,用于视频中的目标检测[9]。Xingping Dong 和 Jianbing Shen 将一种新的三重损失加入孪生网络框架中,用于目标跟踪[10]。Jonas Mueller 和 Aditya Thyagarajan 等人提出通过两个 LSTM 网络作为孪生网络中的两个子网络来处理句子对[11]。使用曼哈顿距离来度量两个句子的空间相似度,从而计算两个句子之间的相似度。

孪生网络的应用相当广泛。此外,孪生网络除了可以单独使用外,还可以组装在各种网络架构中,用于组成适合不同任务的模型。

3.5 案例:利用孪生网络进行图像识别

以上介绍了孪生网络的结构及应用。接下来从实战出发,通过一个图像识别的案例动手训练一个简单的孪生网络,利用该网络判断两张图片是否相似,进而识别出该图像的类别。

案例中使用的数据集为 Fashion-MNIST 数据集,它由德国公司 Zalando 旗下的研究部门提供。该数据涵盖了 10 种类别的共 70000 个不同商品的正面图

片，其中训练集包含 60000 个样本，测试集包含 10000 个样本。样本来自日常穿着的衣裤鞋包，每一个都是 28×28 的灰度图像，如图 3-6 所示。

图 3-6　Fashion-MNIST 数据集中前 24 张图片

接下来需要创建训练数据。由于孪生网络有两个输入，因此训练数据必须成对并带有标签。从相同类别中随机选取两张图片作为**正样本对**，从一个类别中选出一张图片与其他类别中的一张图片组成**负样本对**。如表 3-2 所示，正样本对的两张图片是同一类别，负样本对的两张图片是不同类别。

之后就开始构建孪生网络。创建两个卷积网络，用于提取特征向量，两个网络的激活函数使用线性整流函数（ReLU）。将图像对中的两个图片分别输入两个卷积网络中，输出提取的特征向量。之后把这两个特征向量作为能量函数的输入，输出两张图片的相似度。

下面根据案例一步一步地训练一个孪生网络。该案例可以在提供的源代码中查看具体代码。

表 3-2　输入样本对

输　入　对		标　签
		正
		负
		正
		负

（1）导入需要的库。

```
import random
import tensorflow as tf
from tensorflow import keras
from keras.layers import Input, Flatten, Dense, Dropout, Lambda, MaxPooling2D
from keras.models import Model
from keras.optimizers import RMSprop
from keras import backend as K
from keras.layers.convolutional import Conv2D
from keras.layers import LeakyReLU
from keras.regularizers import l2
from keras.models import Model, Sequential
from tensorflow.keras import regularizers
import numpy as np
import matplotlib.pyplot as plt
```

（2）加载数据。

使用 Fashion-MNIST 数据集，该数据已在 keras 数据集中有所包含，可以直接使用以下代码加载数据而不用提前下载。

```
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train = x_train / 255.0
x_test = x_test / 255.0
```

显示数据集的第一张图片，如图 3-7 所示。

```
plt.figure(figsize=(5,5))
plt.imshow(x_train[0], cmap=plt.cm.binary)
plt.xticks([])
plt.yticks([])
plt.grid(False)
```

图 3-7　Fashion-MNIST 数据集中的一张图片

按照数据标签对数据进行分类。选取“上衣”“裤子”“套头衫”“外套”“凉鞋”“靴子”6 个类别。划分训练集和测试集，比例为 8∶2。

```
digit_indices = [np.where(y_train == i)[0] for i in {0,1,2,4,5,9}]
digit_indices = np.array(digit_indices)
n = min([len(digit_indices[d]) for d in range(6)])
train_set_shape = n * 0.8
test_set_shape = n * 0.2
y_train_new = digit_indices[:, :int(train_set_shape)]
y_test_new = digit_indices[:, int(train_set_shape):]
print(y_train_new.shape)
print(y_test_new.shape)test_set_shape = n * 0.2
```

(3) 制作训练数据。

定义 create_pairs 函数来生成数据。前面讲到,Siamese 网络的输入数据应该是成对存在的(正样本和负样本)。按照上面已经分好的类别,从同一个类别中选取图像($z1$,$z2$),并存储到 pairs 数组中。同时从不同类别中选取图像($z1$,$z2$),同样存储到 pairs 数组中。此时该条样本中包含一正、一负,将 labels 赋值为[1,0]。最终生成了训练数据,包含训练集和测试集。

```
def create_pairs(x, digit_indices):
    pairs = []
    #标签为 1 或 0,用于标识样本对是正的还是负的
    labels = []
    class_num = digit_indices.shape[0]
    for d in range(class_num):
        for i in range(int(digit_indices.shape[1]) - 1):
        #使用来自同一类的图像来创建正样本对
        z1, z2 = digit_indices[d][i], digit_indices[d][i + 1]
        pairs += [[x[z1], x[z2]]]
        #使用随机数从另一个类中找到图像来创建负样本对
        inc = random.randrange(1, class_num)
        dn = (d + inc) % class_num
        z1, z2 = digit_indices[d][i], digit_indices[dn][i]
        pairs += [[x[z1], x[z2]]]
        #add two labels which the first one is positive class and the second is
        #negative
        labels += [1, 0]
    return np.array(pairs), np.array(labels)
#训练集
tr_pairs, tr_y = create_pairs(x_train, y_train_new)
tr_pairs = tr_pairs.reshape(tr_pairs.shape[0], 2, 28, 28, 1)
#测试集
te_pairs_1, te_y_1 = create_pairs(x_train, y_test_new)
te_pairs_1 = te_pairs_1.reshape(te_pairs_1.shape[0], 2, 28, 28, 1)
```

(4) 构建孪生网络并训练模型。

先建立基本网络,它是一个用于特征向量提取的卷积网络。用 ReLU 为激活函数构建两个卷积层和一个平面层。

```
def create_base_network(input_shape):
    input = Input(shape=input_shape)
```

```
    x = Conv2D(32, (7, 7), activation='relu', input_shape=input_shape,
                           kernel_regularizer=regularizers.l2(0.01),
                           bias_regularizer=regularizers.l1(0.01))(input)
    x = MaxPooling2D()(x)
    x = Conv2D(64, (3, 3), activation='relu', kernel_regularizer=regularizers.
                           l2(0.01), bias_regularizer=regularizers.l1(0.01))(x)
    x = Flatten()(x)
    x = Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.01),
                   bias_regularizer=regularizers.l1(0.01))(x)
    return Model(input, x)
```

接下来，将图像对输入到基础网络中，它将返回 Embeddings，即特征向量。

```
input_shape = (28,28,1)
base_network = create_base_network(input_shape)
input_a = Input(shape=input_shape)
input_b = Input(shape=input_shape)
processed_a = base_network(input_a)
processed_b = base_network(input_b)
```

processed_a 和 processed_b 是图像对的特征向量。将这些特征向量提供给能量函数来计算它们之间的距离，这里使用欧氏距离作为能量函数。同时给出了损失函数 contrastive_loss，并定义精确度。

```
#距离函数
def euclidean_distance(vects):
    x, y = vects
    sum_square = K.sum(K.square(x - y), axis=1, keepdims=True)
    return K.sqrt(K.maximum(sum_square, K.epsilon()))
#输出类型函数
def eucl_dist_output_shape(shapes):
    shape1, shape2 = shapes
    return (shape1[0], 1)
#损失函数
def contrastive_loss(y_true, y_pred):
    margin = 1
    square_pred = K.square(y_pred)
    margin_square = K.square(K.maximum(margin - y_pred, 0))
    return K.mean(y_true * square_pred + (1 - y_true) * margin_square)
#精确度函数
def accuracy(y_true, y_pred):
```

```
    #Compute classification accuracy with a fixed threshold on distances.
    return K.mean(K.equal(y_true, K.cast(y_pred < 0.5, y_true.dtype)))
distance = Lambda(euclidean_distance, output_shape=eucl_dist_output_shape)
                  ([processed_a, processed_b])
```

接下来设置轮数(epoch)为 13,并使用 RMsprop 进行优化。之后定义模型 model。

```
epochs = 13
rms = RMSprop()
model = Model([input_a, input_b], distance)
model.compile(loss=contrastive_loss, optimizer=rms, metrics=[accuracy])
```

所有都准备好后,就可以开始训练模型。

```
tr_y = np.array(tr_y, dtype='float32')
results = model.fit([tr_pairs[:, 0], tr_pairs[:, 1]], tr_y, batch_size=128,
epochs=epochs, verbose=2, validation_split=.25)
```

可以绘制出图像来查看模型损失的变化,如图 3-8 所示。

```
plt.plot(results.history['loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.show()
```

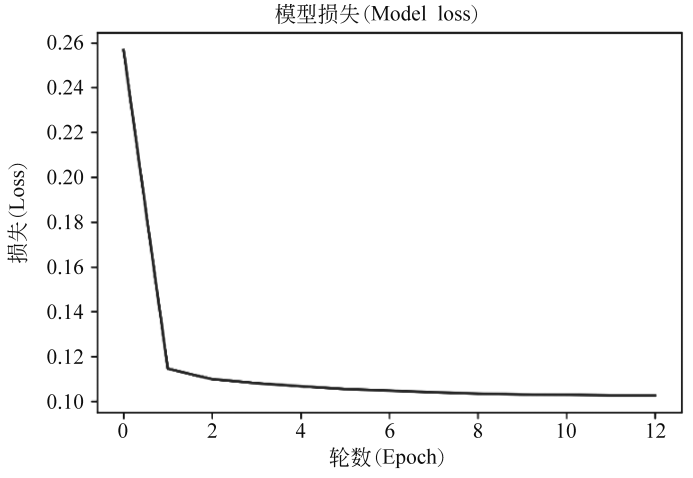

图 3-8　模型损失变化

可以看到，随着训练轮数的增加，损失在不断减少。

(5) 预测及评估。

训练好模型后，就可以用测试集来预测。

```
y_pred = model.predict([te_pairs_1[:, 0], te_pairs_1[:, 1]])
```

定义精确度计算函数，查看模型的准确性。

```
#定义精确度函数
def compute_accuracy(y_true, y_pred):
    pred = y_pred.ravel() < 0.5
    return np.mean(pred == y_true)
```

计算模型的准确性，并输出。

```
te_acc = compute_accuracy(te_y_1, y_pred)
print('Accuracy on test set: %0.2f%%' % (100 * te_acc))
```

输出：Accuracy on test set：92.19%。

3.6　小　　结

本章讲解了孪生网络是用于判断两个输入相似性的一种网络，以及孪生网络是如何判断两个输入的相似性的。它是由结构相同、权值共享的两个神经网络组成的，通过这两个网络提取出特征向量，并输入到能量函数中计算相似性。最后讲解了孪生网络的一些常用应用，并通过一个案例动手实现了一个简单的孪生网络模型。

3.7　思　考　题

1. 详细解释一下孪生网络的基本概念及其在各种场景中的应用。
2. 详述孪生网络如何利用其特定的设计来判断两个输入样本的相似性。
3. 详细描述孪生网络的典型结构，以及这种结构对网络性能的影响。

4. 在孪生网络中，两个子网络的权值是否始终保持相同？这种设计背后的逻辑是什么？

5. 列举一些在孪生网络基础上发展出来的网络结构，并简述其特点。

参 考 文 献

[1] Hughes L H，Schmitt M，Mou L，et al. Identifying corresponding patches in SAR and optical images with a pseudo-siamese CNN[J]. IEEE geoscience and remote sensing letters，2018，15(5)：784-788.

[2] Hoffer E，Ailon N. Deep metric learning using triplet network [C]//Similarity-based pattern recognition：third international workshop，SIMBAD 2015，Copenhagen，Denmark，October 12-14，2015. proceedings 3. springer international publishing，2015：84-92.

[3] Liu Y，Huang C. Scene classification via triplet networks[J]. IEEE Journal of selected topics in applied earth observations and remote sensing，2017，11(1)：220-237.

[4] Li Z，Chen H，Ma X，et al. Triple pseudo-siamese network with hybrid attention mechanism for welding defect detection[J]. Materials & Design，2022(217)：110645.

[5] Bromley J，Guyon I，LeCun Y，et al. Signature verification using a "siamese" time delay neural network [J]. Advances in neural information processing systems，1993(6)：669-688.

[6] Nair V，Hinton G E. Rectified linear units improve restricted Boltzmann machines[C]//Proceedings of the 27th international conference on machine learning (ICML-10). 2010：807-814.

[7] Zagoruyko S，Komodakis N. Learning to compare image patches via convolutional neural networks [C]//Proceedings of the IEEE conference on computer vision and pattern recognition，2015：4353-4361.

[8] Melekhov I，Kannala J，Rahtu E. Siamese network features for image matching [C]//2016 23rd international conference on pattern recognition (ICPR). IEEE，2016：378-383.

[9] Bertinetto L，Valmadre J，Henriques J F，et al. Fully-convolutional siamese networks for object tracking [C]//Computer vision - eccv 2016 workshops：amsterdam，the netherlands，October 8-10 and 15-16，2016，proceedings，part II 14. springer international publishing，2016：850-865.

[10] Dong X，Shen J. Triplet loss in siamese network for object tracking[C]//Proceedings of the European conference on computer vision (ECCV). 2018：459-474.

[11] Mueller J，Thyagarajan A. Siamese recurrent architectures for learning sentence similarity[C]//Proceedings of the AAAI conference on artificial intelligence，2016：30(1)：2786-2792.