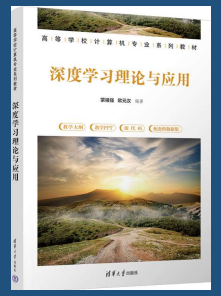
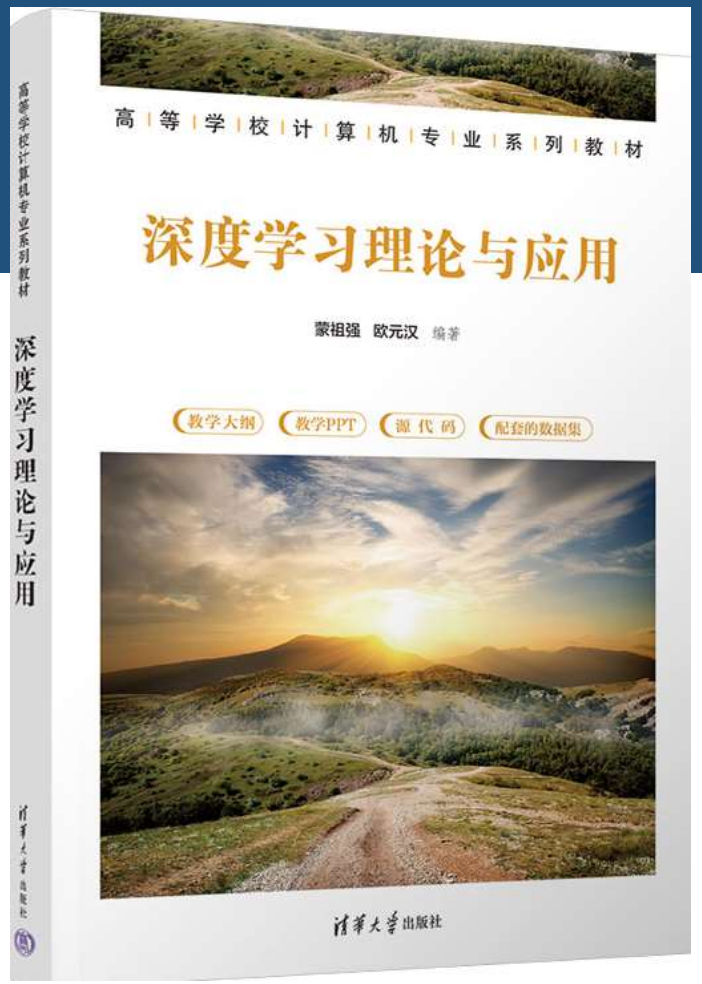


深度学习理论与应用

Deep Learning Theory and Applications

蒙祖强，欧元汉 编著



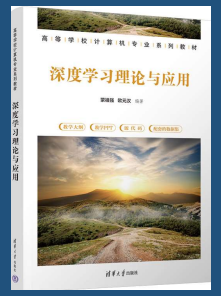
教　材

全国各大书店网店均有销售

- **教学大纲**：提供面向教育工程认证的教学大纲
- **教学PPT**：提供课堂教学用的PPT课件
- **源代码**：提供教材涉及的全部源代码
- **数据集**：提供教材示例、案例用到的全部数据集

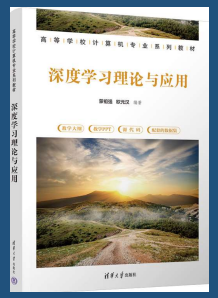
获取教学资源：
http://www.tup.tsinghua.edu.cn/booksCenter/book_09988101.html

教材： 蒙祖强，欧元汉 编著. 深度学习理论与应用. 北京: 清华大学出版社，2023年7月. (书号: 978-7-302-63508-6)



第10章　多模态学习与多模态数据分类

蒙祖强，欧元汉 编著. 深度学习理论与应用. 北京: 清华大学出版社，2023年7月.



本 章 内 容

contents

10.1 多模态学习

10.1.1 多模态学习的发展过程

多模态学习（Multimodal Learning）：旨在建立能够处理和关联多种模态信息的模型和方法，是对多源异构的多模 态数据进行分析和挖掘的一种机器学习，也称为多视角学习（Multi-view Learning）。在多模态学习方面，早期比较有影响的工作是Petajan 等人开发的唇语–声音语音识别系统，获得了较高的识别准确率。也有的学者设计了基于手势、声音、语义三模态的机器感知系统，并用于人机交互场景。

10.1 多模态学习

10.1.1 多模态学习的发展过程

- **深度学习技术的迅速发展：**深度卷积神经网络 （CNN）在图像分类任务中取得了超越人类的表现，具有代表性的成果包括 AlexNet、ResNet 和 GoogleNet 等大模型；循环神经网络（RNN，包括 LSTM、GRU 等）以及基于 Transformer 架构的 GPT、BERT 等预训练模型则在自然语言处理方面取得了巨大的进步。
- **系统的多模态学习：**始于 Ngiam 等发表于 ICML 2011 的《Multimodal Deep Learning》。该工作利用深度学习模型对视频、音频数据进行编码，形成多模态数 据的联合表示，进而实现对各模态的识别。
- **近几年的发展：**多模态学习已成为机器学习、数据挖掘领域的研究热点之一，在图像描述、图像检索、视频检索、多模态信息摘要生成、多模态 情感分析、机器翻译、多模态人机对话等方面得到了广泛应用。

10.1 多模态学习

10.1.2 多模态学习的主要任务

1. 多模态传译

模态传译：将一种模态数据翻译或映射为另一种模态数据，翻译前后两种模态所蕴含的语义信息是一样的。模态传译实际上是实现信息在不同模态间的流通，主要包括图像和文本、图像和图像、文本和文本、视频和语音、语音和文本等模态之间的模态传译。

一个例子：图像描述是根据输入的图片自动生成其对应的文本描述，属于“图像到文本”这一类模态传译。在基于深度神经网络的方法中，图像描述任务通常采用 CNN+RNN 架构（编码-解码结构）。也就是说，用卷积神经网络提取图像的特征向量，然后将特征向量输入到一种RNN中去解码，形成文字输出，即为该图像的描述。

10.1 多模态学习

10.1.2 多模态学习的主要任务

2. 多模态传译对齐

模态对齐：指辨别不同模态中元素之间的对应关系。

一个例子：我们希望视频中人物讲话和文本字幕同步，这是视频和文本这两种模态之间的对齐。在翻译任务中，我们希望源语言和目标语言之间对应的词汇要关联起来，如在将“I am a student”翻译为“我是一位学生”时，“I”、“am”、“a”、“student”应该分别关联到“我”、“是”、“一位”、“学生”。实际上，这是一种细粒度的对齐，根据实现对齐方式的不同，模态对齐可以分为注意力对齐和语义对齐。

10.1 多模态学习

10.1.2 多模态学习的主要任务

2. 多模态对齐

注意力机制：在注意力对齐方式中，主要考虑当前生成的目标模态元素一般跟输入模态中的某些元素关联程度大，因而用注意力机制通过加权求和方法来实现这种关联，它可以较好解决长程依赖问题。注意力对齐多应用于涉及模态传译的多模态学习任务，如机器翻译、图像标注、语音识别等。实际上，不仅在模态对齐和模态传译任务中，而且在多模态表征学习、数据融合、数据分类等领域中 注意力机制均有良好的表现。

10.1 多模态学习

10.1.2 多模态学习的主要任务

多模态数据分类：指利用各模态数据构成的数据集训练一个分类器， 然后用该分类器对新的多模态数据进行类别预测。

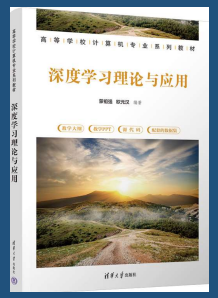
- Truong 等人提出了一种称为视觉注意力网络的情感分析新方法，实际上就是一种多模态情感数据分类。还有的文献通过特征融合方法提出了一个统一的网络来共同学习图像和文本之间的联合表征，并可以在部分模态缺失环境下实现多模态数据分类任务。
- 多模态数据分类过程通常包括数据预处理、 特征提取、特征学习和分类四个步骤。特征提取一般用神经网络（如 CNN 或 RNN 等）来 完成。在特征学习中，一般用到前面提及的特征融合方法来实现，而这种融合的前提是用于 特征提取的模态数据应先对齐，这对数据预处理提出了比较高的要求。

10.1 多模态学习

10.1.2 多模态学习的主要任务

多模态联合学习：由于存在模态表示强弱不一致、缺少标注模态数据、噪声数据等问题，需要用一个模态知识来辅助另一个资源匮乏的模态进行建模。这时就需要利用多模态联合学习方法，其应用场景主要分为两类：平行数据和非平行数据。平行数据是指已经对齐的多模态数据，而非平行数据则是指未对齐或存在模态缺失的多模态数据。

基本思想：对各模态，先分别用带标签的少量数据样本训练各自的分类器，然后用这些分类器预测各自模态内无标签的数据，并用置信度较高的分类结果作为相应未标注数据的标签，接着将新标注的数据连同原来数据重新训练各自的分类器。多次重复这种训练，当达到一定条件时，交互模态数据，继续训练这些分类器。



本 章 内 容

contents

10.2 多模态数据分类

10.2.1 文本特征提取方法

文本：典型的序列数据，通常用循环神经网络对其进行建模和表示。

处理文本的主要步骤：

（1）对文本进行必要的清洗，然后对其进行分词或 token 化。对于英文文本，一般以空格为分隔符，将句子切分为单词序列。如果使用预训练模型（如 BERT），则由预训练模型自带的工具和词表对其进行 token 化。对于中文文本，可以使用 jieba 等工具对其进行分 词，形成词汇序列。如果使用预训练模型，则由预训练模型提供的工具对其进行 token 化， 形成 token 序列。

10.2 多模态数据分类

10.2.1 文本特征提取方法

处理文本的主要步骤：

（2）建立词表并对文本进行索引编码和等长化表示，进而转化为张量。等长化是转化为张量的基本前提，这在索引编码之前或之后做都可以。

（3）对编码形成的索引向量进行嵌入表示。这可以利用 Word2vec 或词嵌入技术来实现。

（4）利用循环神经网络提取文本的特征。将形成的词嵌入表示输入到相应的神经网络，以网络的相应输出作为文本的特征向量。LSTM、GRU 以及预训练模型 BERT 等都可以作为 神经网络来提取文本的特征。

10.2 多模态数据分类

10.2.2 图像特征提取方法

基本方法：使用卷积神经网络。卷积神经网络几乎可以实现端到端的数据处理功能，所以在提取图像特征时主要是变换图像的尺寸，然后进行张量化和数据打包，接着送入卷积神经网络，所需步骤要比提取文本特征时少许多。

卷积神经网络的输出通常是包含若干个通道的特征图，这时只要对特征图进行扁平化即可将其转变为向量（针对单张输入图像而言），这个向量即为输入图像的特征向量。

10.2 多模态数据分类

10.2.3 多模态数据融合方法

多模态数据融合方法：基于模型的融合方法和模型无关的融合方法。

- 基于**模型**的融合方法：指融合的过程和方式与具体的任务有关，因而与所设计的模型结构有关。例如，视觉问答、多模态对话系统等都是采用此类融合方法。
- **模型无关**的融合方法：指多模态数据融合算法与其所依赖的模型无关，适合于任何的特征提取网络。传统的多模态数据融合方法一般是指这一类的融合方法，可进一步分为特征级 融合、决策级融合以及混合融合方法等。

10.2 多模态数据分类

10.2.3 多模态数据融合方法

1. 特征级融合

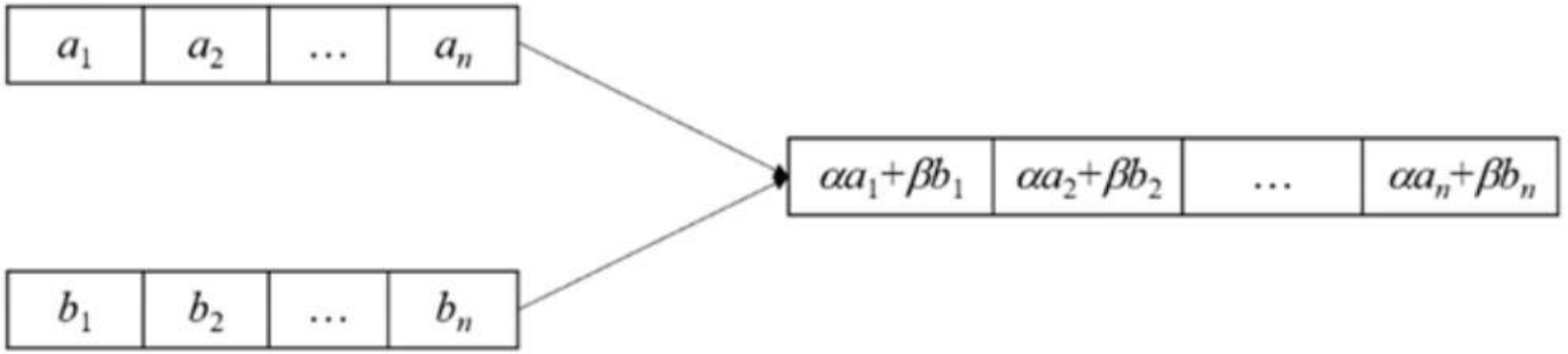
在对图像和文本提取特征以后，将这两种特征通过一定的方式融合在一起，从而实现多模态数据在**特征级上的融合**。这种融合方式分为两种：

- 线性组合（加权组合）
- 拼接方式

10.2 多模态数据分类

10.2.3 多模态数据融合方法

一般情况：假设两种模态数据在提取特征后分别得到向量(a1, a2, …, an)和向量(b1, b2, …, bm)。如果两个向量的长度相等，即 n = m，则可以使用**线性组合**方式进行融合。如下图所示。

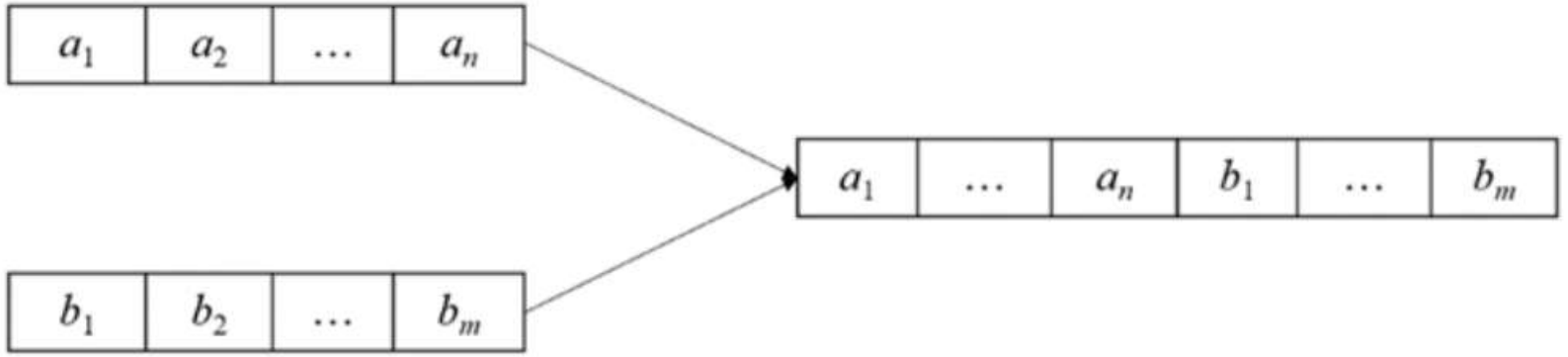


图中，α和β多为[0, 1]范围内的权值系数，可根据模态的重要性来设置，或者利用注意力机制通过学习产生。例如，如果令α=β=0.5，则相当于取两个向量的平均值。

10.2 多模态数据分类

10.2.3 多模态数据融合方法

如果两个向量的长度不相等，即 n ≠ m，一般采用**拼接的方式**进行融合。在 n = m 的情况下也可以运用这种方式，要视具体的效果而定。如下图所示。



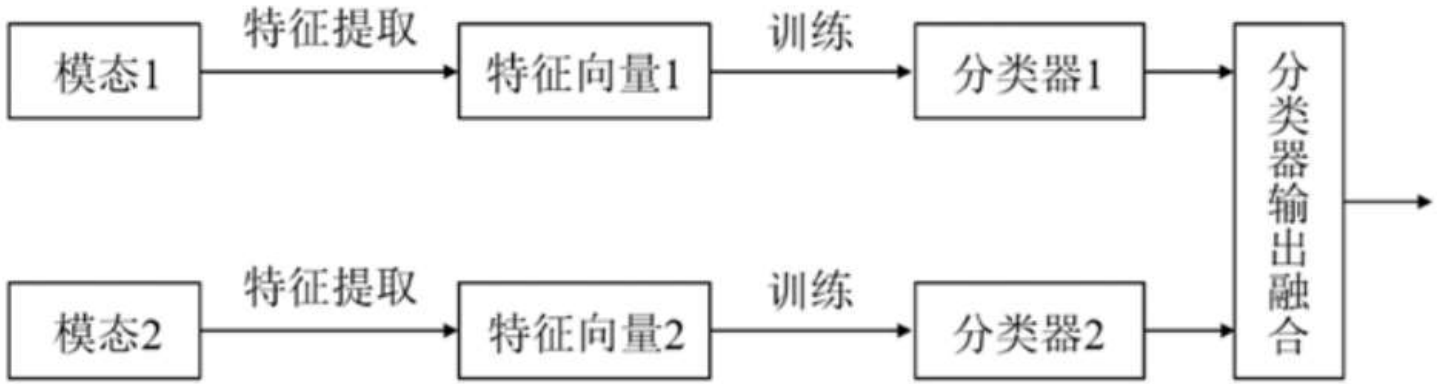
拼接方式是将两个向量的“尾”和“首”连在一起，从而产生长度为 n + m 的向量。

10.2 多模态数据分类

10.2.3 多模态数据融合方法

2. 决策级融合

基本思路：对各模态分别提取特征，然后分别训练各自的分类器，最后在分类器输出上通过投票、取最大值、线性组合等方式综合各个分类器的结果，形成最终的输出。在决策级融合方式下，当某一个模态缺失时可用另一模态数据继续训练分类器，因而决策级融合可以在模态缺失的情况下继续工作。



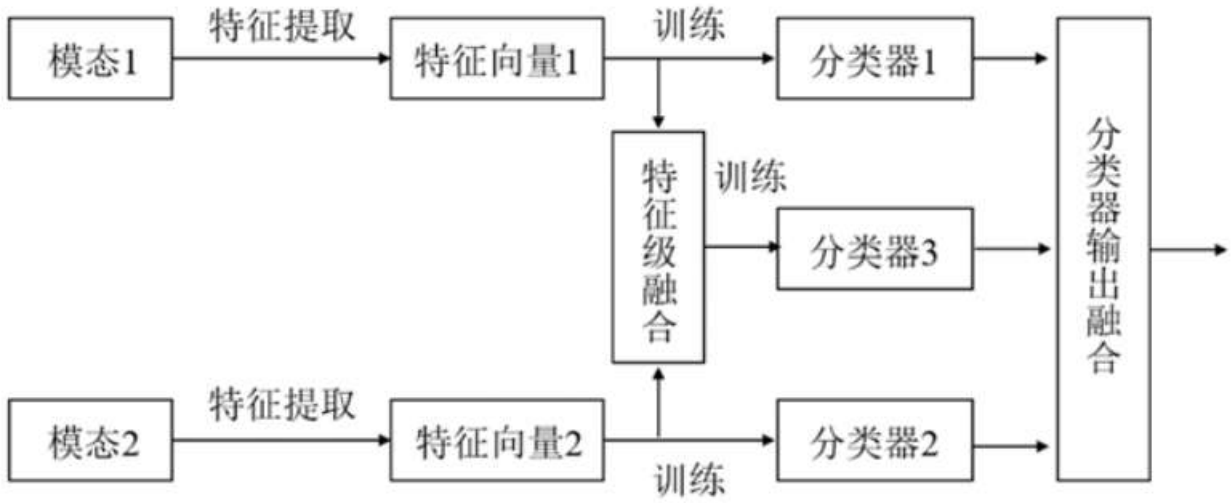
10.2 多模态数据分类

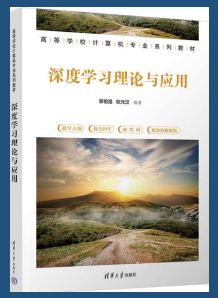
10.2.3 多模态数据融合方法

3. 混合融合方法

混合融合：将特征级融合和决策级融合综合在一起，进而形成多级别、多层次的融合方法，称为混合融合方法。混合融合方法的基本架构可用右图 表示。

特点：混合融合方法综合了两种方法的优点，同时继承了它们的一些缺点，其中调试困难、训练时间长是比较突出的问题。





本章内容

contents

10.3 多模态数据分类案例

【例 10.1】利用特征级融合方法，实现一个图-文二模态数据的分类程序。

本例使用的多模态数据集来自网站 https://mcrlab.net/research/mvsa-sentiment-analysis-on-multi-view-social-data/。从该网站上下载文件 MVSA_Single.rar，解压后产生两个文件：ID-label.txt 和 labelResultAll.txt，以及一个目 录 data。目录 data 中存放了图像文件和文本文件，它们按文件名关联。例如文件 3.jpg 和文 件 3.txt 是对应的，前者是图像文件，后者是对应的文本文件，它们是一个图像-文本对。文件ID-label.txt则存放各图像-文本对的ID号和类别信息，用 0, 1 和 2 分别表示三个类别。我们按照 8:2 将 ID-label.txt 随机分为文件 ID-label-train.txt 和文件 ID-label-test.txt，然后用这两个文件分别构建训练集和测试集。

10.3 多模态数据分类案例

（1）读取文本文件，对文本进行 token 化，并进行索引编码和等长化，同时读取图像 文件，转化为张量。为此，首先定义函数 readTxtFile(fn)，令其读取由 fn 指定文件名的文本文件。

```
def readTxtFile(fn): #读指定文件的内容
    fg = open(fn, 'r', encoding='gb18030') #读中文文本
    text = list(fg)
    assert len(text)==1
    text = text[0]
    text = text.replace('\n',‘’)
    return text
```

10.3 多模态数据分类案例

定义函数 get_txt_img_lb(path,txtname)，其作用是读取文件 ID-label-train.txt 或文 件 ID-label-test.txt 中的信息，该函数返回由文本、图像文件名和类别标记构成的数据集，其 中 txtname 为 ID-label-train.txt 或 ID-label-test.txt。

```
def get_txt_img_lb(path,txtname): #获取由“文本-图像路径-类别标记”构成的数据集
    fn = path+'\\'+txtname
    fg = open(fn, encoding='utf-8’)
    samples = []
    for line in fg:
        line = line.strip()
        if 'ID' in line:
            continue
        file_id, label = line.split(',’)
        text_path = path + '\\data\\' + file_id + '.txt’
        img_path = path + '\\data\\' + file_id + '.jpg’
        text = readTxtFile(text_path)
        item = (text,img_path,label)
        samples.append(item )
    return samples
```

10.3 多模态数据分类案例

定义数据集类 MyDataSet，在该类中，调用 AutoTokenizer 对文本进行索引编码和等长化，同时构造句子掩码矩阵和注意力掩码矩阵，此外还读取图像文件并调整图像的形状，转化为张量。

```
def __getitem__(self, idx):
    text, img_path, label = self.samples[idx]
    text_list = [text]
    #索引编码:
    txtdata =tokenizer.batch_encode_plus(batch_text_or_text_pairs=text_list,
    truncation=True, padding='max_length', max_length=128, #固定长度为 128 return_tensors='pt', return_length=True)
    input_ids = txtdata['input_ids']
    token_type_ids = txtdata['token_type_ids']
    attention_mask = txtdata['attention_mask']
    img = Image.open(img_path)
    if img.mode != 'RGB':
        print('不是 RGB 图像！')
        exit(0)
    img = tsf(img) #改变形状为 torch.Size([3, 224, 224])
    label = int(label)
    return input_ids[0],token_type_ids[0],attention_mask[0], img, label
```

10.3 多模态数据分类案例

（2）定义神经网络类 Multi_Model。该类利用预训练模型 AlbertModel，并基于文本的 索引编码提取文本的特征；通过微调，利用 EfficientNet 提取图像的特征。然后采用拼接融 合方法对文本和图像的特征进行融合，最后送入全连接网络进行分类。

```
#加载预训练模型 Bert:
bert_model = AlbertModel.from_pretrained('albert-base-v2', \
                    cache_dir="./AlBert_model").to(device)
#加载预训练模型 EfficientNet:
effi_model = EfficientNet.from_pretrained('efficientnet-
b7').to(device)
for e in effi_model.parameters():
        e.requires_grad = False #冻结参数
effi_model._fc = nn.Linear(2560, 768) #修改预训练模型的输出层
```

10.3 多模态数据分类案例


```
class Multi_Model(nn.Module): #定义深度神经网络模型类
    def __init__(self):
        super().__init__()
        self.bert_model = bert_model
        self.effi_model = effi_model
        self.fc = nn.Linear(768 + 768, 3)
        def __init__(self):
        def forward(self,data): input_ids, token_type_ids, attention_mask, img, _ = data input_ids, token_type_ids,
        attention_mask, img = input_ids.to(device), \ token_type_ids.to(device), attention_mask.to(device),
        img.to(device) outputs = self.bert_model(input_ids=input_ids, #输入文本的索引编码
                                                 attention_mask=attention_mask, token_type_ids=token_type_ids)
        text_feature = outputs[1] #文本的特征，形状为 torch.Size([8, 768])
         effi_outputs = self.effi_model(img) #图像的特征，形状为 torch.Size([8, 768])
         #采用拼接融合方式，cat_feature 的形状为 torch.Size([8, 1536]):
         cat_feature = torch.cat([text_feature, effi_outputs], -1)
         out = self.fc(cat_feature) # torch.Size([16, 3])
         return out
```

10.3 多模态数据分类案例

（3）定义函数 train()，用于对模型进行训练，代码如下：

```
def train(model:Multi_Model, data_loader): #对模型进行训练
    optimizer = optim.Adam(model.parameters(), lr=1e-4, weight_decay=1e-6, \ amsgrad=False)
    scheduler = CosineAnnealingWarmRestarts(optimizer, T_0=10,\T_mult=1, eta_min=1e-6, last_epoch=-1)
    criterion = nn.CrossEntropyLoss()
    lr = scheduler.get_last_lr()[0]
    print('epochs :0 lr:{}'.format(lr))
    print('训练中.....')
    epochs = 11
    for ep in range(epochs):
        for k,data in enumerate(data_loader):
            input_ids, token_type_ids, attention_mask, img, label = data
            label = label.to(device)
            pre_y = model(data)
            loss = criterion(pre_y, label)
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()
    scheduler.step()
```

10.3 多模态数据分类案例


```
lr = scheduler.get_last_lr()[0]
if not ep + 1 == epochs:
        print('epochs :{} lr:{:.6f}'.format(ep + 1, lr))
if ep % 5 == 0: #每 5 轮循环保存一次模型参数
        torch.save({'model_state_dict': model.state_dict()}, f'multi_model_new.pt')
        check_point = torch.load(f'multi_model_new.pt')
        model.load_state_dict(check_point['model_state_dict'])
torch.save({'model_state_dict': model.state_dict()}, f'multi_model_new.pt')
 print('训练完毕！ ')
return None
```

训练过程使用了学习率衰减技术，实际上是一种学习率的周期性循环衰减方法。其中，初始学习率设置为 1e-4，最小学习率为 1e-6，学习率逐轮减小，每 10 轮重新循环。此外，每循环 5 轮保存一次模型参数。与保存整个模型相比，仅保存模型参数的方式可以提高保存速度，加快训练过程。

10.3 多模态数据分类案例

以下是测试模型准确率的函数，该函数跟训练函数 train()的部分代码相似：

```
def getAccOnadataset(model:Multi_Model, data_loader): #测试模型的准确率
    model.eval()
    correct = 0
    with torch.no_grad():
        for i, data in enumerate(data_loader):
            input_ids, token_type_ids, attention_mask, img, label = data
            label = label.to(device)
            pre_y = multi_model(data)
            pre_y = torch.argmax(pre_y, dim=1)
            t = (pre_y == label).long().sum()
            correct += t
        correct = 1. * correct / len(data_loader.dataset)
    model.train()
    return correct.item()
```

10.3 多模态数据分类案例

（4）编写主函数代码。通过调用上述函数，读取文本数据和图像数据，并进行张量化和打包，然后创建网络类实例，构建网络模型，并利用文本和图像张量对模型进行训练，最后测试模型的准确率。

```
if __name__ == '__main__':
    batch_size = 8
    path = r'.\data\multimodal-cla'
    samples_train = get_txt_img_lb(path, 'ID-label-train.txt') # 读取训练集,3609
    samples_test = get_txt_img_lb(path, 'ID-label-test.txt') # 读取测试集,902
     # 实例化训练集和测试集
    train_dataset = MyDataSet(samples_train)
    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
    test_dataset = MyDataSet(samples_test)
    test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=True)
    multi_model = Multi_Model().to(device)
    train(multi_model, train_loader) #对模型进行训练
    print('测试中.....')
    check_point = torch.load(f'multi_model_new.pt') #加载已训练的模型参数
    multi_model.load_state_dict(check_point['model_state_dict'])
    acc_test = getAccOnadataset(multi_model, test_loader)
    print('在测试集上的准确率：{:.1f}%'.format(acc_test*100))
```

10.3 多模态数据分类案例

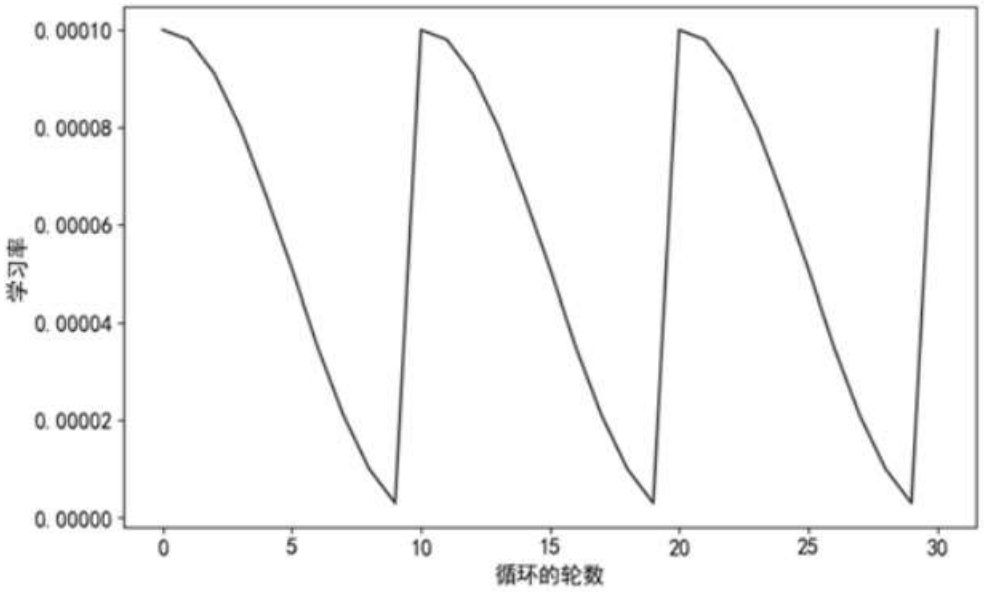
执行由上述代码构成的 Python 文件，输出结果如下：

```
训练中.....
训练完毕！
测试中.....
在测试集上的准确率：63.2%
```

10.3 多模态数据分类案例

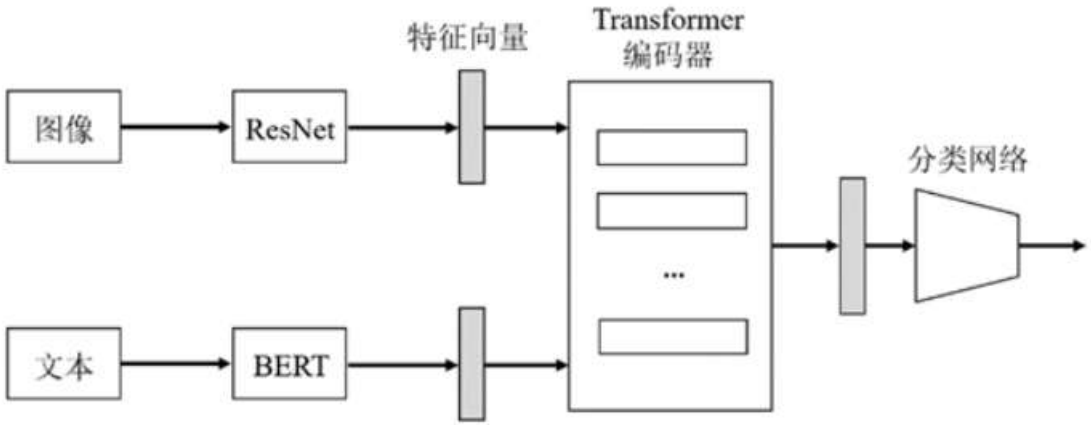
结果表示，该模型在测试集上的准确率为63.2%。该例子给我们展示了如何对多模态数据进行分类，由此不难总结处理多模态数据的一般过程和方法。

在训练过程中，我们还采用了**学习率衰减技术**，具体做法是：学习率从 1e-4 逐步衰减至 1e-6，以 10 轮循环为一个周期，不断重复，以尽可能找到精准解。本例学习率的变化曲线如右图所示，该图非常清晰地展示了本例学习率的变化趋势。



10.3 多模态数据分类案例

实际上，拼接融合和线性加权融合在复杂场景下显得有点“机械化”，它无法体现不同模态特征之间的互补性和相关性。我们将特征融合方式改为由 Transformer 编码器来实现，并 采用右图 10-6 所示的网络结构。结果发现，准确率可以提高到 75%左右。读者可以按照该 网络结构尝试用 Transformer 编码器来实现多模态特征的融合。



10.4 本章小结

本章内容：

- 多模态学习的概念和发展过程
- 多模态数据分类
- 基于 PyTorch 框架的多模态数据分类的实现方法