华东师范大学数据科学与工程学院实验报告

课程名称：当代人工智能	**年级：**2021 级	**上机实践成绩：**
指导教师：李翔	**姓名：**李睿恩	**学号：**10215501434
上机实践名称：多模态情感分析		**上机实践日期：**2024.1.27
上机实践编号：5	**组号：**无	**上机实践时间：**21:54

一、 实验目的

- 给定配对的文本和图像，预测对应的情感标签（positive, neutral, negative）

二、 实验任务

- 设计一个多模态融合模型
- 自行从训练集中划分验证集，调整超参数
- 预测测试集（test_without_label.txt）上的情感标签

三、 实验环境

本次实验基于 **AutoDL 算力云**平台。我们在 AutoDL 算力云平台中租用了 1 块 RTX 4090(24GB)的 GPU，并在其提供的 JupyterLab 平台中进行了本次实验。

我们所租用的实验环境基于 Ubuntu 22.04，采用 Python 3.10 作为编程语言，并使用了 PyTorch 2.1.0 框架进行深度学习任务，同时借助 CUDA 12.1 实现对 GPU 的加速。

本次实验被上传到了 github。github 地址：

https://github.com/HonokaKousaka/Contemporary-Artificial-Intelligence/tree/main/Project_5

四、 实验过程

4.1 多模态情感分析简介

情感分析是一个被长期研究的问题。根据论文 *A survey of multimodal sentiment analysis* (Ziyuan Zhao et al. 2019)，网络的出现让人们在社交媒体上大量地传播自己的观点与情感，这为情感分析提供了大量的材料。早期，人们主要关注基于纯文本的情感分析，并且这已经有了许多较好的解决方案。但是，现在的人们往往会多模态地表达情感，比如在 Youtube 上传视频，在 Instagram 或 X（曾经的 Twitter）中图文并茂地分享自己的情感，这导致了现在的情感分析往往需要多模态进行。其中的一个研究分支，就是在社交媒体上结合图片与文本分析一段推文的情感。

根据论文 *Multimodal Sentiment Analysis With Image-Text Interaction Network* (Tong Zhu et al. 2023)，我们注意到现在的社交媒体上存在一些现象，即对于一个既包含文本又包含图像的推文，如果我们只观看图像，我们可能会得到错误的情感判断。一个看似积极的图片所配的文本可能是悲观的，而一个看似严肃的图片所配的文本却可能是喜庆的，也有可能一张没有

任何情感倾向的风景照片实际上配上了发送者喜悦的文本，这使得只基于一个介质来分析情感是易错的。在社交媒体上，也广泛存在反讽的现象，如文字“今天天气真好！”所搭配的图片很可能是乌云密布的雨天，如果由人来判断其情感，可能会判断其情感为“Negative”，但一个由计算机训练的模型则可能判断其为“Positive”。这也体现了多模态情感分析的必要性。

4.2 实验数据集分析

我们点开几张图片进行观察，发现图片的内容非常丰富。有的图片是一张梗图，有的图片是一位明星，有的图片是手机截图，有的图片是记录生活的拍摄……这很难发现过多共同的特征。

我们利用代码来阅读每一个图片所对应的文本文件，可以发现其内容大致如图 1 所示。

```
RT @SamriddhaRay: @AshleyJudd @shailenewoodley @Divergent @AnselElgort Yes and I am a fan of the Divergent
series too! #excited #fanArt htt…

#February #Winter #Rainy #Stormy #Windy #Wednesday #Morning #Love #Happy #Positive #Passionate #Calm #M
#Coffee ?????

Kat just before she gets #beaten by Killer Sex in a #topless #submission #catfight.

#February #Winter #Rainy #Stormy #Windy #Wednesday #Evening #Love #Happy #Positive #Passionate #Calm #Fun
#Life ?? ?

Get your Friday Night Look Sorted!! #Newin #LBD #Littleblackdress #Strappy #Plunge #Neckline #Mini #Black
#Bodycon

RT @N2312Neagoe: #love #caring #beautiful

#February #Winter #Rainy #Stormy #Windy #Wednesday #Morning #Love #Happy #Positive #Passionate #Reading
#Coffee ?????

RT @ThatGuy__Kai: Honored to have Pittsburgh Pirates Consultant @CoachOTip speak to our ball club
#passionate #relentless #ownership http:/…
```

图 1 数据集文本概览

我们发现，这些文本往往会带有如“@”，“#”这样的在社交媒体上广泛出现的字符，并且这些文本大多格式随意，非常像在社交媒体上随意发出的推文。我们在有的文本中发现了完整的链接，在浏览器中打开时我们发现这些链接指向了社交媒体 X（曾经的 Twitter）。这意味着，这些图文数据大概率来自于社交媒体 X。

对问题的判断有助于我们寻找最适合该问题的论文进行阅读，也有助于我们寻找最适合该问题的模型。

4.3 模型选用

论文 *A survey of multimodal sentiment analysis* (Ziyuan Zhao et al. 2019)中提到，可能目前最有前景的文本情感分析运用了**深度学习**。这启发我们联系到上次实验中运用过的一些大语言模型（T5, BERT）可能对文本情感分析有帮助。论文 *Multimodal Sentiment Analysis With*

Image-Text Interaction Network (Tong Zhu et al. 2023)与论文 *Transfer Learning with Joint Fine-Tuning For Multimodal Sentiment Analysis* (Guilherme Lourenco de Toledo et al. 2022)中对于文本情感分析均提到了 BERT，这启发我们在此次实验中可以使用 BERT 作为文本情感分析的模型。

论文 *A survey of multimodal sentiment analysis* (Ziyuan Zhao et al. 2019)中提到，最近视觉上的情感分析也会考虑到**卷积神经网络（CNN）**的成功。这篇论文高度赞扬了 AlexNet 在图像分类上的成果，从而提到了一些论文使用预训练的 AlexNet 来进行视觉角度的情感分析。这启发我们使用 AlexNet 来进行视觉角度的情感分析。专门研究社交媒体多模态情感分析的论文 *An Image-Text Consistency Driven Multimodal Sentiment Analysis Approach for Social Media* (Ziyuan Zhao et al. 2019)提及了对 VGGNet 与 ResNet 的使用。论文 *Multimodal Sentiment Analysis With Image-Text Interaction Network* (Tong Zhu et al. 2023)则是直接使用了 ResNet 作为视觉情感分析的一环。因此，我们会在此次实验中使用 VGGNet，ResNet 与 AlexNet 来进行视觉角度的情感分析。多模态情感分析的示意图如图 2.

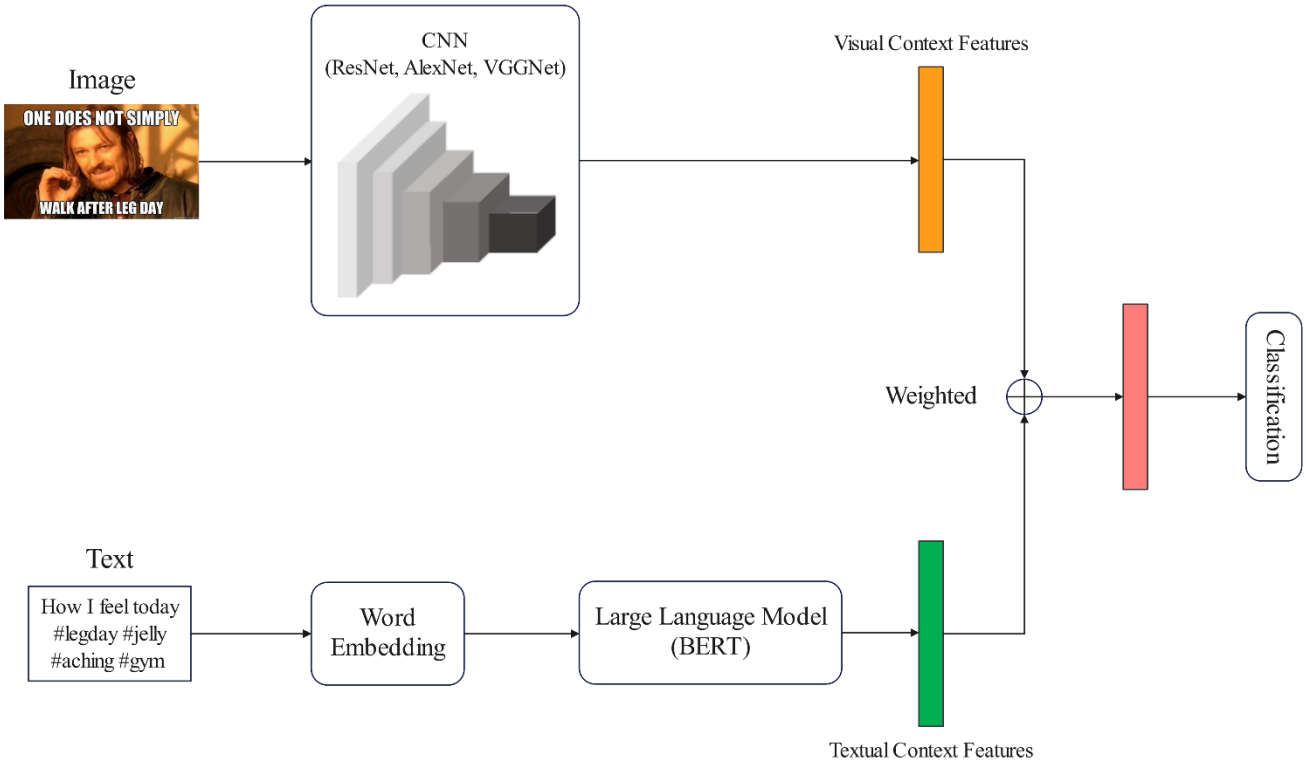

图 2 多模态情感分析示意图

4.4 功能实现

本次功能的实现参考了多个 github 仓库。它们分别是：

- https://github.com/abhimishra91/transformers-tutorials
- https://github.com/guitld/Transfer-Learning-with-Joint-Fine-Tuning-for-Multimodal-Sentiment-Analysis/tree/main
- https://github.com/Miaheeee/AI_lab5

由于此次实验需要自行编写深度学习相关的代码，这对我而言仍然是一个挑战，因此需要一些现存的 github 上的参考资料。其中一个 github 仓库即上次实验中我们参考的 github 仓库，由于其中涉及了大量有关深度学习的代码的编写方式，因此进行了一定的参考学习。

另外，此次实验的实现也参考了个别前辈们实现相同工作时的代码。因为此次实验涉及到语言模型与图像模型的交叉使用，我本人缺乏对相关领域的工程能力，因此在必要的时候参考了前辈们对该工作的实现。

我们先编写模型部分。我们本次实验主要尝试三种模型，一种是 BERT 结合 AlexNet，一种是 BERT 结合 ResNet34，一种是 BERT 结合 VGGNet19。在这里我们以 BERT 结合 ResNet 为例观察代码的编写。

我们调用 BERT 模型的方式来源于调用 HuggingFace 中已经预训练过的模型，调用 ResNet 的方式来源于调用 torchvision 中预训练过的各类模型。另外，我们结合 BERT 与卷积神经网络的方式是，将输入的图片与文本分别转化为一个有 128 个神经元的特征，将这两个特征进行加权求和，从而得到一个含有图片与文本特征的 128 维的特征，最后将该 128 维的特征再利用一次全连接神经网络化为 3 分类的 3 个值，谁的数值大，表明最有可能是哪一个情感分类。其代码结构如图 3 所示。

```python
def __init__(self):
    super().__init__()
    # 使用Bert与ResNet34的预训练模型
    self.textual_model = BertModel.from_pretrained('bert-base-uncased')
    self.visual_model = torchvision.models.resnet34(pretrained=True)
    # 事先定义神经网络层
    self.textual_linear = nn.Linear(768, 128)   # 文本的线性层
    self.visual_linear = nn.Linear(1000, 128)   # 图片的线性层
    self.image_weight = nn.Linear(128, 1)       # 为图片计算权重
    self.text_weight = nn.Linear(128, 1)        # 为文本计算权重
    self.result = nn.Linear(128, 3)             # 三分类全连接层
    self.relu = nn.ReLU()                       # 非线性激活层

HonokaKousaka *
def forward(self, input_ids, attention_mask, image):
    visual_output = self.visual_model(image)
    visual_output = self.visual_linear(visual_output)
    visual_output = self.relu(visual_output)
    image_weight = self.image_weight(visual_output)

    textual_output = self.textual_model(input_ids=input_ids, attention_mask=attention_mask)
    textual_output = textual_output.last_hidden_state[:, 0, :]
    textual_output.view(textual_output.shape[0], -1)
    textual_output = self.textual_linear(textual_output)
    textual_output = self.relu(textual_output)
    text_weight = self.text_weight(textual_output)

    final_output = image_weight * visual_output + text_weight * textual_output
    final_output = self.result(final_output)
    return final_output
```

图 3 BERT 与卷积神经网络的结合

在主函数 main.py 中，我们编写了用于 Word Embedding 的函数。我们使用来自 Hugging Face 的 BERT 的分词器（Tokenizer），设置 truncation=True，规定如果文本长度超过指定的最大长度（max_length）就要截断文本，设置 padding 的值为 max_length，对于长度不足 max_length 的文本末尾都会进行填充，另外我们要求它返回 PyTorch 张量。最后，函数返回编码后的输入 input_ids 和注意力掩码 attention_mask，这两个张量会作为之后的模型的输入。

具体实现如图 4.

```
def word_embed(txt, token):
    result = token.batch_encode_plus(batch_text_or_text_pairs=txt,
                                     truncation=True,
                                     padding='max_length',
                                     max_length=32,
                                     return_tensors='pt')
    input_ids = result['input_ids']
    attention_mask = result['attention_mask']
    return input_ids, attention_mask
```

图 4 Word Embedding 代码

我们还编写了 NewDataset 类，加载图像、文本描述和相应的情感标签。__init__方法会初始化数据集，__len__会返回数据集的大小，__getitem__会获取数据集中指定索引处的图像、文本、情感标签、输入 input_ids 和注意力掩码。具体实现如图 5.

```
class NewDataset:
    HonokaKousaka
    def __init__(self, images, descriptions, tags, token):
        self.images = images
        self.descriptions = descriptions
        self.tags = tags
        self.input_ids, self.attention_masks = word_embed(self.descriptions, token)

    HonokaKousaka
    def __len__(self):
        return len(self.descriptions)

    HonokaKousaka
    def __getitem__(self, idx):
        img = self.images[idx]
        des = self.descriptions[idx]
        tag = self.tags[idx]
        input_id = self.input_ids[idx]
        attention_mask = self.attention_masks[idx]
        return img, des, tag, input_id, attention_mask
```

图 5 NewDataset 代码

我们另开了 train.py，这是一个具体描绘了模型训练过程的函数，会在主函数运行时被调用。除了调用交叉熵损失函数，使用 Adam 优化器与进行反向传播以外，在训练的过程中，它会持续记录训练集的损失函数，并在每一轮训练结束时显示目前的损失函数值，训练集准确率与验证集准确率。其工作原理与第四次作业非常相似，部分代码呈现如图 6.

```python
for epoch in range(epoch_num):
    loss = 0.0
    # 训练集和验证集正确判读的个数
    train_cor_count = 0
    valid_cor_count = 0
    for _, (img, des, target, idx, mask) in enumerate(train_dataloader):
        img, mask, idx, target = img.to(device), mask.to(device), idx.to(device), target.to(device)
        output = model(idx, mask, img)
        optimizer.zero_grad()
        loss = loss_c(output, target)
        # 反向传播
        loss.backward()
        optimizer.step()
        pred = output.argmax(dim=1)
        train_cor_count += int(pred.eq(target).sum())
        if _ % 10 == 0:
            loss_array.append(loss.item())
        if _ % 500 == 0:
            print('Epoch: {}, Train Loss: {:.4f}'.format( *args: epoch + 1, loss.item()))
    # 训练集准确率
    train_acc.append(train_cor_count / train_count)
    for img, des, target, idx, mask in valid_dataloader:
        img, mask, idx, target = img.to(device), mask.to(device), idx.to(device), target.to(device)
        output = model(idx, mask, img)
        pred = output.argmax(dim=1)
        valid_cor_count += int(pred.eq(target).sum())
    # 验证集准确率
    valid_acc.append(valid_cor_count / valid_count)
```

图 6 train.py 核心代码

在主函数中，我们对文本的处理较为特别。考虑到本次数据所提供的文本大概率来自社交媒体 X，其文本中大多含有“@”（后跟用户名），“#”（后跟话题标签）以及网址（指向 X），因此我们需要对文本进行预处理。我们认为，用户名与网址对情感分析没有任何帮助，但话题标签可能对情感分析有帮助，因此我们对所有的话题标签仅仅是去除“#”，而对于“@”后的文本与所有网址（含有 http 的字符串）我们则采取删除的操作。最后，由于社交媒体的推文往往语言结构上不严谨，再加上我们会考虑话题词汇，因此我们最后得到的大概率是不成文的离散词汇表。为了能够对离散词汇表进行比较合理的情感分析，我们使用了在第一次实验中就使用过的“**停用词**”的概念，对一些非常常用的，不能提供情感倾向分析的词汇进行了剔除。最后的文本预处理代码如图 7.

```python
# 收录词语
for i in range(len(descriptions)):
    des = descriptions[i]
    # 空格分词，且去除Hashtag
    word_list = des.replace( _old: '#', _new: '').split(' ')
    words_result = []
    for word in word_list:
        # 是一个词，且不是网站，也不是用户名
        if len(word) >= 1 and ('http' not in word) and (word[0] != '@') and word.lower() not in stop_words:
            words_result.append(word)
    descriptions[i] = ' '.join(words_result)
```

图 7 文本预处理代码

我们对图像也会进行一定的预处理，因为社交媒体上的推文给出的图像往往规模不同一，这难以放入同一个深度学习的模型中。因此，我们对图像的大小也进行了调整，将图片

的规模均调整为 224*224，采取 LANCZOS 插值方法。具体代码如图 8 所示。

```
# 读取数据
train_dataframe = pd.read_csv("./train.txt")
for i in range(train_dataframe.shape[0]):
    guid = train_dataframe.iloc[i]['guid']
    tag = train_dataframe.iloc[i]['tag']
    img_path = f'./data/{guid}.jpg'
    img = Image.open(img_path).resize( size: (224, 224), Image.Resampling.LANCZOS)
    img = np.asarray(img, dtype='float32')
```

图 8 图像预处理代码

4.5 实验结果

通过在 AutoDL 中运行代码，我们可以得到三个不同模型的如图 9 的准确度图像。

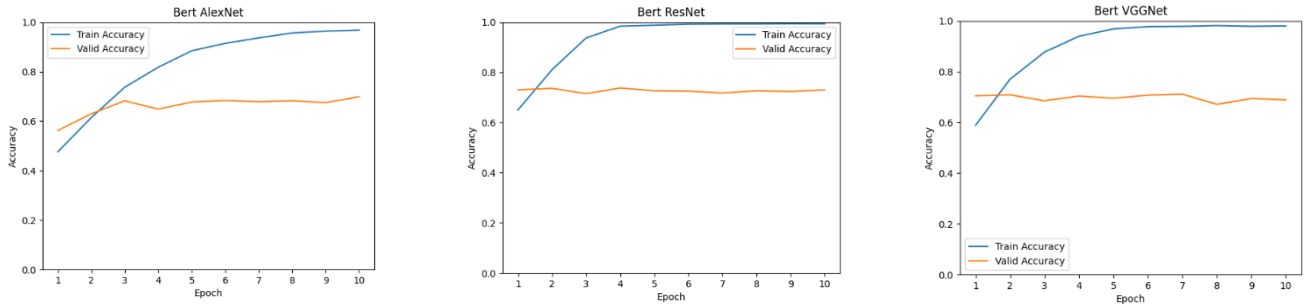

图 9 三个模型的准确度图像

通过代码的运行结果，我们可以得到如表 1 的验证集准确率表格。

表 1 三个模型的验证集准确率表

模型	验证集准确率
BERT + AlexNet	69.87%
BERT + ResNet34	73.00%
BERT + VGGNet19	68.87%

这说明在我们设计的模型中，BERT 与 ResNet34 结合的模型具有较好的效果，由其生成的测试集的预测会作为最终提交的预测文件。

4.6 消融实验

考虑到这样的效果可能不来自于多模态融合模型，而可能主要来源于某一个特定介质的模型，我们计划进行消融实验，即只输入文本或只输入图像可以得到怎么样的预测表现。

消融实验的模型的编写也比较简单，只需要删去另一个不被使用的模型的部分，删除加权的部分即可。由于在刚才的实验中我们发现 BERT 与 ResNet34 的结合效果最好，因此只考虑文本的模型我们仍然采用 BERT，只考虑图像的模型我们采用 ResNet34.

图 10 展现了一个只输入图像的模型。

```
class imageModel(nn.Module):
    HonokaKousaka
    def __init__(self):
        super().__init__()
        self.img_model = torchvision.models.resnet34(pretrained=True)
        self.i_linear = nn.Linear(1000, 256)
        self.fc = nn.Linear(256, 3)
        self.relu = nn.ReLU()

    HonokaKousaka
    def forward(self, input_ids, attention_mask, image):
        img_out = self.img_model(image)
        img_out = self.i_linear(img_out)
        img_out = self.relu(img_out)
        last_out = self.fc(img_out)
        return last_out
```

图 10 只输入图像的模型

最终我们可以得到如表 2 的消融实验结果。

表 2 消融实验结果

模型	验证集准确率
BERT + ResNet34（供参考）	73.00%
ResNet34	68.87%
BERT	67.87%

这个表格充分说明了，只使用文本来进行情感分析或只使用图像来进行情感分析，准确率的确是低于多模态情感分析的。实际上这个结果也非常自然，因为社交媒体上的文字往往结构性差，其文本往往非常零散，单独观看文本难以得到正确的情感分类，而社交媒体上的图像则种类更加丰富，直接判断情感也非常困难。这进一步说明了多模态情感分析的重要性。

4.7 难点与解决

本次实验过程中的确遇到了一些难点。

由于我们基于 AutoDL 的平台来实现深度学习的训练，因此我们需要将数据导入至平台上。但是，数千张图片与文本体量过大，他们的同时上传对于线上的 JupyterLab 是一个挑战。经过研究后，我通过将所有图片与文本存入压缩包（.zip 文件），之后将压缩包上传至平台，然后利用 Linux 终端的指令（unzip data.zip）解压了压缩包，实现了文件的传输。这也

启发了我未来如果需要借助其他线上平台实现大文件的操作，可以利用压缩包的解压来实现。

在代码编写过程中，我们遇到了如图 11 的报错。

```
Traceback (most recent call last):
  File "/root/main.py", line 169, in <module>
    main()
  File "/root/main.py", line 82, in main
    model = bertResModel.to(device)
  File "/root/miniconda3/lib/python3.10/site-packages/torch/nn/modules/module.py", line 1160, in to
    return self._apply(convert)
AttributeError: 'torch.device' object has no attribute '_apply'
```

图 11 代码编写过程中的报错

在之前复现代码时，我对.to(device)了解过少，因此导致了该错误的发生。to 方法用于将模型或张量移动到指定的设备上，但该方法要传递的应当是一个对象，而在代码编写的过程中，我传递的是 bertResModel 而不是 bertResModel()，因此导致了错误。这也启发我在深度学习的编写中关注到细节。

我在代码中引用了停用词的概念，但这需要我们进行下载。然而，我在 AutoDL 上执行时却发现下载会成为代码执行的一个瓶颈，它可能会卡在这一步而不进行后续的代码操作。这说明，网络会成为一些内容分发网络使用情况下的瓶颈。我通过阅读 AutoDL 的帮助文档，找到了在终端与.ipynb 文件中加速的方式，从而解决了该问题。这启发我在运行代码时注意到网络环境，也启发我去阅读不同平台的帮助文档来解决问题。

五、 总结与分析

多模态情感分析（Multimodal Sentiment Analysis）是目前一个非常重要的研究方向，目前主要的研究方向之一就是基于社交媒体上的图片与文字来进行情感分析。文本情感分析目前往往借助深度学习或大语言模型的协助，而图片情感分析则会基于成熟的卷积神经网络技术。

在进行多个模型混合的实验时，我们可以进行消融实验，即让其中一个模型的作用消失，从而观测该模型对整个混合模型有怎么样的效果。

深度学习模型的编写非常有难度，它有别于传统的代码结构，需要对 PyTorch 架构或 TensorFlow 架构有深刻的认识。这还需要我的进一步学习与理解。

六、 参考文献与技术博客

[1] Zhu, T., Li, L., Yang, J., Zhao, S., Liu, H., & Qian, J. (2022). Multimodal sentiment analysis with image-text interaction network. *IEEE Transactions on Multimedia*.

[2] Zhao, Z., Zhu, H., Xue, Z., Liu, Z., Tian, J., Chua, M. C. H., & Liu, M. (2019). An image-text consistency driven multimodal sentiment analysis approach for social media. *Information Processing & Management, 56*(6), 102097.

[3] Soleymani, M., Garcia, D., Jou, B., Schuller, B., Chang, S. F., & Pantic, M. (2017). A survey of

multimodal sentiment analysis. *Image and Vision Computing*, *65*, 3-14.

[4] Yang, X., Feng, S., Wang, D., & Zhang, Y. (2020). Image-text multimodal emotion classification via multi-view attentional network. *IEEE Transactions on Multimedia*, *23*, 4014-4026.

[5] de Toledo, G. L., & Marcacini, R. M. (2022). Transfer Learning with Joint Fine-Tuning for Multimodal Sentiment Analysis. *arXiv preprint arXiv:2210.05790*.

[6] https://github.com/abhimishra91/transformers-tutorials

[7] https://github.com/guitld/Transfer-Learning-with-Joint-Fine-Tuning-for-Multimodal-Sentiment-Analysis/tree/main

[8] https://github.com/Miaheeee/AI_lab5