

代码复现（包括复现过程、所用包、关键网络结构即对应代码、关键公式）

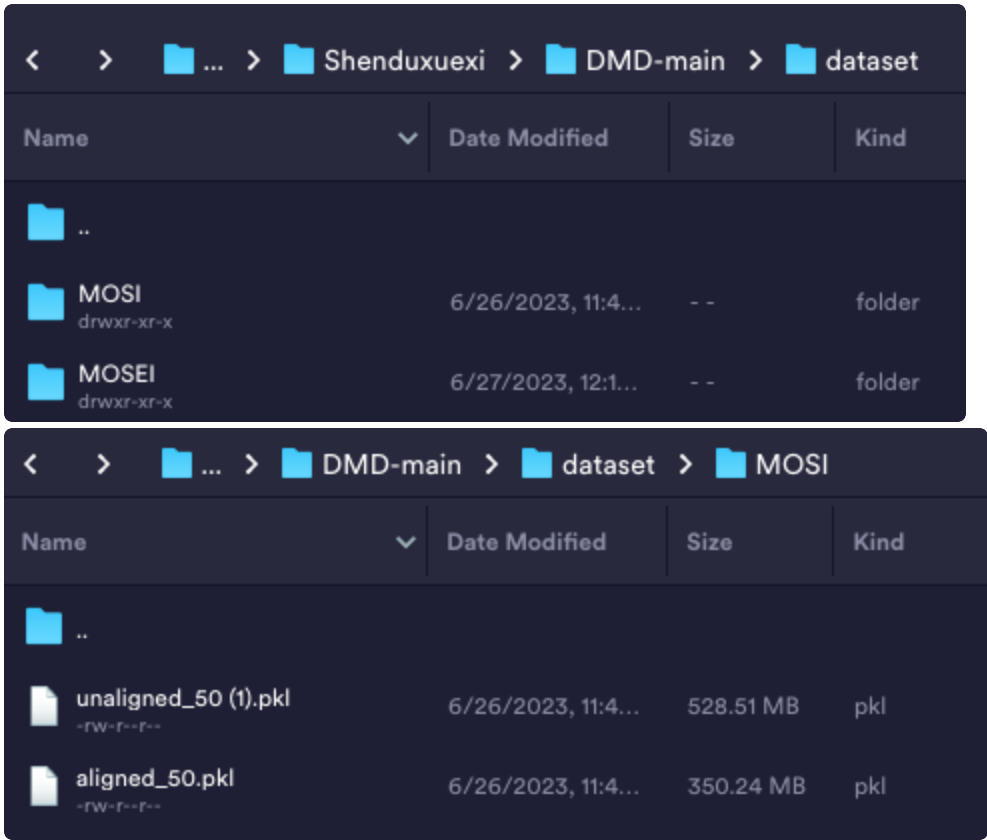
2、代码复现过程

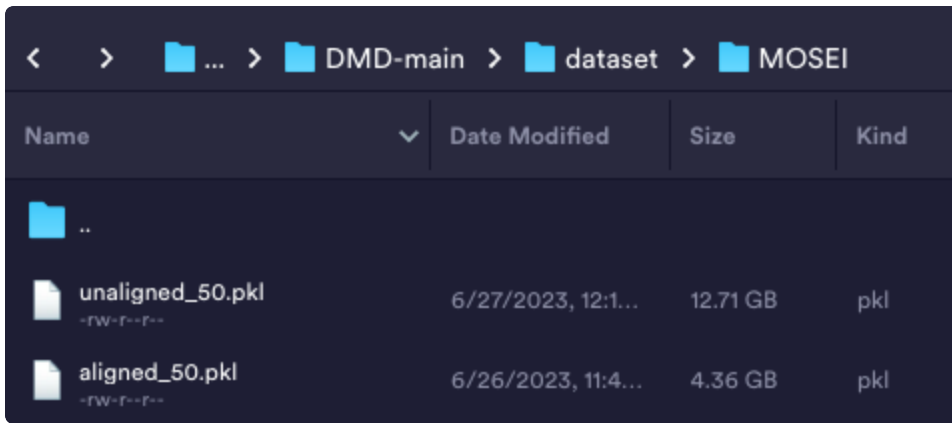
2.1 下载数据集

数据文件（包含经过处理的MOSI、MOSEI数据集）

从https://drive.google.com/drive/folders/1BBadVSptOe4h8TWchkhWZRLJw8YG_aEi?usp=sharing下载。

将下载的数据集放入 `./dataset` 目录中。





Name	Date Modified	Size	Kind
..			
unaligned_50.pkl	6/27/2023, 12:1...	12.71 GB	pkl
aligned_50.pkl	6/26/2023, 11:4...	4.36 GB	pkl

2.2 在./config/config.json文件中修改必要的参数。

修改后的json文件：

对相关路径 `"featurePath"` 进行了修改

```
{
  "datasetCommonParams": {
    "dataset_root_dir": "./dataset",
    "mosi": {
      "aligned": {
        "featurePath": "MOSI/aligned_50.pkl",
        "feature_dims": [768, 5, 20],
        "train_samples": 1284,
        "num_classes": 3,
        "language": "en",
        "KeyEval": "Loss"
      },
      "unaligned": {
        "featurePath": "MOSI/unaligned_50.pkl",
        "feature_dims": [768, 5, 20],
        "train_samples": 1284,
        "num_classes": 3,
        "language": "en",
        "KeyEval": "Loss"
      }
    },
    "mosei": {
      "aligned": {
        "featurePath": "MOSEI/aligned_50.pkl",
        "feature_dims": [768, 74, 35],
        "train_samples": 16326,
        "num_classes": 3,
        "language": "en",
        "KeyEval": "Loss"
      },
      "unaligned": {
        "featurePath": "MOSEI/unaligned_50.pkl",
        "feature_dims": [768, 74, 35],
        "train_samples": 16326,
        "num_classes": 3,
        "language": "en",
        "KeyEval": "Loss"
      }
    }
  },
  "dmd": {
    "commonParams": {
      "need_data_aligned": true,
      "need_model_aligned": true,
      "early_stop": 10,
    }
  }
}
```

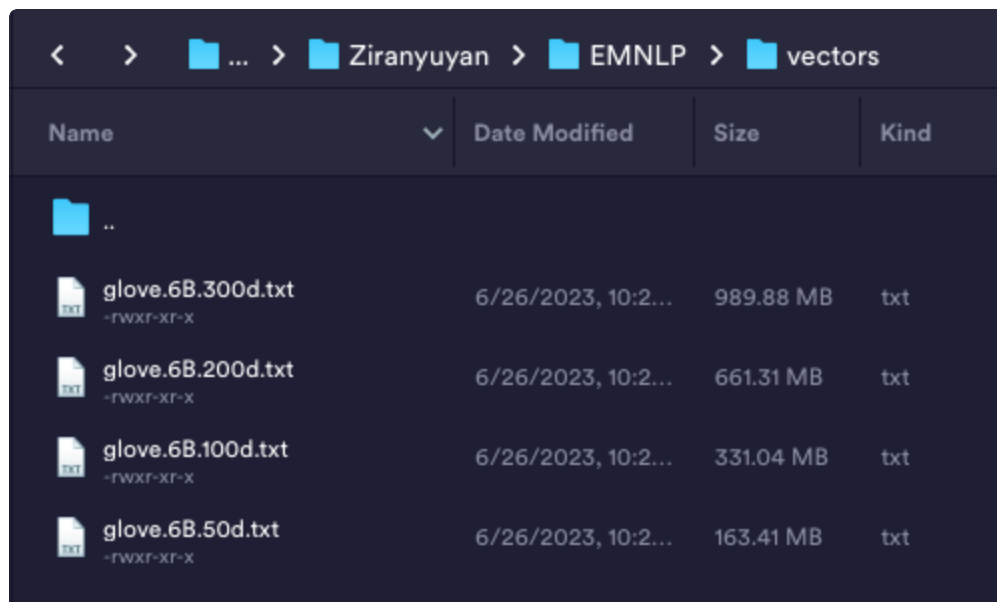
```

    "use_bert": true,
    "use_finetune": true,
    "attn_mask": true,
    "update_epochs": 10
},
"datasetParams": {
    "mosi": {
        "attn_dropout_a": 0.2,
        "attn_dropout_v": 0.0,
        "relu_dropout": 0.0,
        "embed_dropout": 0.2,
        "res_dropout": 0.0,
        "dst_feature_dim_nheads": [50, 10],
        "batch_size": 16,
        "learning_rate": 0.0001,
        "nlevels": 4,
        "conv1d_kernel_size_l": 5,
        "conv1d_kernel_size_a": 5,
        "conv1d_kernel_size_v": 5,
        "text_dropout": 0.5,
        "attn_dropout": 0.3,
        "output_dropout": 0.5,
        "grad_clip": 0.6,
        "patience": 5,
        "weight_decay": 0.005,
        "transformers": "bert",
        "pretrained": "bert-base-uncased"
    },
    "mosei": {
        "attn_dropout_a": 0.0,
        "attn_dropout_v": 0.0,
        "relu_dropout": 0.0,
        "embed_dropout": 0.0,
        "res_dropout": 0.0,
        "dst_feature_dim_nheads": [30, 6],
        "batch_size": 16,
        "learning_rate": 0.0001,
        "nlevels": 4,
        "conv1d_kernel_size_l": 5,
        "conv1d_kernel_size_a": 1,
        "conv1d_kernel_size_v": 3,
        "text_dropout": 0.3,
        "attn_dropout": 0.4,
        "output_dropout": 0.5,
        "grad_clip": 0.6,
        "patience": 5,
        "weight_decay": 0.001,
        "transformers": "bert",

```

```
        "pretrained": "bert-base-uncased"
    }
}
}
```

2、下载预训练模型 **Pretrained GloVe Embeddings**，嵌入并保存在 `/vectors` 中。



2.3 训练

Python | 复制代码

```
python train.py
```

复现时在训练的38个epoch得到收敛的DMD模型。

epoch1-9训练结果截图：

```

2023-06-27 00:30:00,223 - MMSA [INFO] - train samples: (1284,)
2023-06-27 00:30:00,518 - MMSA [INFO] - valid samples: (229,)
2023-06-27 00:30:00,792 - MMSA [INFO] - test samples: (686,)
2023-06-27 00:30:56,795 - MMSA [INFO] - >> Epoch: 1 TRAIN-(dmd) [1/1/1] >> total_loss:
10.6731 Acc_2: 0.5248 F1_score: 0.4299 Acc_7: 0.1924 MAE: 1.3290
2023-06-27 00:30:58,679 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.5741 F1_score: 0.4187
Acc_7: 0.2140 MAE: 1.4164 Loss: 1.3909
2023-06-27 00:31:02,779 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.4223 F1_score: 0.2507
Acc_7: 0.1545 MAE: 1.4817 Loss: 1.4827
2023-06-27 00:31:30,063 - MMSA [INFO] - >> Epoch: 2 TRAIN-(dmd) [1/2/1] >> total_loss:
8.883 Acc_2: 0.5581 F1_score: 0.4066 Acc_7: 0.1955 MAE: 1.2803
2023-06-27 00:31:32,020 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.5741 F1_score: 0.4187
Acc_7: 0.2227 MAE: 1.3075 Loss: 1.2868
2023-06-27 00:31:35,985 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.4238 F1_score: 0.2540
Acc_7: 0.1676 MAE: 1.3793 Loss: 1.3798
2023-06-27 00:32:04,120 - MMSA [INFO] - >> Epoch: 3 TRAIN-(dmd) [1/3/1] >> total_loss:
7.8938 Acc_2: 0.6190 F1_score: 0.5349 Acc_7: 0.2492 MAE: 1.1549
2023-06-27 00:32:05,800 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8056 F1_score: 0.8003
Acc_7: 0.2882 MAE: 1.1766 Loss: 1.1599
2023-06-27 00:32:09,943 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.7698 F1_score: 0.7692
Acc_7: 0.2274 MAE: 1.2357 Loss: 1.2363
2023-06-27 00:32:36,812 - MMSA [INFO] - >> Epoch: 4 TRAIN-(dmd) [1/4/1] >> total_loss:
7.0991 Acc_2: 0.8318 F1_score: 0.8281 Acc_7: 0.2765 MAE: 0.9785
2023-06-27 00:32:38,609 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8611 F1_score: 0.8598
Acc_7: 0.3013 MAE: 1.0069 Loss: 1.0273
2023-06-27 00:32:42,377 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8201 F1_score: 0.8212
Acc_7: 0.2507 MAE: 1.0609 Loss: 1.0604
2023-06-27 00:33:09,484 - MMSA [INFO] - >> Epoch: 5 TRAIN-(dmd) [1/5/1] >> total_loss:
6.4069 Acc_2: 0.9155 F1_score: 0.9153 Acc_7: 0.3388 MAE: 0.7675
2023-06-27 00:33:11,119 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8611 F1_score: 0.8607
Acc_7: 0.3493 MAE: 0.8804 Loss: 0.8852
2023-06-27 00:33:15,501 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8247 F1_score: 0.8255
Acc_7: 0.3630 MAE: 0.9152 Loss: 0.9145
2023-06-27 00:33:42,080 - MMSA [INFO] - >> Epoch: 6 TRAIN-(dmd) [1/6/1] >> total_loss:
5.7699 Acc_2: 0.9310 F1_score: 0.9309 Acc_7: 0.5210 MAE: 0.5333
2023-06-27 00:33:43,951 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8750 F1_score: 0.8747
Acc_7: 0.3974 MAE: 0.7811 Loss: 0.7917
2023-06-27 00:33:47,733 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8293 F1_score: 0.8295
Acc_7: 0.4519 MAE: 0.7529 Loss: 0.7540
2023-06-27 00:34:14,512 - MMSA [INFO] - >> Epoch: 7 TRAIN-(dmd) [1/7/1] >> total_loss:
5.3561 Acc_2: 0.9496 F1_score: 0.9497 Acc_7: 0.6184 MAE: 0.4079
2023-06-27 00:34:16,235 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8241 F1_score: 0.8250
Acc_7: 0.3843 MAE: 0.7894 Loss: 0.7845
2023-06-27 00:34:20,012 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8369 F1_score: 0.8346
Acc_7: 0.4373 MAE: 0.7389 Loss: 0.7386
2023-06-27 00:34:46,658 - MMSA [INFO] - >> Epoch: 8 TRAIN-(dmd) [1/8/1] >> total_loss:
5.125 Acc_2: 0.9553 F1_score: 0.9553 Acc_7: 0.6519 MAE: 0.3649
2023-06-27 00:34:48,555 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8611 F1_score: 0.8611
Acc_7: 0.3974 MAE: 0.7500 Loss: 0.7507
2023-06-27 00:34:52,489 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8277 F1_score: 0.8284
Acc_7: 0.4257 MAE: 0.7720 Loss: 0.7715
2023-06-27 00:35:19,360 - MMSA [INFO] - >> Epoch: 9 TRAIN-(dmd) [1/9/1] >> total_loss:
4.8758 Acc_2: 0.9602 F1_score: 0.9602 Acc_7: 0.6967 MAE: 0.3284

```

epoch35-38训练结果截图：

```

2023-06-27 00:48:41,409 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8354 F1_score: 0.8354
Acc_7: 0.4694 MAE: 0.7257 Loss: 0.7263
2023-06-27 00:49:06,648 - MMSA [INFO] - >> Epoch: 35 TRAIN-(dmd) [7/35/1] >> total_loss:
3.2327 Acc_2: 0.9959 F1_score: 0.9959 Acc_7: 0.8855 MAE: 0.1195
2023-06-27 00:49:08,452 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8657 F1_score: 0.8656
Acc_7: 0.4192 MAE: 0.6956 Loss: 0.6976
2023-06-27 00:49:12,329 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8323 F1_score: 0.8321
Acc_7: 0.4694 MAE: 0.7224 Loss: 0.7225
2023-06-27 00:49:38,458 - MMSA [INFO] - >> Epoch: 36 TRAIN-(dmd) [8/36/1] >> total_loss:
3.216 Acc_2: 0.9935 F1_score: 0.9935 Acc_7: 0.8902 MAE: 0.1160
2023-06-27 00:49:40,300 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8565 F1_score: 0.8566
Acc_7: 0.4236 MAE: 0.6933 Loss: 0.6762
2023-06-27 00:49:44,352 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8293 F1_score: 0.8291
Acc_7: 0.4636 MAE: 0.7276 Loss: 0.7277
2023-06-27 00:50:09,458 - MMSA [INFO] - >> Epoch: 37 TRAIN-(dmd) [9/37/1] >> total_loss:
3.1926 Acc_2: 0.9968 F1_score: 0.9968 Acc_7: 0.8808 MAE: 0.1183
2023-06-27 00:50:11,196 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8565 F1_score: 0.8566
Acc_7: 0.4279 MAE: 0.6907 Loss: 0.6884
2023-06-27 00:50:15,116 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8323 F1_score: 0.8321
Acc_7: 0.4723 MAE: 0.7255 Loss: 0.7262
2023-06-27 00:50:40,466 - MMSA [INFO] - >> Epoch: 38 TRAIN-(dmd) [10/38/1] >> total_loss:
3.1904 Acc_2: 0.9968 F1_score: 0.9968 Acc_7: 0.8933 MAE: 0.1134
2023-06-27 00:50:42,392 - MMSA [INFO] - VAL-(dmd) >> Acc_2: 0.8750 F1_score: 0.8747
Acc_7: 0.4279 MAE: 0.6901 Loss: 0.6804
2023-06-27 00:50:46,353 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8232 F1_score: 0.8233
Acc_7: 0.4665 MAE: 0.7278 Loss: 0.7283
2023-06-27 00:50:52,046 - MMSA [INFO] - TEST-(dmd) >> Acc_2: 0.8384 F1_score: 0.8383
Acc_7: 0.4665 MAE: 0.7206 Loss: 0.7214
2023-06-27 00:50:53,200 - MMSA [INFO] - Results saved to result/normal/mosi.csv.
2023-06-27 00:50:53,212 - MMSA [INFO] - train samples: (1284)

```

训练后的模型将保存在 `./pt` 目录下:

< > ... > Shenduxuexi > DMD-main > pt

Name	Date Modified	Size	Kind
38.pth	6/27/2023, 12:50 AM	436.07 MB	pth
37.pth	6/27/2023, 12:50 AM	436.07 MB	pth
36.pth	6/27/2023, 12:49 AM	436.07 MB	pth
35.pth	6/27/2023, 12:49 AM	436.07 MB	pth
34.pth	6/27/2023, 12:48 AM	436.07 MB	pth
33.pth	6/27/2023, 12:48 AM	436.07 MB	pth
32.pth	6/27/2023, 12:47 AM	436.07 MB	pth
31.pth	6/27/2023, 12:47 AM	436.07 MB	pth
30.pth	6/27/2023, 12:46 AM	436.07 MB	pth
29.pth	6/27/2023, 12:46 AM	436.07 MB	pth
28.pth	6/27/2023, 12:45 AM	436.07 MB	pth
27.pth	6/27/2023, 12:45 AM	436.07 MB	pth
26.pth	6/27/2023, 12:44 AM	436.07 MB	pth
25.pth	6/27/2023, 12:43 AM	436.07 MB	pth
24.pth	6/27/2023, 12:43 AM	436.07 MB	pth
23.pth	6/27/2023, 12:42 AM	436.07 MB	pth

结果保存在result/normal/mosi.csv文件中:

	A	B	C	D	E	F
1	Model	Acc_2	F1_score	Acc_7	MAE	Loss
2	dmd	(83.84, 0.0)	(83.83, 0.0)	(46.65, 0.0)	(72.06, 0.0)	(72.14, 0.0)

2.4 测试


```
python test.py
```

在run.py文件的第174行（下图第二行）设置已训练模型的路径:

```
if args.mode == 'test':
    model.load_state_dict(torch.load('pt/38.pth'))
    results = trainer.do_test(model, dataloader['test'], mode="TEST")
    sys.stdout.flush()
    input('[Press Any Key to start another run]')
```

```
(mmdiffusion) liyunqi@wangyutian-SYS-4028GR-TR2-1-EC028:~/Shenduxuexi/DMD-main$ python test.py
/home/liyunqi/miniconda3/envs/mmdiffusion/lib/python3.8/site-packages/scipy/__init__.py:143: UserWarning: A NumPy
version >=1.19.5 and <1.27.0 is required for this version of SciPy (detected version 1.19.2)
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")
MMSA - train samples: (1284,)
MMSA - valid samples: (229,)
MMSA - test samples: (686,)
testing phase for DMD
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertModel: ['cls.predictions.decoder.weight', 'cls.predictions.transform.dense.bias', 'cls.seq_relationship.weight', 'cls.predictions.transform.dense.weight', 'cls.predictions.transform.LayerNorm.weight', 'cls.predictions.transform.LayerNorm.bias', 'cls.predictions.bias', 'cls.seq_relationship.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
100%|██████████████████████████████████████████████████████████████████████████████| 43/43 [00:31<00:00, 1.37it/s]
MMSA - TEST-(dmd)>>> Acc_2: 0.8247 F1_score: 0.8247 Acc_7: 0.4708 MAE: 0.7276 Loss: 0.7279
```

测试结果: Acc_2: 0.8247 F1_score: 0.8247 Acc_7: 0.4708 MAE: 0.7276 Loss: 0.7279

2.5 复现过程使用到的包

以下是使用到的主要Python包名称：

1. json
2. os
3. pathlib
4. EasyDict (easydict)
5. gc
6. logging
7. time
8. numpy (np)
9. pandas (pd)
10. torch
11. torch.utils.data.DataLoader
12. torch.utils.data.Dataset

3、模型架构（重要公式和代码）

DMD框架如图2所示，主要由三个部分组成：多模态特征解耦（multimodal feature decoupling）、同质GD（homogeneous GD，简称HomoGD）和异质GD（heterogeneous GD，简称HeteroGD）。

考虑到不同模态之间显著的分布不匹配，我们通过学习共享的和独占的多模态编码器，将多模态表示解耦为同质和异质的多模态特征。

解耦的详细过程在第3.1节中介绍。为了实现灵活的知识传递，我们接下来从同质/异质特征中提取知识，这在两个图蒸馏单元（GD-Unit）中进行，即HomoGD和HeteroGD。

在HomoGD中，同质的多模态特征互相蒸馏以弥补彼此的表示能力。

在HeteroGD中，引入了多模态Transformer来显式建立跨模态的相关性和语义对齐，以进行进一步的蒸馏。

GD的详细内容在第3.2节中介绍。

最后，通过蒸馏得到的精炼多模态特征进行自适应融合，以实现稳健的情感识别。接下来，我们将详细介绍DMD的这三个部分。

3.1 多模态特征解耦

3.1.1 获取低级多模态特征

我们考虑了三种模态，即语言（L）、视觉（V）和声音（A）。

首先，我们利用三个单独的一维时间卷积层来聚合时间信息并获取低级多模态特征：

Python 复制代码

```
# 聚合时间信息并获取低级多模态特征
proj_x_l = x_l if self.orig_d_l == self.d_l else self.proj_l(x_l)
proj_x_a = x_a if self.orig_d_a == self.d_a else self.proj_a(x_a)
proj_x_v = x_v if self.orig_d_v == self.d_v else self.proj_v(x_v)
```

通过这种浅层编码，每个模态保留输入的时间维度，以便同时处理不对齐和对齐的情况。此外，为了方便后续的特征解耦，所有模态都被缩放到相同的特征维度。

3.1.2 各模态的异质特征和同质特征

为了将多模态特征解耦为同质特征 $\mathbf{X}_m^{\text{com}}$ 和异质特征 $\mathbf{X}_m^{\text{prt}}$ ，我们利用一个共享的多模态编码器 \mathcal{E}^{com} 和三个私有编码器 $\mathcal{E}_m^{\text{prt}}$ 来明确预测解耦后的特征。

为了区分 $\mathbf{X}_m^{\text{com}}$ 和 $\mathbf{X}_m^{\text{prt}}$ 之间的差异并减少特征的模糊性，我们以自回归的方式合成了vanilla耦合特征 $\tilde{\mathbf{X}}_m$ 。

$$\mathbf{X}_m^{\text{com}} = \mathcal{E}^{\text{com}}(\tilde{\mathbf{X}}_m), \mathbf{X}_m^{\text{prt}} = \mathcal{E}_m^{\text{prt}}(\tilde{\mathbf{X}}_m)$$

Python

复制代码

```
# 三个异质部分编码器
s_l = self.encoder_s_l(proj_x_l)
s_v = self.encoder_s_v(proj_x_v)
s_a = self.encoder_s_a(proj_x_a)
# 一个同质部分编码器
c_l = self.encoder_c(proj_x_l)
c_v = self.encoder_c(proj_x_v)
c_a = self.encoder_c(proj_x_a)
c_list = [c_l, c_v, c_a]
```

3.1.3 耦合特征

然后我们对每个模态的 $\mathbf{X}_m^{\text{com}}$ 和 $\mathbf{X}_m^{\text{prt}}$ 进行连接，并利用私有解码器 \mathcal{D}_m 生成耦合特征，即 $\mathcal{D}_m([\mathbf{X}_m^{\text{com}}, \mathbf{X}_m^{\text{prt}}])$ 。

随后，通过私有编码器 $\mathcal{E}_m^{\text{prt}}$ 重新对耦合特征进行编码，以回归异质特征。

Python

复制代码

```
# 解码器用于重建三种模态
recon_l = self.decoder_l(torch.cat([s_l, c_list[0]], dim=1))
recon_v = self.decoder_v(torch.cat([s_v, c_list[1]], dim=1))
recon_a = self.decoder_a(torch.cat([s_a, c_list[2]], dim=1))
# 重新对耦合特征进行编码
s_l_r = self.encoder_s_l(recon_l)
s_v_r = self.encoder_s_v(recon_v)
s_a_r = self.encoder_s_a(recon_a)
```

形式上，vanilla/合成的耦合多模式特征之间的差异可以公式化为：

$$\mathcal{L}_{\text{rec}} = \left\| \tilde{\mathbf{X}}_m - \mathcal{D}_m([\mathbf{X}_m^{\text{com}}, \mathbf{X}_m^{\text{prt}}]) \right\|_F^2 \quad (2)$$

进一步，vanilla/合成异构特征之间的差异可以公式化为

$$\mathcal{L}_{\text{cyc}} = \left\| \mathbf{X}_m^{\text{prt}} - \mathcal{E}_m^{\text{prt}}(\mathcal{D}_m([\mathbf{X}_m^{\text{com}}, \mathbf{X}_m^{\text{prt}}])) \right\|_F^2 \quad (3)$$

但对于上述的重构损失，仍然不能完全保证特征解耦。

事实上，信息可能在表示之间自由泄漏，例如，所有的模态信息可能仅仅被编码在 $\mathbf{X}_m^{\text{prt}}$ 中，使得解码器可以轻松地合成输入，从而使得均匀的多模态特征变得无意义。为了巩固特征解耦，我们认为来自相同情感但不同模态的均匀表示应该比来自相同模态但不同情感的表示更相似。为此，我们定义了一个边际损失：

$$\mathcal{L}_{\text{mar}} = \frac{1}{|S|} \sum_{(i,j,k) \in S} \max \left(0, \alpha - \cos \left(\mathbf{X}_{m[i]}^{\text{com}}, \mathbf{X}_{m[j]}^{\text{com}} \right) + \cos \left(\mathbf{X}_{m[i]}^{\text{com}}, \mathbf{X}_{m[k]}^{\text{com}} \right) \right) \quad (4)$$

其中，我们收集了一个三元组集合 $S = \{(i, j, k) \mid m[i] \neq m[j], m[i] = m[k], c[i] = c[j], c[i] \neq c[k]\}$ 。 $m[i]$ 是样本 i 的模态， $c[i]$ 是样本 i 的类标签， $\cos(\cdot, \cdot)$ 表示两个特征向量之间的余弦相似度。公式4中的损失限制了属于相同情感但不同模态或相反的同质特征之间的差异，并避免了推导出平凡的同质特征。 α 是距离边界。正样本（相同情感；不同模态）的距离被约束为小于负样本（相同模态；不同情感）的距离，距离边界为 α 。

考虑到解耦特征分别捕捉到了与模态无关/独特的特征，我们进一步制定了一个软正交性损失函数，以减少同质和异质多模态特征之间的信息冗余：

$$\mathcal{L}_{\text{ort}} = \sum_{m \in \{L, V, A\}} \cos \left(\mathbf{X}_m^{\text{com}}, \mathbf{X}_m^{\text{prt}} \right) \quad (5)$$

最后，我们将这些约束条件结合起来形成解耦损失：

$$\mathcal{L}_{\text{dec}} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{cyc}} + \gamma (\mathcal{L}_{\text{mar}} + \mathcal{L}_{\text{ort}}) \quad (6)$$

3.2 使用解耦多模态特征的GD

对于解耦的同质和异质多模态特征，我们在每个特征上设计了一个图形蒸馏单元（GD-Unit），以进行自适应的知识蒸馏。通常，GD-Unit由一个有向图G组成。设 v_i 是一个节点表示一个模态， $w_{i \rightarrow j}$ 表示从模态 v_i 到 v_j 的蒸馏强度。从 v_i 到 v_j 的蒸馏定义为它们相应逻辑的差异，用 $\epsilon_{i \rightarrow j}$ 表示。设 E 为蒸馏矩阵，其中 $E_{ij} = \epsilon_{i \rightarrow j}$ 。

对于目标模态 j ，加权蒸馏损失可以通过考虑注入边来进行表达，即：

$$\zeta_{:j} = \sum_{v_i \in \mathcal{N}(v_j)} w_{i \rightarrow j} \times \epsilon_{i \rightarrow j} \quad (7)$$

$\mathcal{N}(v_j)$ 表示注入到 v 的顶点集

为了学习与蒸馏强度 w 相对应的动态和自适应权重，我们提出将模态逻辑和表示编码为图的边，如下所示。通过这样的设计，我们可以根据模态逻辑和表示之间的关系来计算权重，以捕捉模态之间的蒸馏强度。

$$w_{i \rightarrow j} = g([f(\mathbf{X}_i, \theta_1), \mathbf{X}_i], [f(\mathbf{X}_j, \theta_1), \mathbf{X}_j]), \theta_2) \quad (8)$$

其中 $[\cdot, \cdot]$ 表示特征串联， g 是一个全连接层，具有可学习的参数 θ_2 ，而 f 是一个用于回归逻辑的全连接层，具有参数 θ_1 。图中的边权重 W ，其中 $W_{ij}=w_{i \rightarrow j}$ ，可以通过对所有模态对重复应用公式（8）进行构建和学习。

3.2.1 HomoGD

HomoGD 同态图蒸馏

对于解耦的同态特征 $\mathbf{X}_m^{\text{com}}$ ，由于各模态之间的分布差距已经足够小，我们将特征 $\mathbf{X}_m^{\text{com}}$ 和对应的逻辑输入到GD-Unit中，根据公式8计算图边矩阵 W 和蒸馏损失矩阵 E 。然后，通过公式9获取整体的同态蒸馏损失。

$$\mathcal{L}_{\text{dtl}} = \|\mathbf{W} \odot \mathbf{E}\|_1 \quad (9)$$

其中 \odot 表示逐元素乘积。

3.2.2 HeteroGD

HeteroGD 异态图蒸馏

解耦的异态特征 X_{prtm} 关注每个模态的多样性和独特特征，因此展现出明显的分布差距。为了缓解这个问题，我们利用多模态Transformer来缩小特征分布差距并建立模态适应性。

多模态Transformer的核心是跨模态注意力单元（CA），它接收来自一对模态的特征并融合跨模态信息。对于MER中的三个模态，每个模态将受到其他两个模态的增强，并将生成的特征进行串联。

```

# 跨模态注意力单元
# (V,A) --> L
h_l_with_as = self.trans_l_with_a(s_l, s_a, s_a)
h_l_with_vs = self.trans_l_with_v(s_l, s_v, s_v)
h_ls = torch.cat([h_l_with_as, h_l_with_vs], dim=2)
h_ls = self.trans_l_mem(h_ls)
if type(h_ls) == tuple:
    h_ls = h_ls[0]
last_h_l = last_hs = h_ls[-1]

# (L,V) --> A
h_a_with_ls = self.trans_a_with_l(s_a, s_l, s_l)
h_a_with_vs = self.trans_a_with_v(s_a, s_v, s_v)
h_as = torch.cat([h_a_with_ls, h_a_with_vs], dim=2)
h_as = self.trans_a_mem(h_as)
if type(h_as) == tuple:
    h_as = h_as[0]
last_h_a = last_hs = h_as[-1]

# (L,A) --> V
h_v_with_ls = self.trans_v_with_l(s_v, s_l, s_l)
h_v_with_as = self.trans_v_with_a(s_v, s_a, s_a)
h_vs = torch.cat([h_v_with_ls, h_v_with_as], dim=2)
h_vs = self.trans_v_mem(h_vs)
if type(h_vs) == tuple:
    h_vs = h_vs[0]
last_h_v = last_hs = h_vs[-1]

```

e.g. 以语言模态 $\mathbf{X}_L^{\text{prt}}$ 为源和视觉模态 $\mathbf{X}_V^{\text{prt}}$ 为目标，从语言到视觉的增强特征表示如下：

$$\mathbf{Z}_{L \rightarrow V}^{\text{prt}} = \text{softmax} \left(\frac{\mathbf{Q}_V \mathbf{K}_L^\top}{\sqrt{d}} \right) \mathbf{V}_L \quad (10)$$

其中 $\mathbf{Q}_V = \mathbf{X}_V^{\text{prt}} \mathbf{P}_q$ 、 $\mathbf{K}_L = \mathbf{X}_L^{\text{prt}} \mathbf{P}_k$ 、 $\mathbf{V}_L = \mathbf{X}_L^{\text{prt}} \mathbf{P}_v$ 为跨模态注意， d 表示 \mathbf{Q}_V 和 \mathbf{K}_L 的维度。

对于每个目标模态，我们将其他模态的所有增强特征与目标模态连接起来作为增强特征，表示为 $\mathbf{Z}_{\rightarrow m}^{\text{prt}}$ ，这些增强特征在蒸馏损失函数公式9中被使用。

3.2.3 特征融合

我们使用增强的异构特征 $\mathbf{Z}_{\rightarrow m}^{\text{prt}}$ 和原始的解耦同质特征 $\mathbf{X}_m^{\text{com}}$ 进行自适应特征融合，其中的权重是从它们各自学习得到的。通过这样的方式，我们得到了用于多模态情感识别的融合特征。

3.3 目标优化

我们将上述任务的损失整合到完整的目标函数中：

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda_1 \mathcal{L}_{\text{dec}} + \lambda_2 \mathcal{L}_{\text{dtl}}$$

其中 $\mathcal{L}_{\text{task}}$ 是情感任务相关的损失（例如均方误差）， $\mathcal{L}_{\text{dtl}} = \mathcal{L}_{\text{dtl}}^{\text{homo}} + \mathcal{L}_{\text{dtl}}^{\text{hetero}}$ 表示由HomoGD和HeteroGD生成的蒸馏损失， λ_1 和 λ_2 控制不同约束的重要性。