Report of Anomaly Detection HW4

Name: 高煒軒

Student ID: 112062646

Implementation & explanation

Problem 1:

```
(smoothing: True): dataset = avenue, auc = 0.7109168969294841  aver_result: [0.6455352547712172]
(smoothing: True): dataset = avenue, auc = 0.8571041438436667  aver_result: [0.8951413949403774]
(smoothing: True): dataset = avenue, auc = 0.7906356584333849  aver_result: [0.8005677138610977]
```

Problem 2:

```
(smoothing: True): dataset = avenue, auc = 0.694730936234571  aver_result: [0.6181524418357716]
(smoothing: True): dataset = avenue, auc = 0.6318882585921665  aver_result: [0.5418812810853003]
(smoothing: True): dataset = avenue, auc = 0.6634296815054975  aver_result: [0.5743266656891045]
```

我認為 binary classification 的任務可能過於簡單，使得 modle 難以從 temporal permutation 中學習到有用的資訊。

■ Modified code

修改 dataset 的 code，使其若有作 temporal permutation 則將 label 設為 1 否則為 0。

```python
class VideoAnomalyDataset_Bi_C3D(VideoAnomalyDataset_C3D):
    """
    Video Anomaly Dataset.
    ===> Modify the temporal permutation for change prediction as a binary classification (0 : normal) / (1 : abnormal).
    """

    def __getitem__(self, idx):
        temproal_flag = idx % 2 == 0
        record = self.objects_list[idx]
        label = 0
        if self.test_stage:
            perm = np.arange(self.frame_num)
        else:
            # ---------------MODIFY HERE---------------
            if random.random() < 0.5:
                perm = np.arange(self.frame_num)
            else:
                perm = np.random.permutation(self.frame_num)
                if np.any(perm != np.arange(self.frame_num)):
                    # Set the label as 1 (abnormal) only when ‘perm’ does not equal the original order.
                    label = 1
        obj = self.get_object(record["video_name"],
                              record["frame"], record["object"])

        if not temproal_flag and not self.test_stage:
            if random.random() < 0.0001:
                spatial_perm = np.arange(9)
            else:
                spatial_perm = np.random.permutation(9)
        else:
            spatial_perm = np.arange(9)
        obj = self.jigsaw(obj, border=2, patch_size=20,
                          permuation=spatial_perm, dropout=False)
        obj = torch.from_numpy(obj)

        clip_id = str(record["frame"]) + '_' + str(record["object"])
```

並且在 main.py 中，將 temp_loss 改為使用 nn.BCEWithLogitsLoss 計算

```
criterion = nn.CrossEntropyLoss(reduction='mean')
temp_criterion = nn.BCEWithLogitsLoss(reduction='mean')
optimizer = optim.Adam(params=net.parameters(), lr=1e-4)
```

```
    temp_logits, spat_logits = net(obj)
    # temp_logits = temp_logits[t_flag].view(-1, args.sample_num)
    temp_logits = temp_logits[t_flag].view(-1)
    spat_logits = spat_logits[~t_flag].view(-1, 9)

    temp_loss = temp_criterion(temp_logits, temp_labels.float())
    spat_loss = criterion(spat_logits, spat_labels)

    loss = temp_loss + spat_loss
```

Problem 3:

```
(smoothing: True): dataset = avenue, auc = 0.677991901253842  aver_result: [0.6243217356891717]
(smoothing: True): dataset = avenue, auc = 0.6074922272438982  aver_result: [0.504837138566415]
(smoothing: True): dataset = avenue, auc = 0.6490599511236678  aver_result: [0.5491909099300794]
```

■ Modified code

使用一個 list(此處命名為 all_perm)記錄所有可能的 permutation，並根據 temporal permutation 的結果返回 list 中對應的 index 作為 multiclass predction 的 label。

```python
def __getitem__(self, idx):
    temproal_flag = idx % 2 == 0
    record = self.objects_list[idx]
    label = 0
    if self.test_stage:
        perm = np.arange(self.frame_num)
    else:
        # ----------------MODIFY HERE----------------
        if random.random() < 0.5:
            perm = np.arange(self.frame_num)
        else:
            perm = np.random.permutation(self.frame_num)
            if np.any(perm != np.arange(self.frame_num)):
                # Set the label as 1 (abnormal) only when ‘perm’ does not equal the original order.
                label = self.all_perm.index(tuple(perm))
    obj = self.get_object(record["video_name"],
                          record["frame"], record["object"])
```

Additional information of Problem 3:

在此次作業的 **problem3** 中我觀察到一些有趣的現象，以此為記錄。

1. **Temporal permutation** 的機率:

   在我上述的 **Modified code** 中 **Temporal permutation** 的機率同 **Problem 2**，

   而我另外測試了將機率設為 **0.0001**。做法為直接將 **dataset** 繼承原

   dataset

```python
class VideoAnomalyDataset_perm_C3D(VideoAnomalyDataset_C3D):
    """
    Video Anomaly Dataset.
    ===> Modify the temporal permutation for change prediction as a permutation predictor i.e. prob. of normal = 1/(simple_num!).
    """

    def __init__(self,
                 data_dir,
                 dataset='shanghaitech',
                 detect_dir=None,
                 fliter_ratio=0.9,
                 frame_num=7,
                 static_threshold=0.1):
        super(VideoAnomalyDataset_perm_C3D, self).__init__(data_dir, dataset, detect_dir,
                                                           fliter_ratio, frame_num, static_threshold)
        self.all_perm = list(permutations(np.arange(self.frame_num)))

    def __getitem__(self, idx):
        ret = super().__getitem__(idx)
        ret['label'] = self.all_perm.index(tuple(ret["label"]))
        return ret
```

   但是最終實驗的結果顯示兩種機率設定並沒有對 **performance** 造成太大的

   影響

```
(smoothing: True): dataset = avenue, auc = 0.6483190444721401  aver_result: [0.6052123423789282]
(smoothing: True): dataset = avenue, auc = 0.5848984552063051  aver_result: [0.4965623012310326]
(smoothing: True): dataset = avenue, auc = 0.6369814651368049  aver_result: [0.5393306739944885]
```

2. **Anomaly score** 的設定:

   上述的 **Problem3** 實驗結果是根據 **pdf**，將 **anomaly score** 設為

   1 – probability of Permutation #1，如下

```python
temp_probs = F.softmax(temp_logits, -1)
# diag2 = torch.diagonal(temp_probs, offset=0, dim1=-2, dim2=-1)
scores2 = 1 - temp_probs[:, 0].cpu().numpy()
```

   但是我在實驗過程中曾經誤將 **anomaly score** 設定如下

```python
temp_probs = F.softmax(temp_logits, -1)
# diag2 = torch.diagonal(temp_probs, offset=0, dim1=-2, dim2=-1)
scores2 = temp_probs[:, 0].cpu().numpy()
```

而這樣的設定卻反而獲得了比原 **problem3** 更加的結果，甚至接近原

problem 1 的結果

```
(smoothing: True): dataset = avenue, auc = 0.6677167833873694  aver_result: [0.6019024598874825]
(smoothing: True): dataset = avenue, auc = 0.8864652033335255  aver_result: [0.8697488969686828]
(smoothing: True): dataset = avenue, auc = 0.7774326176336227  aver_result: [0.7353518981718512]
```