ML2019FALL Final Project - Domain Adaptation

Report

Team Name: NTU_R08521610_Rainforest

r07521603 蔡松霖 r08521602 王鈞平 r08521610 鄭羽霖

Introduction & Motivation (0.5%)

學期間我們一開始是做NLP的題目，但是後來覺得頗有難度，而且訓練時間所需較長，比較不適合最後衝刺XD，因此在學期末轉投入CV這個題目。另一方面這個題目對於我們自己研究室的研究課題幫助也比較大，希望能藉由做這次final project的機會，充實這個領域相關的知識，也許日後可以加入到自己的研究中實作。

Data Preprocessing/Feature Engineering (0.5%)

資料前處理上沒有做什麼特別的處理，只有將target data (testing data) 也resize 成跟source data (有label好的data) 一樣的3 x 32 x 32。

原先另外有嘗試將source data的部分做Canny edge detection，但後來加入Maximum Classifier Discrepancy的model測試時，感覺沒有很大的幫助因此就沒使用了。

Model Description (At least two different models) (1.5%)

(1) kMeans 分成10個cluster

直接使用sklearn裡的kMeans實作，將testing部分的圖片直接分成10個cluster，再大概用肉眼將cluster排一下label的順序。

(2) Maximum Classifier Discrepancy (MCD) – 3 conv layers CNN

後來有比較好的成果皆是使用Maximum Classifier Discrepancy這個方法，實作上主要是參考Maximum Classifier Discrepancy for Unsupervised Domain Adaptation這篇paper的作者放在github上的sample code去改寫，用在我們的task上。Model

的部分主要包含一個特徵生成器generator (G)，以及兩個架構相同但獨立的分類器classifiers (F1, F2)。訓練時分成三個步驟進行迭代訓練：

Step A: Train on source，用已知label的data來訓練G跟classifiers，使其能夠將圖片好好分類，這邊的loss使用的是cross entropy。

Step B: Maximize discrepancy on target (Fix G)，最大化兩個分類器之間的差異。

Step C: Minimize discrepancy on target (Fix F1, F2)，源域和目標域圖片通過特徵生成器得到的特徵更加相近，使兩個分類器的分類一致，檢測不出兩個數據集之間的類別差異。後面兩步的loss依據的是兩個classifier的output取softmax後的L1-distance。

另外paper中有提到，在每一次的迭代中Step C，會update k次，這個k是一個hyper parameter 不同dataset要進行測試找到較佳的參數設定。經過一些測試以及考量運算時間，我們的model選擇k=4作為我們step C部分的更新次數。

Generator跟Classifier的model是參考原作者的model，因為嘗試加入pretrained model的效果都不如預期，另一方面其實input的圖片也蠻小，應該不需要太深，在時間有限的情況下，最後選擇實作比較簡單的model。詳細實作細節如下所列：

特徵生成器Generator (G)

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.conv1 = nn.Conv2d(3, 64, kernel_size=5, stride=1, padding=2)
        self.bn1 = nn.BatchNorm2d(64)
        self.conv2 = nn.Conv2d(64, 64, kernel_size=5, stride=1, padding=2)
        self.bn2 = nn.BatchNorm2d(64)
        self.conv3 = nn.Conv2d(64, 128, kernel_size=5, stride=1, padding=2)
        self.bn3 = nn.BatchNorm2d(128)
        self.fc1 = nn.Linear(8192, 3072)
        self.bn1_fc = nn.BatchNorm1d(3072)

    def forward(self, x):
        x = F.max_pool2d(F.relu(self.bn1(self.conv1(x))), stride=2, kernel_size=3, padding=1)
        x = F.max_pool2d(F.relu(self.bn2(self.conv2(x))), stride=2, kernel_size=3, padding=1)
        x = F.relu(self.bn3(self.conv3(x)))
        x = x.view(x.size(0), 8192)
        x = F.relu(self.bn1_fc(self.fc1(x)))
        x = F.dropout(x, training=self.training)
        return x
```

分類器 Classifier (F1, F2)

```python
class Classifier(nn.Module):
    def __init__(self, prob=0.5):
        super(Classifier, self).__init__()
        # self.fc1 = nn.Linear(8192, 3072)
        # self.bn1_fc = nn.BatchNorm1d(3072)
        self.fc2 = nn.Linear(3072, 2048)
        self.bn2_fc = nn.BatchNorm1d(2048)
        self.fc3 = nn.Linear(2048, 10)
        self.bn_fc3 = nn.BatchNorm1d(10)
        self.prob = prob

    def set_lambda(self, lambd):
        self.lambd = lambd

    def forward(self, x, reverse=False):
        if reverse:
            x = grad_reverse(x, self.lambd)
        x = F.relu(self.bn2_fc(self.fc2(x)))
        x = self.fc3(x)
        return x
```

Optimizer 選用adam，learning rate = 2.5×10^{-4}。
Batch size 分別選用128跟256。

(3) Maximum Classifier Discrepancy (MCD) – pretrained model (ResNeXt50)

這邊附上嘗試pretrained model的generator跟classifier架構，不過後來沒有來得及調整成能用的model，所以就參考就好 QQ

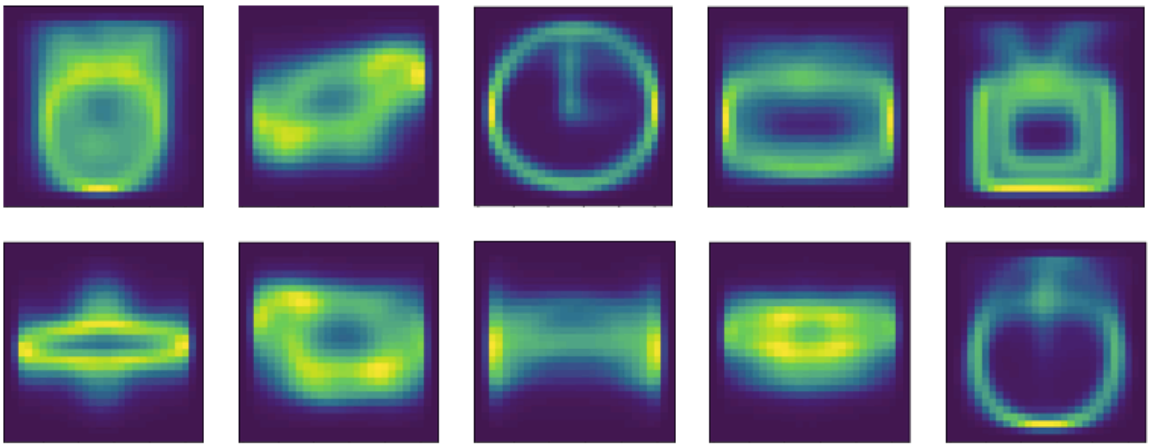
```python
resnet_dict = {"ResNet18": models.resnet18, "ResNet34": models.resnet34,
               "ResNet50": models.resnet50, "ResNet101": models.resnet101,
               "ResNet152": models.resnet152, "ResNeXt50": models.resnext50_32x4d,
               "ResNeXt101": models.resnext101_32x8d}
# in_features: 2048

class ResNetGenerator(nn.Module):
    def __init__(self, model_name):
        super(ResNetGenerator, self).__init__()
        model = resnet_dict[model_name](pretrained=True)
        self.feature_layers = nn.Sequential(*list(model.children())[:-2])
        self.avgpool = model.avgpool
        self.in_features = model.fc.in_features

    def forward(self, x):
        x = self.feature_layers(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), self.in_features)
        x = F.dropout(x, training=self.training)
        return x
```

```python
class FlexClassifier(nn.Module):
    def __init__(self, in_features=2048, hidden_num=2048, num_class=10, prob=0.5):
        super(FlexClassifier, self).__init__()
        self.fc1 = nn.Linear(in_features, hidden_num)
        self.bn1_fc = nn.BatchNorm1d(hidden_num)
        self.fc2 = nn.Linear(hidden_num, num_class)
        self.bn_fc2 = nn.BatchNorm1d(num_class)
        self.prob = prob

    def set_lambda(self, lambd):
        self.lambd = lambd

    def forward(self, x, reverse=False):
        if reverse:
            x = grad_reverse(x, self.lambd)
        x = F.relu(self.bn1_fc(self.fc1(x)))
        x = self.fc2(x)
        return x
```

Experiment and Discussion (3.5%)

在做這項project時，要怎麼衡量我們的結果好不好，我們覺得蠻重要的。我們training 的過程中有三個loss，分別有Classifier F1、F2的cross entropy loss，還有discrepancy loss，前兩者train到後面都差不多，但discrepancy loss大概是影響準確度的關鍵。另外依據每一個類別的分佈個數可以做一個快速的監測，因為助教有提到testing的部分，每一個類別都很平均，假如結果是全部分到某一類大概就是沒train好，可以喊cut。不過在MCD_DA原作者的code裡，testing的部分也有label，感覺如果有這部分的label，就能更直接看出model的好壞。

首先是kMeans做cluster的結果，下圖是10個cluster的中心，可以猜測像是television跟clock這兩個類別 (上3，上5) 應該是有抓到，但是因為其實很多類別蠻相近的，像是bed跟cow (horse)都是四個腳，airplane跟dolphin也很像，因此有的cluster中心看起來就是多種混合在一起。輔以猜測將下列cluster中心做label (有的猜同一個，像是上排最右邊兩個都猜是television)，其餘照片依照kMeans分的結果跟著cluster中心改過去，accuracy最好大概可以落在0.46附近。



Maximum Classifier Discrepancy (MCD)的部分，我們主要在model，batch size，optimizer，跟learning rate上做了一些嘗試。

首先是model使用的部分，如上一個章節所述，當我們使用pretrained model時，效果很不好，train到後面model傾向都預測成其中某一類(如下圖)。

```
Pred1:
2 0 3 99995 0 0 0 0 0 0
0.0% 0.0% 0.0% 99.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
Pred2:
0 0 1 99999 0 0 0 0 0 0
0.0% 0.0% 0.0% 99.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%
```

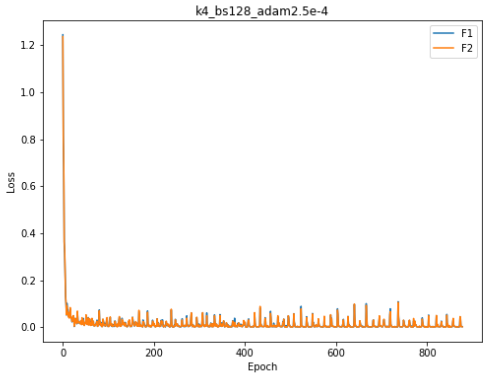
因此我們後來都使用generator是3層卷積層的model，結果合理許多(如下圖)。後面的一些嘗試都僅限於這一個model。

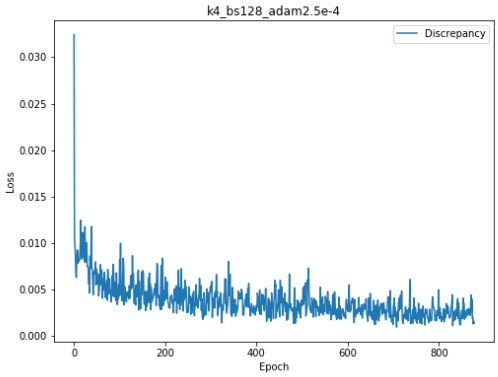
```
Pred1:
10129 8904 11036 10081 8331 9714 10076 11292 11258 9179
10.0% 8.0% 11.0% 10.0% 8.0% 9.0% 10.0% 11.0% 11.0% 9.0%
Pred2:
9753 9020 11009 10142 8329 10327 10093 11258 10992 9077
9.0% 9.0% 11.0% 10.0% 8.0% 10.0% 10.0% 11.0% 10.0% 9.0%
```

Optimizer的部分我們分別選用SGD + momentum跟adam做嘗試，類似上面的觀察發現使用adam作為optimizer時較穩定，可以得到較合理的分佈。
Learning rate的部分經過一些嘗試我們試出取$2.5x10^{-4}$較容易得到較好的結果。Batch size則是大概取128或是256時較好。

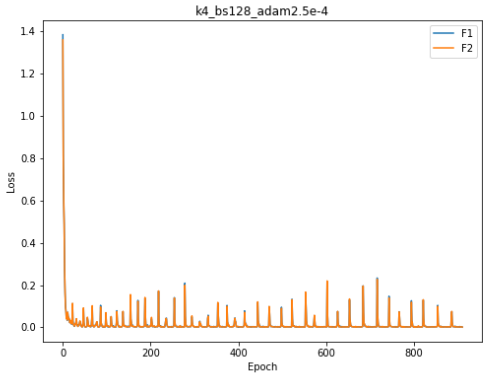
下面是我們有得到較佳結果的兩個job，在這邊列出他們training的loss history。

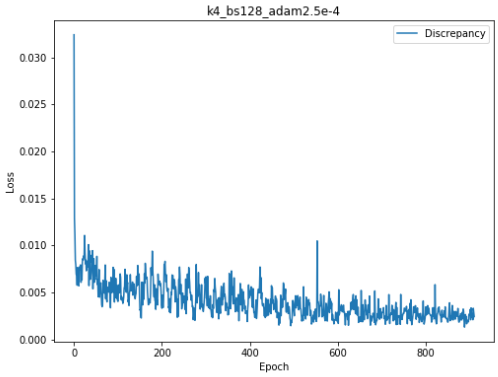
Batch size: 128，adam learning rate: $2.5x10^{-4}$





Batch size: 256，adam learning rate: $2.5x10^{-4}$





基本上，兩個的趨勢蠻相近，batch size較大時偶有的起伏幅度也較大。由這些圖可以看出來，F1跟F2的loss跟準確度的關聯較小，主要還是要依據discrepancy loss，最佳準確度的model在epoch數在6、700附近，都是discrepancy loss比較小的部分。

最後的最佳結果是將8個model (有些是取同一次job的不同epoch) 的output做ensemble所得，public score是0.83703 (第2名)，private score則是0.83470 (第4名QQ)。

Conclusion (1%)

Maximum Classifier Discrepancy真的很強大，能跑出一些準確度高達0.8的model，當然其中也有一點點運氣成分在其中，因為那個learning rate真的是只有在$2.5x10^{-4}$時，我們才能比較穩定得到準確度大於0.7的結果，其他情況下大概只能達到0.6附近甚至更低。後續我們覺得model應該還有很大的進步空間，可能可以參考一些做CIFAR-10這個數據集的一些 state-of-the-art 的model來改良我們的model，做出更好的成果。

References (1%)

[1] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, et al. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation, 2018

Other web sources:

1. https://kknews.cc/zh-tw/code/4nvmnqq.html
2. https://zhuanlan.zhihu.com/p/52085426
3. https://paperswithcode.com/sota/image-classification-on-cifar-10

Code reference:

MCD_DA official github: https://github.com/mil-tokyo/MCD_DA

組內分工

羽霖擔任組長處理所有行政事務，包含kaggle組隊，填表單，最後整理project並上傳github，講幹話維繫組員之間感情

松霖主要處理程式的部分，投入調參，嘗試不同model，整理結果，撰寫report

鈞平協助嘗試不同的model，嘗試canny edge detection，幫忙做ensemble的部分