Computer Vision 2019

HW4 report

108061613 黃雅琳

Part 1. Transfer Learning in CNN PyTorch

Q1-1. Please report the validation accuracy of a pretrained Alexnet used as a feature extractor in the two-class classification problem. (5 pts)

ps. Only 4096x2 layer would be finetuned

1. 限制其他層做 back propagation，清除最後一層，改為 binary classifier。

```
model_conv = models.alexnet(pretrained=True)
for param in model_conv.parameters():
    param.requires_grad = False

# Parameters of newly constructed modules have requires_grad=True by default
model_conv.classifier = nn.Sequential(*[model_conv.classifier[i] for i in range(6)]) # remove the last
addition_fc = nn.Linear(4096, 2) # the layer to be stacked
model_conv.classifier = nn.Sequential(model_conv.classifier,addition_fc)

model_conv = model_conv.to(device)
```

Fig1: model setting of Q1-1

2. Check the structure:

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4,
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0,
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1,
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0,
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0,
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bi
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bi
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=1000, bi
  )
)
```

Fig2: model structure of pretrained Alexnet

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stri
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, paddin
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Sequential(
      (0): Dropout(p=0.5, inplace=False)
      (1): Linear(in_features=9216, out_features=40
      (2): ReLU(inplace=True)
      (3): Dropout(p=0.5, inplace=False)
      (4): Linear(in_features=4096, out_features=40
      (5): ReLU(inplace=True)
    )
    (1): Linear(in_features=4096, out_features=2, b
  )
)
```

Fig3: model structure of pretrained Alexnet set the output equals to 2

3. Best validation Acuuracy = 0.810000

Q1-2 Please report the validation accuracy of a pretrained Alexneta fter it is fine-tuned in the two-class classification problem. (5 pts)

ps. Please try to finetune every layer

1. 每層都會做 back propagation (param.required_grad=(default:True))

```
## Alexnet
model_ft = models.alexnet(pretrained=True)
model_ft.classifier = nn.Sequential(*[model_ft.classifier[i] for i in range(6)])  # remove the last la
addition_fc = nn.Linear(4096, 2) # the layer to be stacked
model_ft.classifier = nn.Sequential(model_ft.classifier,addition_fc)
#model_ft = nn.Sequential(model_ft,addition_fc)
print(model_ft)
##

model_ft = model_ft.to(device)
```

Fig4: model setting of Q1-2

2. Best validation accuracy = 0.885000

Q1-3 Please report the validation accuracy of a non-pretrained Alexnet after it is trained in the two-class classification problem. (5 pts)

ps. Alexnet is trained from scratch

1. Setting : pretrained = False

```
## Alexnet
model_np = models.alexnet(pretrained=False)
model_np.classifier = nn.Sequential(*[model_np.classifier[i] for i in range(6)])  # remove the last la
addition_np = nn.Linear(4096, 2) # the layer to be stacked
model_np.classifier = nn.Sequential(model_np.classifier,addition_fc)

print(model_np)
##

model_np = model_np.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that all parameters are being optimized
optimizer_np = optim.SGD(model_np.parameters(), lr=0.001, momentum=0.9)

# step size could be
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_np, step_size=5, gamma=0.1)
```

Fig5: model setting of Q1-3

2. Best validation accuracy = 0.742500

Q1-4 Please discuss the results of Q1-1, Q1-2, & Q1-3. (5 pts)

	Alexnet as…	Initial validation Accu	Best validation Accu.
Case 1	Pretrained Feature Extractor	0.775	0.810
Case 2	Pretrained and fintuned all layers	0.625	**0.885**
Case 3	Non-pretrained, but trained by our data all layers	0.5	0.743

比較一開始的Initail validation accuracy，只使用alexnet as feature extracture有比較好的結果，但最後於25個epoch結束後，表現最好的卻是finetuned all layers的pretrained alexnet。這是因為一開始的alexnet 還在利用我們的data結果尋找對的weight，故一開始weight 較不穩定→ case 2的initial Accr會較case 1差。然而，因為model feature extract有被我們的圖片訓練過，故最後反而會抓到更符合我們所需的feature。反之，case 1只能抓到以前用來預訓練Alexnet所認為重要的feature，故accuracy略低case 2一點也是正常的。

而case 3 和 case 2都是作為訓練整個model的設定，卻有不同的結果的原因是因為，要將一個深的模型訓練好，需要大量的數據跟時間去抓出node之間的weight，而case 3是從頭開始訓練，case 2 則是站在預訓練的結果基礎上，去找更好的weight，故能夠更快獲得好的結果。

不過，我認為並不是每個case都適合全層訓練，以大型模型像是Bert來講，要訓練這種大型模型會吃掉很多資源，並不是每個使用者都有辦法對其作finetune，反而會選擇使用第一種方法，利用pretrianed好的經典或是目前最好的模型作為feature extractor，再加入其他分類模型/分類器。

Q1-5. Please try to correct the data augmentation strategy in order to let the entire face of each image be seen and report the validation accuracy of a pre-trained Alexnetas a feature extractor in the two-class classification problem. (5 pts)

1. **Augmentation:**

(1) 檢查原本 data_transform 的augmentation setting 會對data有甚麼影響:

```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

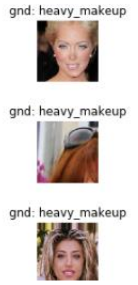


Fig6 : original augmentation strategy

Fig7: the Imgage result under original augmentation strategy

能看出會隨機crop到圖的一塊，故臉部並沒有包含在feature裡面。

(2)

```
data_transforms_2 = {
    'train': transforms.Compose([
        transforms.Resize(256),
        transforms.ColorJitter(contrast=0.25,brightness =0.25),
        transforms.RandomRotation(degrees =15),
        # transforms.RandomOrder[
        #transforms.RandomResizedCrop(224),
        transforms.CenterCrop(224),
        transforms.RandomHorizontalFlip(),
        #],
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

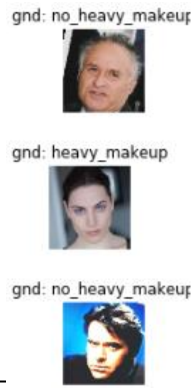


Fig8 : my augmentation strategy

Fig9: the Imgage result under my augmentation strategy

~~RandomResizedCrop(224)~~ → Resize(256) + CentorCrop(224)

: 觀察dataset 裡的照片，大部分的人臉都在影像中央，故使用CentorCrop較為合適。

ColorJitter(contrast=0.25,brightness=0.25)

: 給training data增加一點對比度跟亮度的些微變化，因為test data的亮度、對比度不一定一樣。

RandomRotation(degree=15) : 隨機旋轉影像

2. **Model:** only last layer is trainable, the upper layers worked as feature extractor

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Sequential(
      (0): Dropout(p=0.5, inplace=False)
      (1): Linear(in_features=9216, out_features=4096, bias=True)
      (2): ReLU(inplace=True)
      (3): Dropout(p=0.5, inplace=False)
      (4): Linear(in_features=4096, out_features=4096, bias=True)
      (5): ReLU(inplace=True)
    )
    (1): Linear(in_features=4096, out_features=2, bias=True)
  )
)
```

Fig 10: model of un-pretrained alexnet,

the structure is as same as pretrained one.

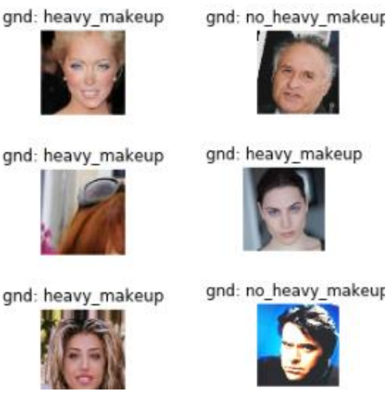
3. **Best validation accuracy** =0.8800

Q1-6. Please try to correct the data augmentation strategy in order to let the entire face of each image be seen and report the validation accuracy of a pretrained Alexnet after it is fine-tunedin the two-class classification problem. (5 pts)

1. **Augmentation: 同 Q1-5 : augmentation**
2. **Model : all the layer in the pretrained Alexnet are trainable**
3. **Best validation accuracy** =0.887500

Q1-7. Please discuss the results of Q1-5 & Q1-6. (5pts)

	Alexnet as…	Data augmentation	Init. Val. Accu.	Best Val. Accu
Case 1	Pretrained Feature Extractor	(default)	0.775	0.810
Case 2	Pretrained and fintuned all layers	(default)	0.625	0.885
Case 4	Case 1 +	My augmentation strategy	0.74	0.88
Case 5	Case 2 +		0.7625	**0.8875**



Left : picture of default augmentation strategy

Right: picture of **my** augmentation strategy

觀察左側，能發現原本的augmentation strategy可能會擷取到沒有人臉特徵的部分，故我更改了切割圖片的的位置，並且引入一些其他augmentation 的技巧。

Before	After	原因
Random-resizedcrop (224)	Resize(256) + CentorCrop(224)	觀察dataset 裡的照片，大部分的人臉都在影像中央，故使用CentorCrop較為合適。
-	ColorJitter (contrast=0.25,brightness=0.25)	給training data增加一點對比度跟亮度的些微變化，因為test data的亮度、對比度不一定一樣。
-	RandomRotation(degree=15)	小幅度隨機旋轉影像

如此確認擷取到的影像是在人臉五官的部分後，希望model學到的確實是臉上的feature而不是背景。

以結果來看的話，加入我的augmentation strategy後，accuracy確實提升 ！

而且，同Q1-4的結論，做full-layer finetune的模型之最終預測準確率會高於以pretrained model作為feature extractor的結果。

Q1-8. Please try to achieve validation accuracy higher than 89.5% using a CNN other than Alexnet& ResNet-18 in the fine-tuning case. (20pts)
ps. Please use the correct data augmentation strategy to achieve the best results.

在此題，我嘗試使用了一些網路上常用的 CNN structure 和兩種經典模型(shuffleNet, modbileNet)，並以不同的 optimizer 跟參數去做調整。

最後，我使用了 mobilenet_v2 作為我的 pretrained model，使用 Adam 作為我的 optimizer (參數: lr=1e-3)

```
Training complete in 10m 46s
Best val Acc: 0.907500
```

Fig 11. Best Validation Accuracy of the pretrained model I chose

Part 2. Semantic Segmentation

Q2-1 **Please try to “eliminate” the skip-connection so the output of convolution layers of FCN8s will be directly upsampledfor 32x. Please report pixel accuracy and mIOUbefore and after. (10 pts)**

Before eliminating the skip-connection (x3,x4) :

```
The highest mIOU is 0.4231790899422671 and is achieved at epoch-17
The highest pixel accuracy  is 0.8459854125976562 and is achieved at epoch-17
```

After eliminating the skip-connection (x3,x4) :

```
The highest mIOU is 0.41727945199707206 and is achieved at epoch-20
The highest pixel accuracy  is 0.83629150390625 and is achieved at epoch-17
```

Q2-2. Please discuss the results of Q2-1. (10 pts)
ps. Is skip connection quantitatively beneficial?

Skip connection 是為了解決當Network層數增加後導致的梯度消散跟梯度爆炸的問題，又有一說是能透過打破網路的對稱性，改善權重矩陣的退化[註一]。
故**如試驗結果，在將skip-connection拿掉之後， mIOU和pixel accuracy都下降了**，這就是因為在反向傳播時，會進行微分的動作，可能得到小於一的數字。而多層數的連乘會導致很多維度的數字變小，信息表達能力嚴重不足。而在影像中，skip-connectin 能夠恢復空間的訊息，透過融合deep layer的semantic information，和shollow layer的appearance，提高分割性能。
故skip connection現在已經廣泛的運用在深的網路之中。
skip connection除了可以連接上下層，於Dense Net更是實現了跨層連接。
所以可以說，skip connection is quantitively beneficial.

[註一]Orhan A E, Pitkow X. Skip connections eliminate singularities[J]. arXiv preprint arXiv:1701.09175, 2017.

Q2-3. Please try to further reduce the number of classes from 11 to 3 and report the pixel accuracy & mIOUof FCN8s. (10 pts)

```
The highest mIOU is 0.47511925782787917 and is achieved at epoch-11
The highest pixel accuracy  is 0.32386505126953125 and is achieved at epoch-13
```

Q2-4. Please discuss the results of Q2-3. Was mIOU increased when the number of classes reduce? Please explain why! (10 pts)

在將class number 數量由11減少到3之後，mIOU增加了 (0.423 → 0.475)，這是因為要分的種類變少了，相較於原本的11類，可能原本有些介於模糊地帶的data，之前沒有成功被識別出來，但在三種種類裡，卻會成功被歸類在某一類別裡。

而且類別減少，代表單一類別的training data量變多，他能夠學習的數據也就更多，有可能學到重要的資訊。