Announcement

- Midterm exam: 10/29 (next Tuesday), 14:10-16:00, Applied Science and Technology Building (應用科技大樓) 336 and 338 classroom
- Practice system: https://smartypantspal.com
- 有相關問題可以寄 email 至 aidslab902@gmail.com

Dropout before softmax layer?

- Dropout “**通常**“不會在Softmax Layer的前一層使用
- 這裡指的 softmax layer 應該改成 softmax function 更為準確

遷移學習與微調

授課老師: 楊景明

Callbacks

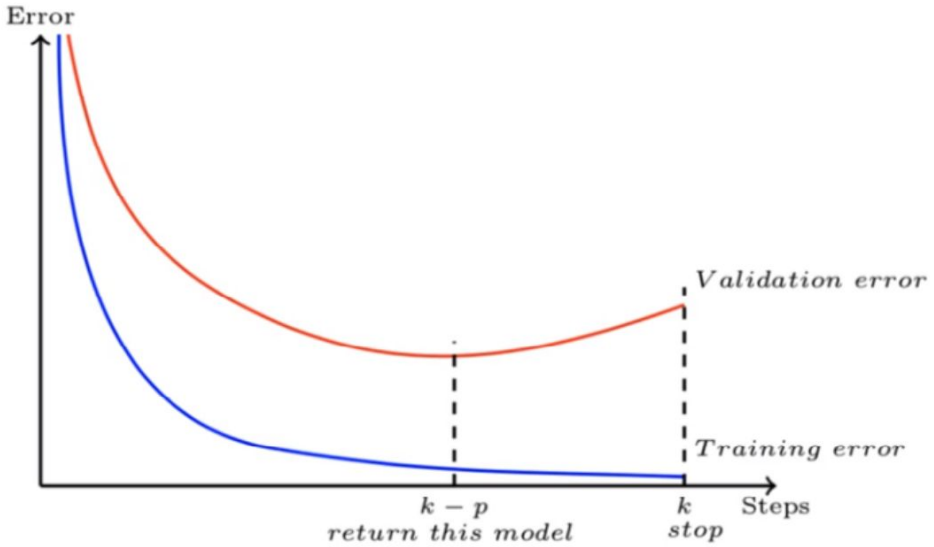
- CallBacks 用於在訓練過程中的不同階段執行不同的操作
- Why Callbacks
  - 假如我們想在每個 epoch 之後保存每個模型呢？
  - 如果我們設定訓練 100 個 epochs，但在 30 個 epochs 後，它開始過度擬合，該怎麼辦？可以停止訓練嗎？
  - 如果我們想將資訊記錄在某處以便稍後分析呢？

Callbacks

- Callbacks 可以用來:
  - Early Stopping
  - Model Checkpointing
  - Learning Rate Scheduler
  - Logging
  - Remote Monitoring
  - Custom Functions

Early Stopping

- 在訓練過程中，驗證損失可能會停滯或停止減少，甚至開始增加（過度擬合）
- 這時候我們可以使用 Callbacks 來實現 Early Stopping



Model Checkpointing

- 在訓練期間，我們可以在每個 epoch 後定期保存權重，讓我們能夠在發生意外時恢復訓練
- 檢查點檔案包含模型的權重或模型本身

```
tf.keras.callbacks.ModelCheckpoint(
    filepath,
    monitor="val_loss",
    verbose=0,
    save_best_only=False,
    save_weights_only=False,
    mode="auto",
    save_freq="epoch",
    options=None,
    **kwargs
)
```

Learning Rate Scheduler

- 我們可以透過將學習率降低指定的值來避免損失在全局最小值附近波動
- 如果我們的監控指標(通常是 val_loss)沒有看到任何改善，我們會等待一定數量的 epoch，然後這個 callback 會將學習率降低一個因子

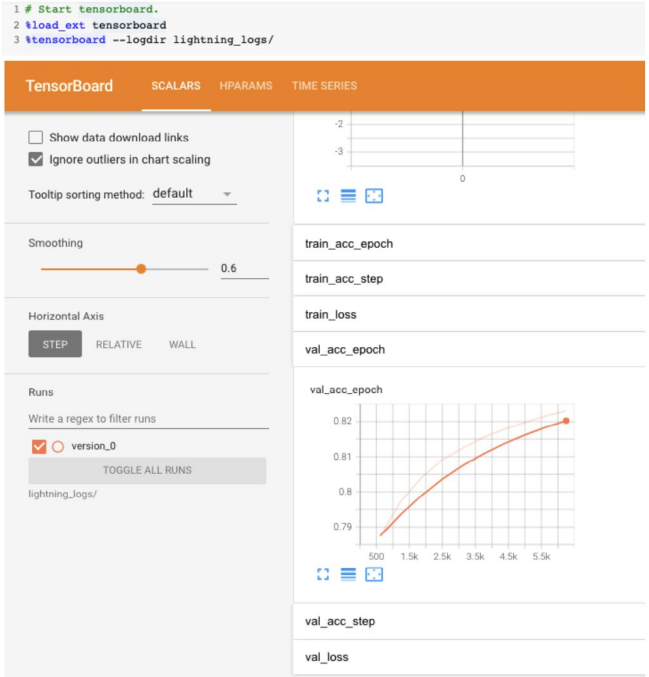
```
from keras.callbacks import ReduceLROnPlateau

reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor = 0.2, patience = 3, verbose = 1, min_delta = 0.0001)
```

Logging

- 我們可以自動記錄模型的訓練統計資料(訓練和驗證損失/準確性)，並在之後使用 TensorBoard 或其他工具查看它們

```
my_callbacks = [
    tf.keras.callbacks.EarlyStopping(patience=2),
    tf.keras.callbacks.ModelCheckpoint(filepath='model.{epoch:02d}-{val_loss:.2f}.h5')
    tf.keras.callbacks.TensorBoard(log_dir='./logs'),
]
model.fit(dataset, epochs=10, callbacks=my_callbacks)
```

```
tf.keras.callbacks.TensorBoard(
    log_dir="logs",
    histogram_freq=0,
    write_graph=True,
    write_images=False,
    update_freq="epoch",
    profile_batch=2,
    embeddings_freq=0,
    embeddings_metadata=None,
    **kwargs
)
```



遷移學習

- 在實務上，很少有人從頭開始訓練整個 CNN（即權重的隨機初始化），因為擁有足夠大小的資料集相對較少
- 相反，通常在非常大的資料集（例如 ImageNet，包含 120 萬張圖像和 1000 個類別）上預先訓練 ConvNet，然後使用 ConvNet 作為感興趣任務的初始化或固定特徵提取器。

遷移學習原理

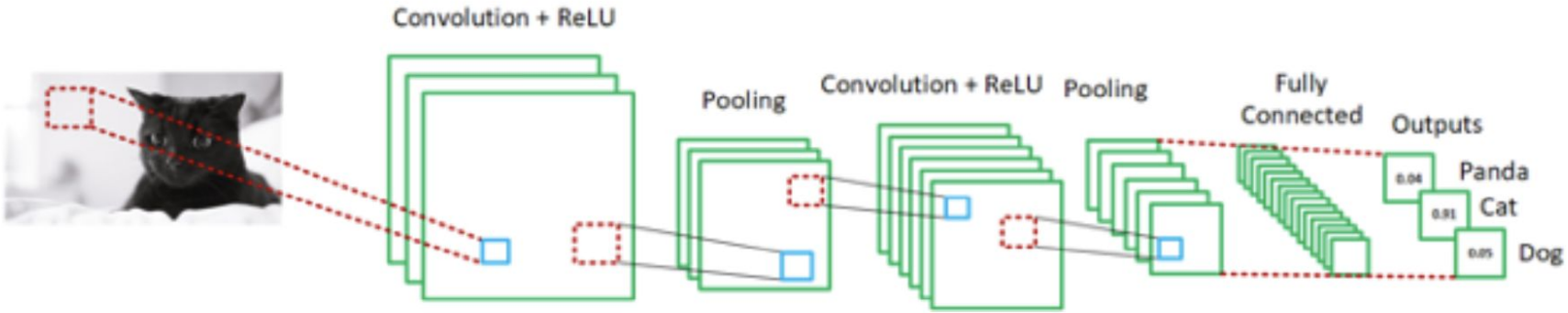
- 在 ImageNet 等大量影像資料集上訓練廣泛的模型有時可能需要數週時間
- 經過如此多資料訓練的模型將具有有用的嵌入，可以應用於其他影像領域，例如邊緣偵測器、圖案和斑點偵測器
- 例如，經過訓練以檢測老虎、獅子和馬的 ImageNet 模型可能有助於檢測其他 4 足哺乳動物
- 遷移學習是我們利用其他領域經過訓練的模型來獲得很高的準確性和更快的訓練時間的概念

遷移學習種類

- 特徵提取
- 微調

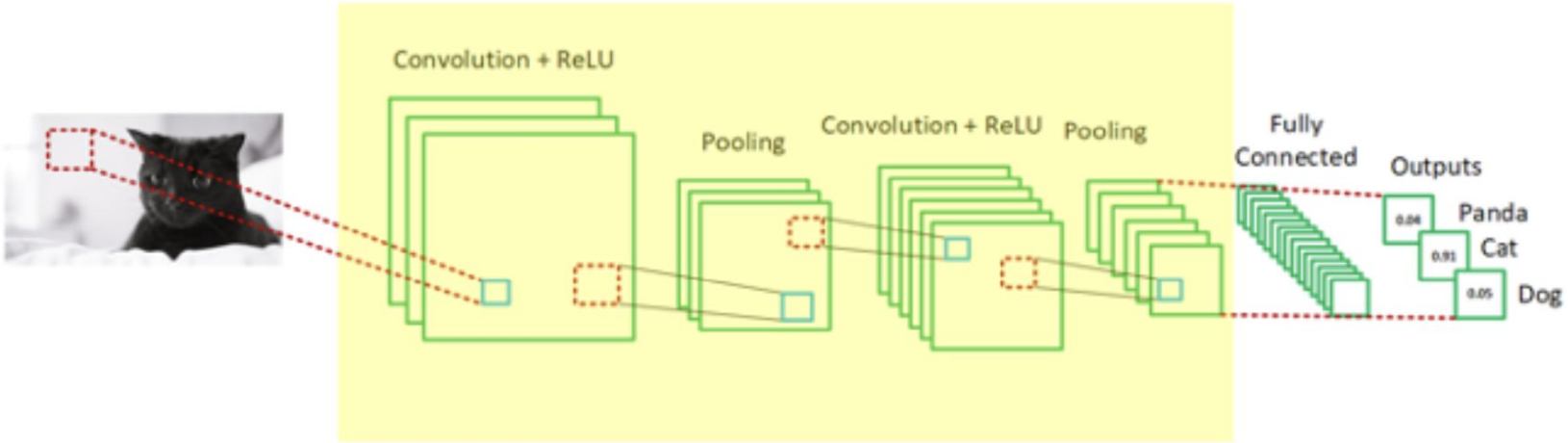
特徵提取

- 取一個預先訓練好的網絡



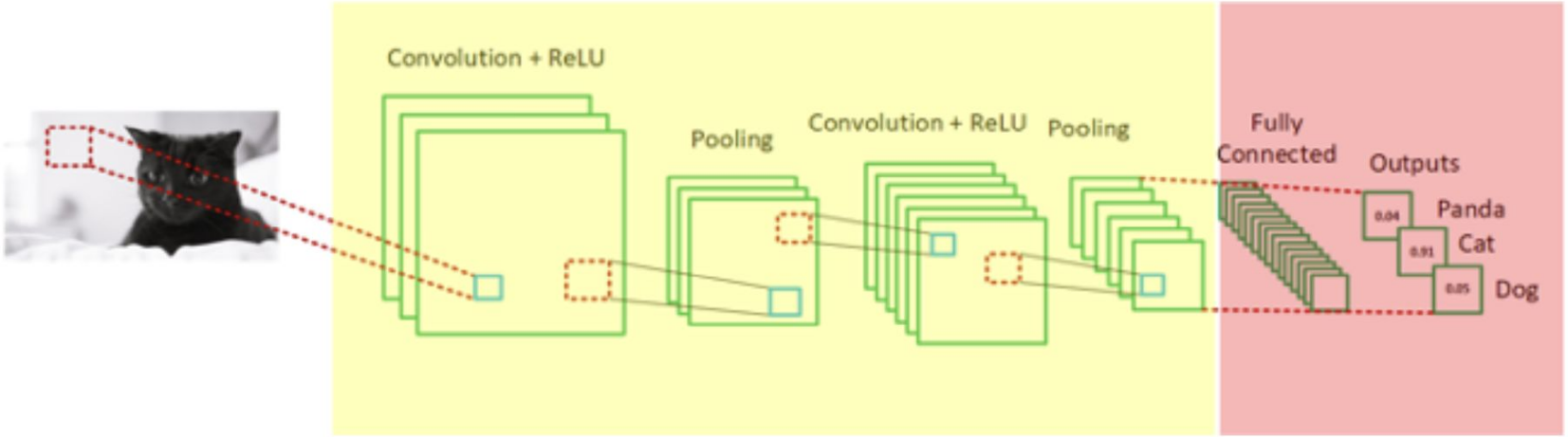
特徵提取器

- Freeze the CONV weights/layers



特徵提取器

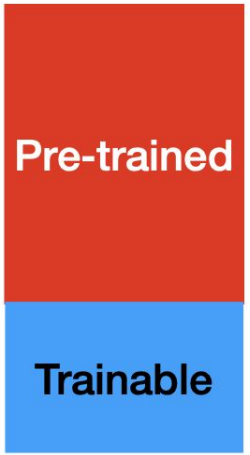
- 替換頂層


Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
Fully Connected
Outputs
Panda
Cat
Dog

特徵提取器

進行特徵提取的步驟

- 凍結預訓練模型的底層
- 將模型的上半部分替換為你的上半部分，以便它僅輸出資料集中的類別數
- 在新資料集上訓練模型



微調

- 在微調中，我們完成特徵提取中的所有步驟，接著：
  - 解凍全部或部分預訓練模型
  - 訓練模型幾個 Epochs，來「微調」預訓練模型的權重
- 因為 ConvNet 中的早期特徵圖學習通用特徵，而後面的層則學習有關圖像資料集的細節。透過微調，我們將這些細節從預訓練模型更改為我們資料集的細節

何時該使用遷移學習

- 最理想情況 - 新資料集很大且與預先訓練的原始資料集相似。模型不太會過度擬合
- 非理想但仍推薦 - 新數據很大但不同
- 如果資料很小，遷移學習和微調通常會過度擬合訓練資料。一個可行的做法是在 CNN 輸出上訓練簡易的分類器，例如：線性分類器

使用遷移學習時的建議

- 學習率 - 對預訓練模型使用非常小的學習率，尤其是在微調時。因為原本相關的權重已經很好，因此不需要過度改變

Implementation

Download notebook at:
https://github.com/albert831229/nchu-computer-vision/tree/main/113/day