MATLAB與深度學習

深度學習是目前人工智慧的主流，透過MATLAB，我們可以簡單且容易的實現深度學習相關的應用。

前言

"深度學習"(Deep Learning)是機器學習範疇中的分支，其根源為類神經網絡(Artificial Neural Net work)，利用疊加多層的網絡架構來處理複雜的非線性問題。在這樣的架構中，深度學習的網絡可以從資料當中取出特徵(feature)，並得到最後的結果，有時該結果甚至可以超越人類辨識準確度。近期G oogle以AlphaGo人工智慧圍棋程式擊敗排名頂尖的世界棋手、IBM Watson的益智問答比賽等，都是讓人為之驚豔的應用。除了這些特殊應用外，深度學習也被廣為應用在影像的分類、語音的辨識等，可以說是目前人工智慧的主流，MATLABR也納入了深度學習的相關功能。

深度學習模型的訓練 (Training from Scratch)

從以上的影片示範，我們可以知道要能辨識影像需要事先訓練深度學習模型。

深度學習模型的準確性在很大程度上取決於用於訓練模型的數據量。準確的模型需要數千甚至數百萬個樣本，這可能需要很

長時間才能訓練完成。一旦模型被訓練完成，它可以用於即時應用，例如先進駕駛輔助系統（Advanc ed Driver Assistance Systems; ADAS）中的行人偵測。

模型的訓練包含以下四個步驟:

i. 資料集的準備
ii. 建立深度學習模型架構
iii. 訓練模型
iv. 測試模型

而這四個步驟在MATLAB中的實現方式如下:

1. 透過imageDatastore指令以及其中所提供的參數，使用者能輕鬆定義影像資料儲存的位置以及標示類別名稱的方式:

```
digitDatasetPath =
fullfile(matlabroot,'toolbox','nnet','nndemos', ...
    'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
    'IncludeSubfolders',true,'LabelSource','foldernames');
```

2. MATLAB提供許多Convolution Neural Network (CNN)模型層的指令，例如:

影像輸入層(im ageInputLayer)、二d卷積層(convolution2dLayer)等，方便使用者自訂CNN架構。

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

3. 將訓練資料、CNN模型架構以及訓練參數輸入trainNetwork指令即可開始進行模型的訓練。

```
options =
trainingOptions('sgdm','MaxEpochs',15,'InitialLearnRate',0.0001
);

convnet = trainNetwork(trainImageData,layers,options);
```

4. 最後使用者能用clssify指令來測試模型的正確性。

```
YTest = classify(convnet,testImageData);
```

2-1 two examples Deep Learning Network for Classification

1. Create Simple Deep Learning Network for Classification

此示例說明如何創建和訓練用於深度學習分類的簡單卷積神經網絡(a simple convolutional neural network for deep learning classification)。 卷積神經網絡是深度學習必不可少的工具，尤其適用於圖像識別。

- The example demonstrates how to：
  - 加載和探索圖像數據。
  - 定義網絡架構。
  - 指定訓練選項 Specify training options。
  - 訓練網絡。
  - 預測新數據的標籤併計算分類準確率 (Predict the labels of new data and calculate the classification accuracy)。

有關顯示如何交互式(interactively)創建和訓練簡單圖像分類網絡 examples，請參閱 see Create Simple Image Classification Network Using Deep Network Designer.

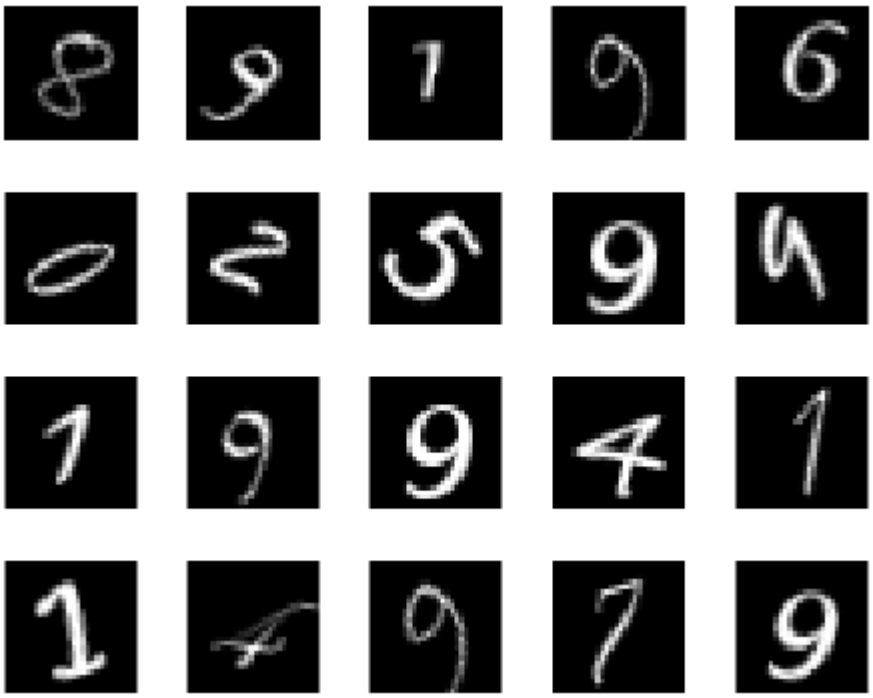
加載和探索圖像數據(Load and Explore Image Data)

將數字樣本 image 加載為圖像數據存儲 image datastore。 imageDatastore 根據文件夾名稱自動標記圖像並將數據存儲為 ImageDatastore 。 圖像數據存儲，使您能夠存儲大型圖像數據，including data that does not fit in memory, and efficiently read batches of images 批量圖像 during training of a convolutional neural network.

```
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet','nndemos', ...
    'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
    'IncludeSubfolders',true,'LabelSource','foldernames');
```

顯示儲存中的一些圖像(in the datastore).

```
figure;
perm = randperm(10000,20);
for i = 1:20
    subplot(4,5,i);
    imshow(imds.Files{perm(i)});
end
```



計算每個類別中的圖像數量。 labelCount 是一個表，其中包含標籤和具有每個標籤的圖像數量。 數據存儲包含 1000 張圖像，每個數字 0-9，總共 10000 張圖像。
您可以將網絡最後一個完全連接層中的類別數指定為 OutputSize 參數。

```
labelCount = countEachLabel(imds)
```

```
labelCount=10×2 table
    Label    Count
    _____    _____

      0      1000
      1      1000
      2      1000
      3      1000
      4      1000
      5      1000
      6      1000
      7      1000
      8      1000
      9      1000
```

您必須指定網絡輸入層中圖像的大小。 檢查 digitData 中第一個圖像的大小。 每個圖像都是 28x28x1 像素

```
img = readimage(imds,1);
size(img)
```

```
ans = 1×2

    28    28
```

Specify Training and Validation Sets

將數據分為訓練和驗證數據集，使訓練集中的每個類別包含 750 張圖像，驗證集包含每個標籤的剩餘圖像。 splitEachLabel 將數據存儲 digitData 拆分為兩個新的數據存儲，trainDigitData 和 valDigitData

```
numTrainFiles = 750;
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
```

Define Network Architecture

Define the convolutional neural network architecture.

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
```

```
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

Image Input Layer imageInputLayer 是您指定圖像大小的地方，在本例中為 28x28x1。 這些數字對應於高度、寬度和通道大小。 數字數據由灰度圖像組成，因此通道大小（顏色通道）為 1。對於彩色圖像，通道大小為 3，對應 RGB 值。 您不需要打亂數據，因為 trainNetwork 默認情況下會在訓練開始時打亂數據。 trainNetwork 還可以在訓練期間的每個 epoch 開始時自動打亂數據。

Convolutional Layer 在卷積層中，第一個參數是 filterSize，它是訓練函數在掃描圖像時使用的過濾器的高度和寬度。 在此示例中，數字 3 表示過濾器大小為 3x3。 第二個參數是過濾器的數量，numFilters，它是連接到輸入的同一區域的神經元的數量。 該參數決定了特徵圖的數量。 使用 'Padding'（名稱-值）對輸入特徵圖添加填充。 對於 default stride 為 1 的捲積層，“相同”填充可確保空間輸出大小與輸入大小相同。 您還可以使用 convolution2dLayer 的(名稱-值)對參數定義該層的步幅 stride 和學習率。

Batch Normalization Layer 批量歸一化 Batch Normalization 層對通過網絡傳播的激活和梯度進行歸一化，使網絡訓練成為更容易的優化問題。 在卷積層和非線性層(例如 ReLU 層)之間，使用批量歸一化層，以加速網絡訓練並降低對網絡初始化的敏感性。

Max Pooling Layer 卷積層（具有激活函數）有時會進行下採樣(down sample) 操作，以減小特徵圖的空間大小並刪除冗餘空間信息。 下採樣可以在不增加每層所需計算量的情況下增加更深卷積層中的濾波器數量。

下採樣(down sample) 的一種方法是使用最大池化，您使用 maxPooling2dLayer 創建它。 最大池層返回輸入矩形區域的最大值，由第一個參數 poolSize 指定。 在這個例子

中，矩形區域的大小是 [2,2]。 'Stride'（名稱-值）對參數，指定訓練函數在沿輸入掃描時採用的步長。

Fully Connected Layer(全連接層) 卷積層和下採樣層之後是一個或多個全連接層。 顧名思義，全連接層是其中的神經元連接到前一層中的所有神經元的層。 該層結合了圖像中前一層學習到的所有特徵，以識別更大的圖案。 最後一個全連接層結合特徵對圖像進行分類。 因此，最後一個全連接層中的 OutputSize 參數等於目標數據中的類別數。 在此示例中，輸出大小為 10，對應於 10 個類。 使用 fullyConnectedLayer 創建一個全連接層。

Softmax Layer softmax 激活函數對全連接層的輸出進行歸一化 normalizes the output。 softmax 層的輸出由總和為 1 的正數組成，然後可以將其用作分類層的分類概率。 在最後一個全連接層之後使用 softmaxLayer 函數創建一個 softmax 層。

Classification Layer 最後一層是分類層。 該層使用 softmax 激活函數為每個輸入返回的概率 probabilities，將輸入分配給互斥類別之一，同時計算損失。 要創建分類層，請使用 use classificationLayer.

Specify Training Options

定義網絡結構後，指定訓練參數選項。 使用初始學習率為 0.01 的隨機梯度下降加上動量 gradient descent with momentum (SGDM) 訓練網絡。 將 epoch 的最大數量設置為 4。一個 epoch 是整個訓練數據集上的完整訓練週期。 通過指定驗證數據和驗證頻率 validation data and validation frequency，來監控訓練期間的網絡準確性。 每個 epoch 都對數據進行洗牌 Shuffle。 The software 在訓練數據上訓練網絡，並在訓練期間定期計算驗證數據的準確性。 驗證數據(The validation data)不用於更新網絡權重 (not used to update the network weights)。 打開訓練進度圖 training progress plot，關閉命令窗口(verbose)輸出。

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

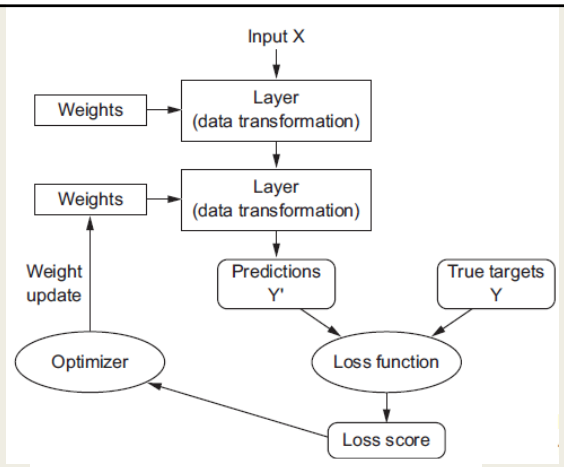
Train Network Using Training Data

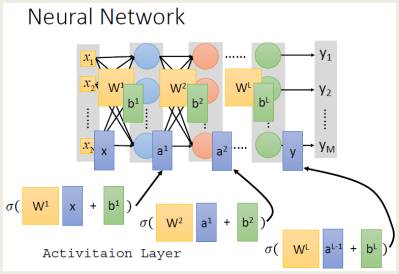
使用由(Layers)、訓練數據(imdsTrain)和訓練選項(options)定義的架構來訓練網絡。 默認(default))情況下，trainNetwork 使用可用的 GPU，否則使用 CPU。 在 GPU 上訓練需要 Parallel Computing Toolbox™ 和支持的 GPU 設備。 有關受支持設備的信息，

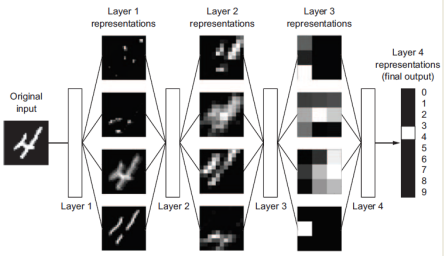
Training process 1. Define the loss function.

2. 求梯度：Evaluate the gradient function through backpropagation algorithm for each weight & bias. (i.e. to change the weight & bias, it will lead what change for the loss function)

3. 梯度法：In each batch, change the weight & bias to minimize the loss function in the gradient direction to achieve the minimum of the loss function.







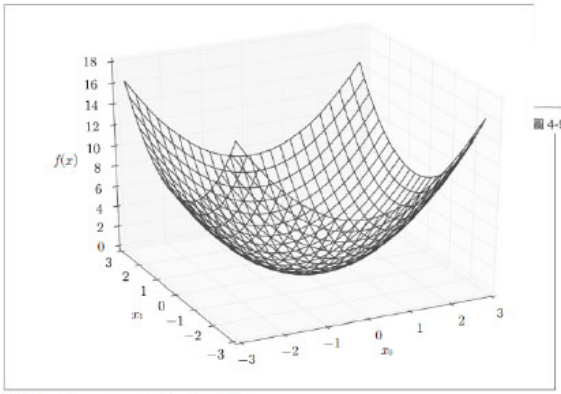


图4-8 $f(x_0, x_1) = x_0^2 + x_1^2$ 的图像

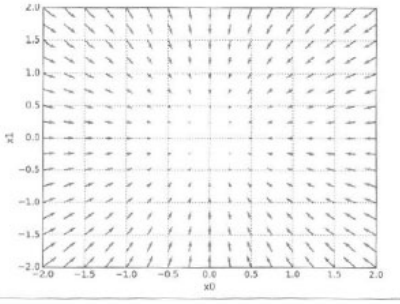
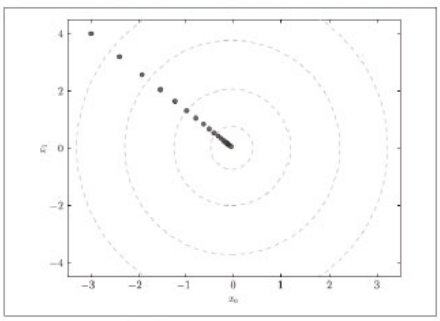


图4-9 $f(x_0$



訓練神經網路，預測數據並計算準確度

- 使用訓練用資料訓練神經網路

```
net = trainNetwork(imdsTrain,layers,options);
```

- 預測數據並計算及顯示每批次的準確度

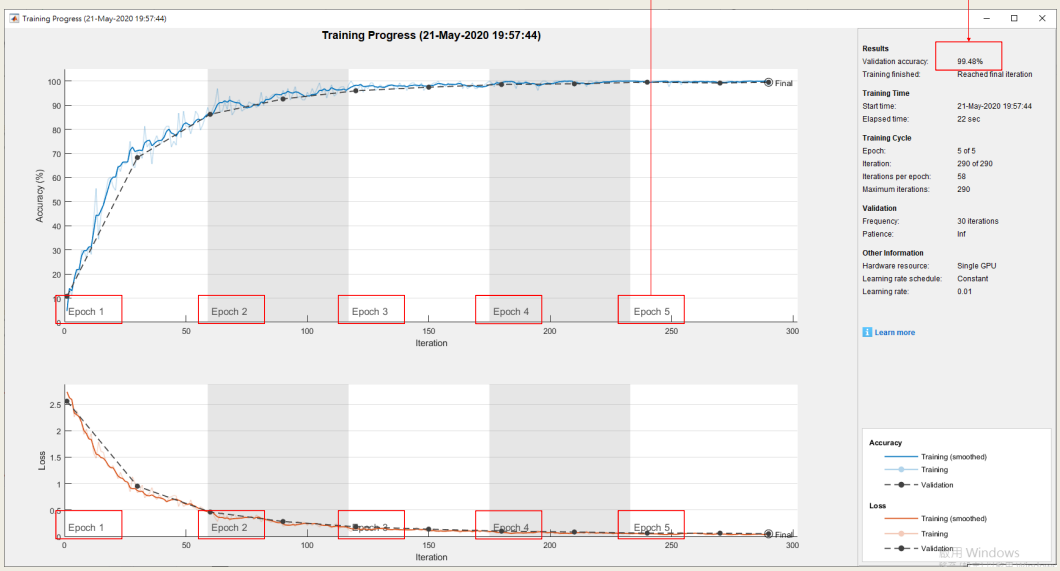
```
YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;

accuracy = sum(YPred == YValidation)/numel(YValidation)
```

```
accuracy = 0.9988
```

結果

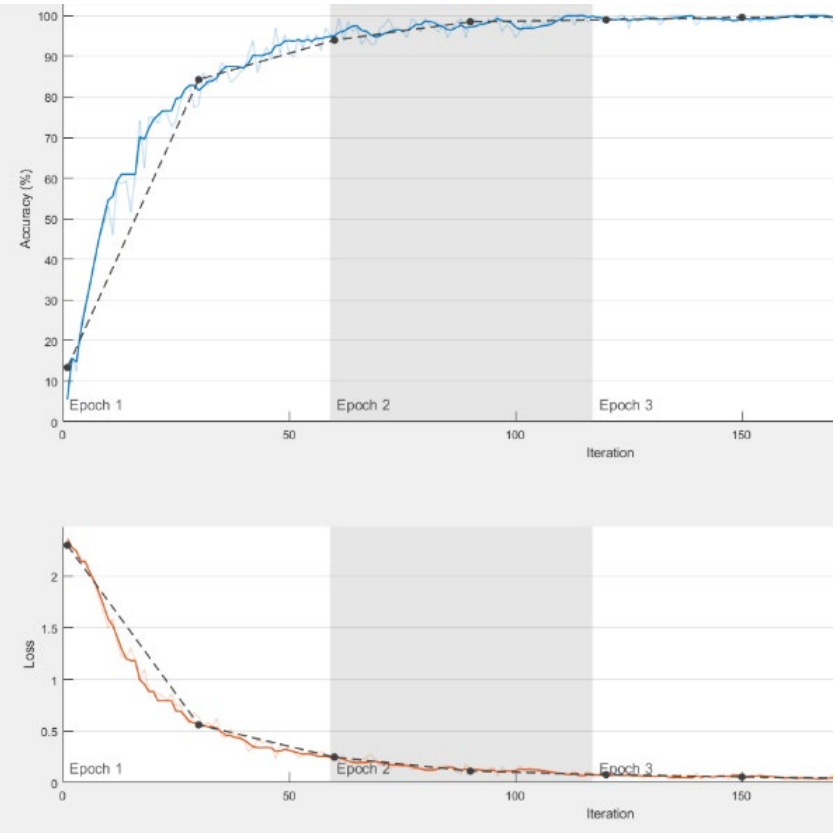
訓練批次

準確度 : 99.48%



請參閱各版本的 GPU 支持。 您還可以使用 trainingOptions 的 “ExecutionEnvironment” 名稱-值對參數指定執行環境。

```
net = trainNetwork(imdsTrain,layers,options);
```



Classify Validation Images and Compute Accuracy

使用訓練好的網絡預測驗證數據的標籤，併計算最終的驗證準確率。 準確性是網絡正確預測的標籤的分數。 在這種情況下，超過 99% 的預測標籤與驗證集的真實標籤匹配。

```
YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;
accuracy = sum(YPred == YValidation)/numel(YValidation)
```

```
accuracy = 0.9988
```

2. Create Simple Deep Learning Network for Classification

載入 MNIST 訓練資料

```
[XTrain,YTrain] = digitTrain4DArrayData;
```

建立卷積神經網路

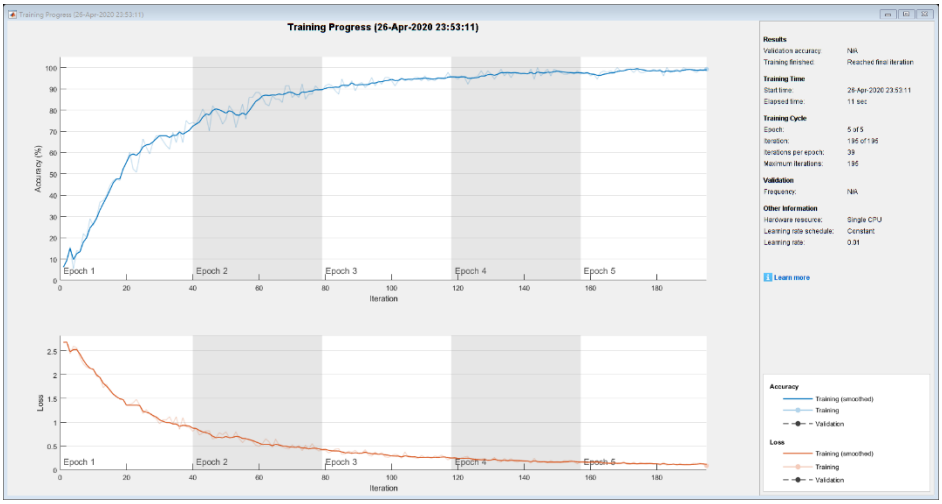
```
layers = [

    imageInputLayer([28 28 1]) %輸入大小為 28*28*1

    convolution2dLayer(3,8,'Padding','same') %卷積核大小為 3*3，其數量為 8，卷積後大小不變
    batchNormalizationLayer %批次正規化，避免過度擬合
    reluLayer
    convolution2dLayer(3,16,'Padding','same','Stride',2)
    batchNormalizationLayer
    reluLayer
    convolution2dLayer(3,32,'Padding','same','Stride',2)
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

設置訓練選項

```
options = trainingOptions('sgdm',... %優化器為具有動量的隨機梯度下降
    'MaxEpochs',5,... %最大訓練回合數
    'Verbose',false,... %是否顯示訓練資訊
    'Plots','training-progress',... %是否畫出訓練過程
    "ExecutionEnvironment","cpu");
```

訓練網路

```
net = trainNetwork(XTrain,YTrain,layers,options);
```



載入 MNIST 測試資料

```
[XTest,YTest] = digitTest4DArrayData;
```

分類 MNIST 測試資料

```
YPredicted = classify(net,XTest);
```

顯示混淆矩陣

```
plotconfusion(YTest,YPredicted)
```