CREATE SIMPLE DEEP LEARNING NETWORK FOR MNIST CLASSIFICATION

- 該範例將告訴你如何創建和訓練透過深度學習進行分類的簡單卷積神經網路。該範例會有以下幾個步驟
  - *載入和瀏覽圖片*
  - *定義神經網路架構*
  - *設定各項訓練參數*
  - *訓練神經網路*
  - *預測數據並計算準確度*

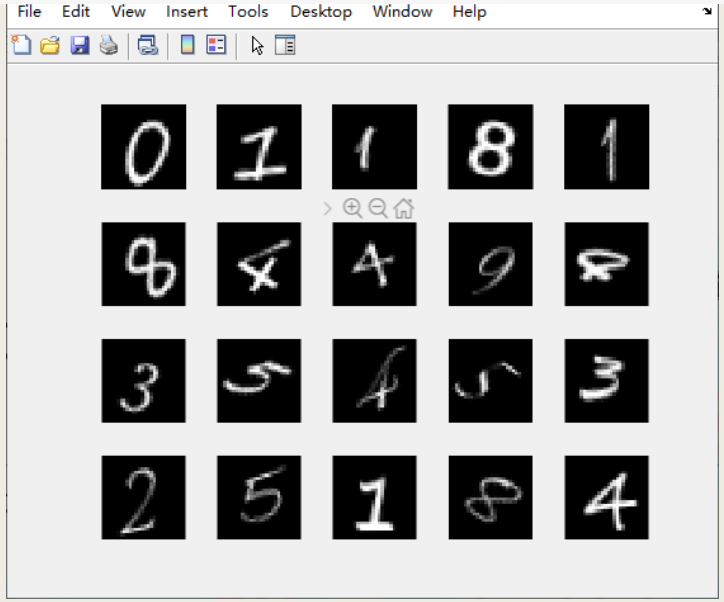
載入和瀏覽圖片

- 從目標資料夾讀取所需資料並將圖片儲存成ImageDatastore object

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

- 顯示部分圖片

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



載入和瀏覽圖片

- 計算每個類別中的圖片數量

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

```
labelCount = 10×2表
    Label計數
    _____ _____

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

指定輸入圖片的大小並顯示圖片尺寸

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

```
ans = 1×2

     28     28
```

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

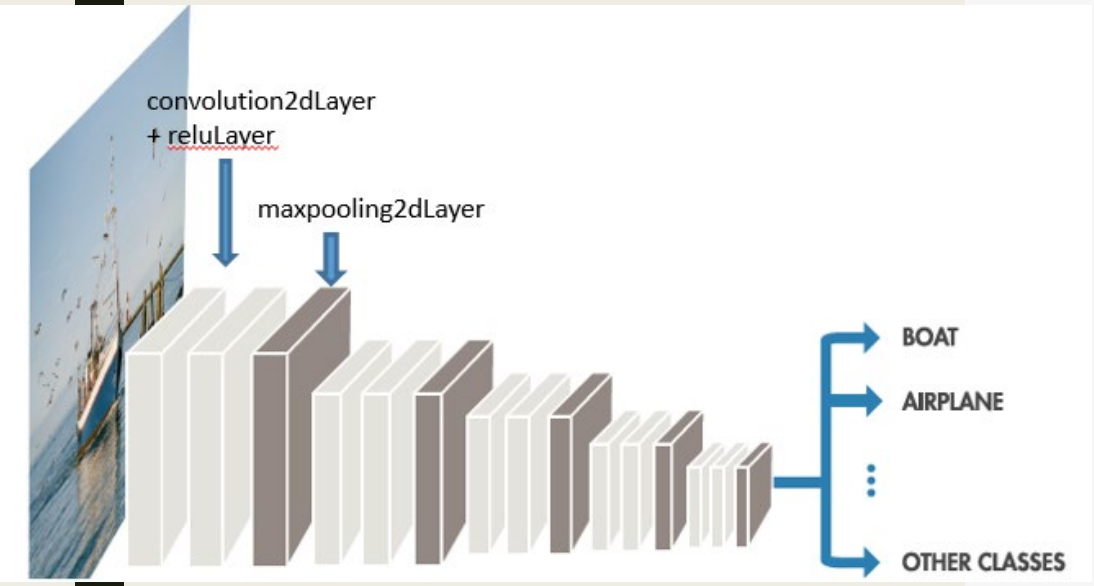
定義神經網路架構

- 將資料分為訓練用資料以及驗證用資料，

  其中每個類別的訓練用資料都包含750張圖片

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

Define deep learning architecture




```
layers = [
    imageInputLayer([28 28 1])
    convolution2dLayer(3,16,'Padding',1)
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(3,32,'Padding',1)
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    convolution2dLayer(3,64,'Padding',1)
    batchNormalizationLayer
    reluLayer
    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

設定各項訓練參數

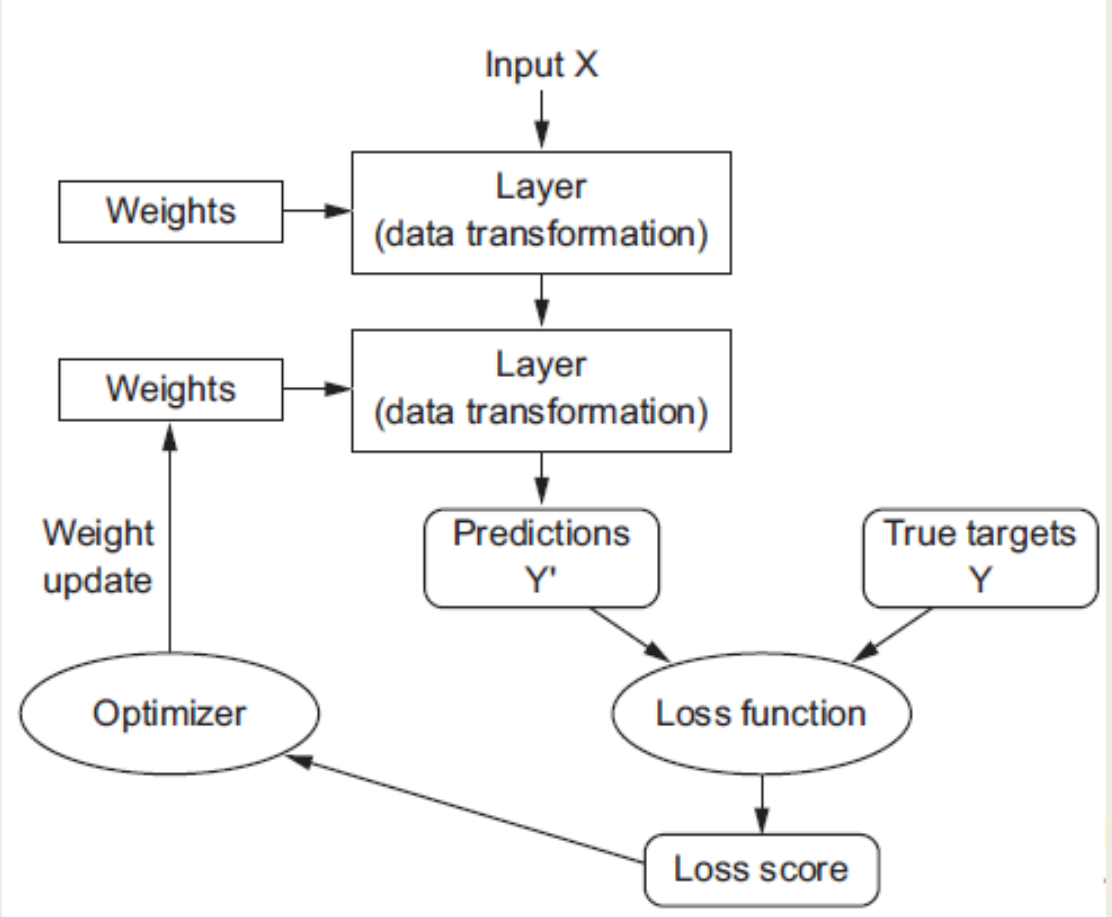
```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

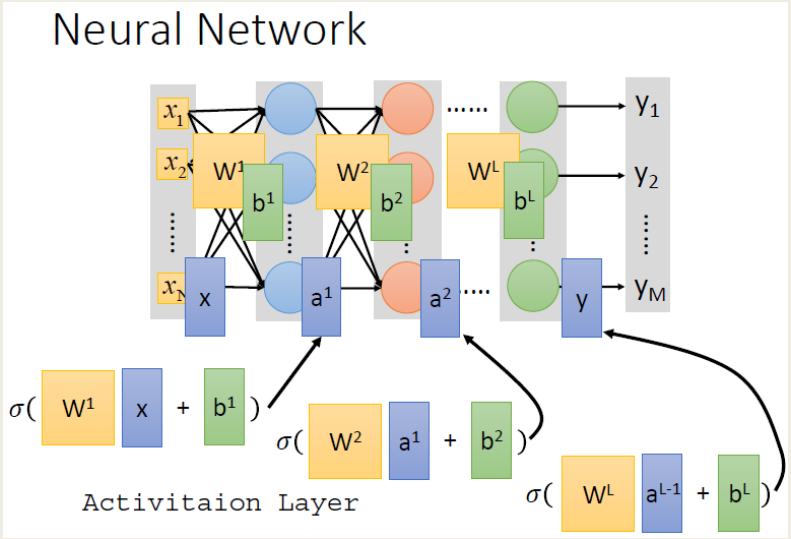
Training Options	Definition	Hint
Plot of training progress	The plot shows the mini-batch loss and accuracy. It includes a stop button that lets you halt network training at any point.	('Plots','training-progress') Plot the progress of the network as it trains.
Max epochs	An epoch is the full pass of the training algorithm over the entire training set.	('MaxEpoch',20) The more epochs specified, the longer the network will train, but the accuracy may improve with each epoch.
Minibatch size	Minibatches are subsets of the training dataset that are processed on the GPU at the same time.	('MiniBatchSize',64) The larger the minibatch, the faster the training, but the maximum size will be determined by the GPU memory. If you get a memory error when training, reduce the minibatch size.
Learning rate	This is a major parameter that controls the speed of training.	A lower learning rate can give a more accurate result, but the network may take longer to train.

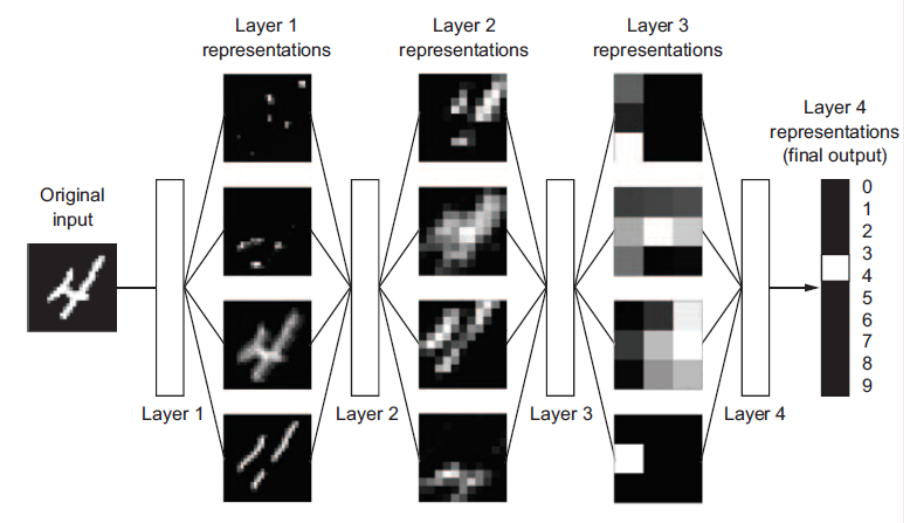
Training process 1. Define the loss function.

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.







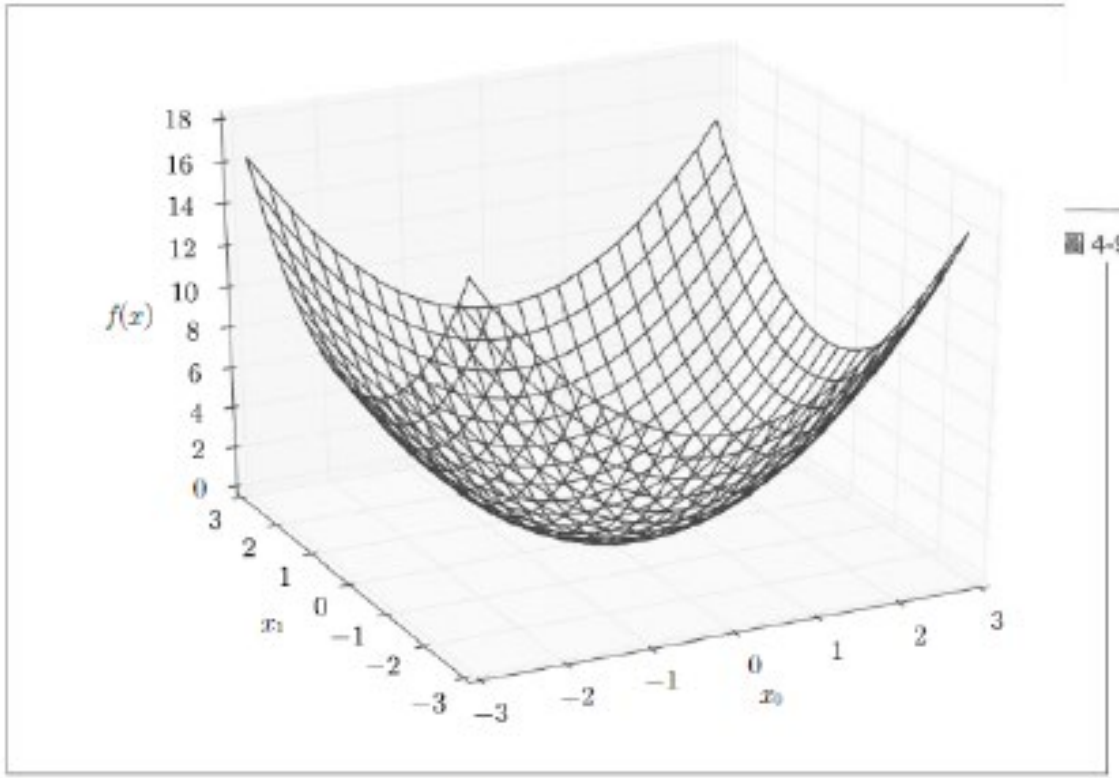


图4-8 $f(x_0, x_1) = x_0^2 + x_1^2$ 的图像

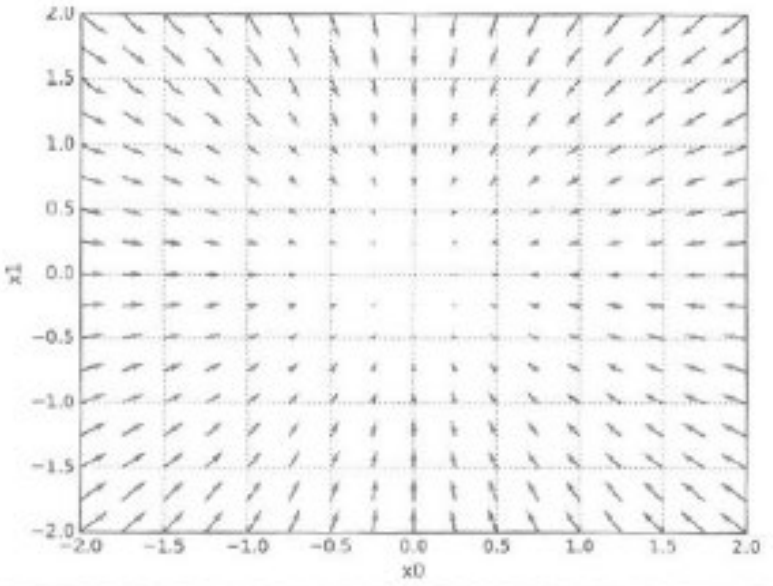
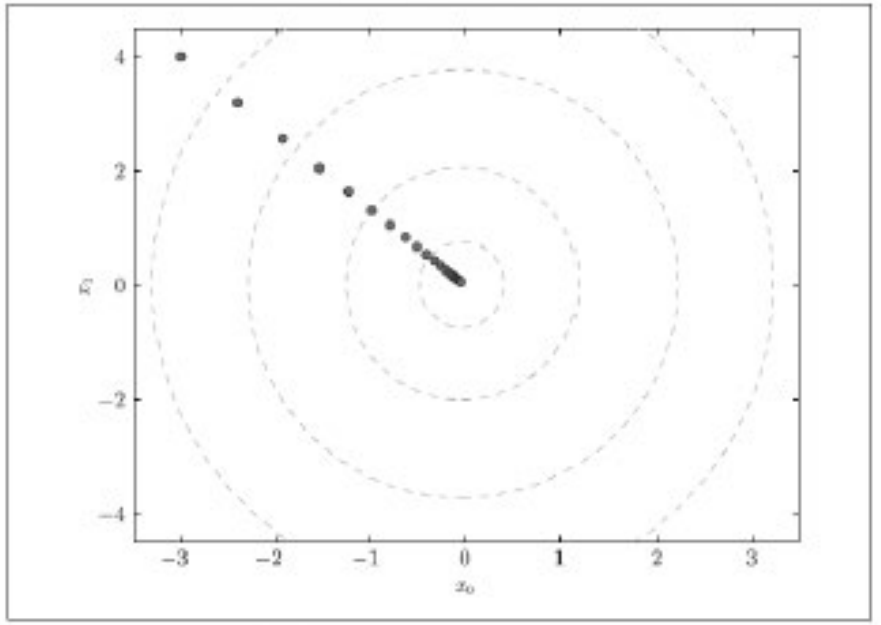


图4-9 $f(x_0$



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%