Day 104 電腦視覺實務延伸

史丹佛線上 ConvNetJS 簡介

陳宇春
出題教練

知識地圖 卷積網路實務延伸

CUPOY

互動式網頁神經網路視覺化

深度神經網路
Supervised LearningDeep Neural Network (DNN)

- 簡介 Introduction
- 套件介紹 Tools: Keras
- 組成概念 Concept
- 訓練技巧 Training Skill
- 應用案例 Application

卷積神經網路
Convolutional Neural Network (CNN)

- 簡介 introduction
- 套件練習 Practice with Keras
- 訓練技巧 Training Skill
- **電腦視覺 Computer Vision**

電腦視覺實務延伸
Computer Vision and CNN

- **互動式網頁 - CNN 視覺化**
- 電腦視覺常用資料集
- 常用的 CNN 模型
- 電腦視覺常見應用

本日知識點目標

- 透過互動式網頁了解卷積網路
- 卷積網路超參數對於訓練與預測率的影響
- 特徵圖 (feature map) 視覺化的效果

何謂 ConvNetJS

CUPOY

- ConvNetJS 是一個 Javascript 庫，用於完全在您的瀏覽器中訓練深度學習模型（神經網路）
- 線上網址：
  - https://cs.stanford.edu/people/karpathy/convnetjs/

- 進入畫面：

圖片來源：cs.stanford

何謂 ConvNetJS

CUPOY

瀏覽器演示

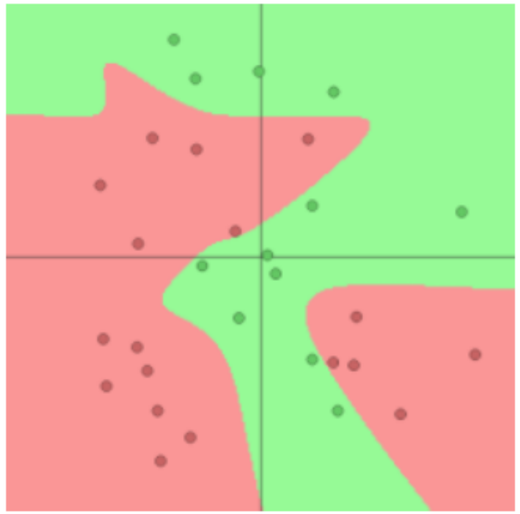
分類MINIST數字與卷積神經網路

利用CNN 做手寫辨識

分類CIFR-10與卷積神經網路

利用CNN 做CIFAR10 影像辨識

使用神經網路對玩具二維數據進行交互式分類



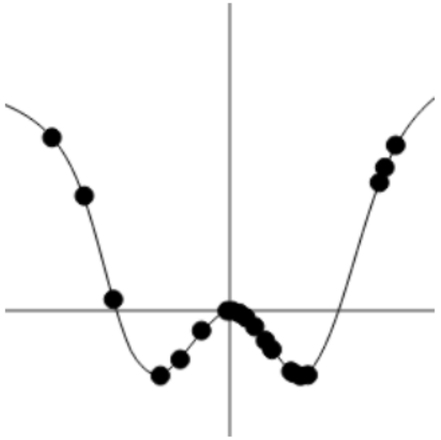
利用NN 做二維分類演示

圖片來源：cs.stanford

何謂 ConvNetJS

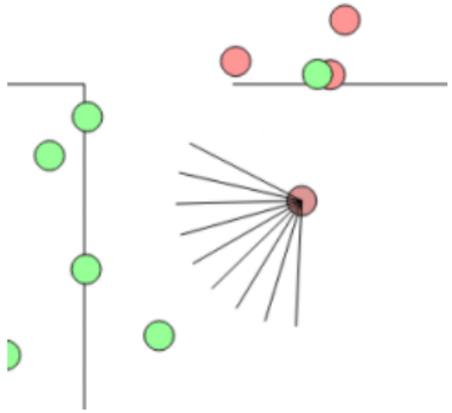
CUPOY

以交互方式回歸玩具1-D數據



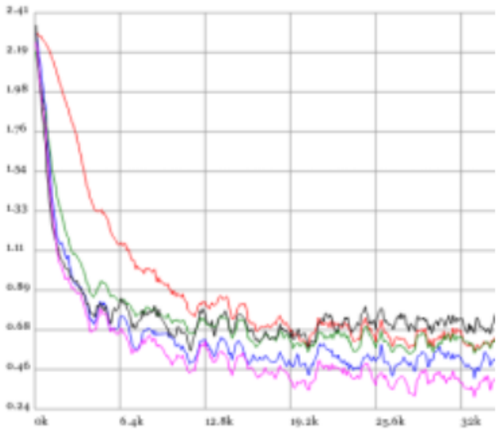
訓練MINST數字自動編碼器

深度學習強化學習



神經網路「繪製」圖像

比較中SGD / Adadelta / Adagrad



圖片來源：cs.stanford

利用 CNN 做 CIFAR10 影像辨識

CUPOY

這個演示在您的瀏覽器中訓練CIFAR-10數據集上的卷積神經網路，只有 Javascript。該數據集的最新技術水平約為 90%，人類表現約為 94%（不完美，因為數據集可能有點模棱兩可）。

這個數據集，數據增強包括隨機翻轉和水平和邏輯上最多 2px 的隨機圖像移位。

在本演示中，使用 Adadelta，它是每個參數自適應步長方法之一，因此我們不必擔心隨著時間的推移而改變學習率或動量。

分類CIFR-10與卷積神經網路

圖片來源：cs.stanford

利用 CNN 做 CIFAR10 影像辨識 – 參數設定

CUPOY

Training Stats

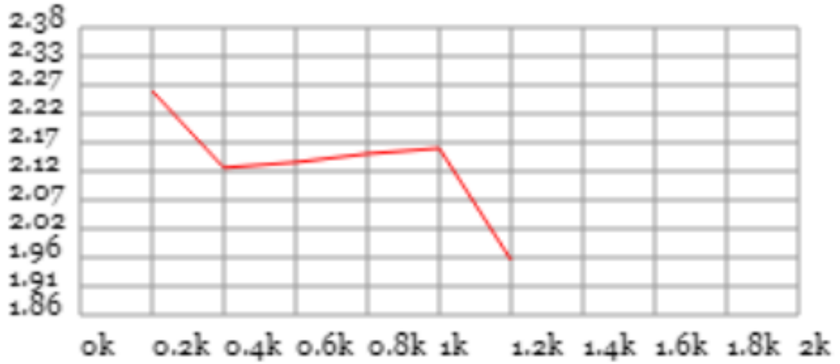
pause

每個batch sample 前行時間 Forward time per example: 8ms
每個batch sample 後行時間 Backprop time per example: 14ms
分類損失函數的值 Classification loss: 1.93805
採用L2 的學習率更新率 L2 Weight decay loss: 0.00115
訓練精確度 Training accuracy: 0.25
驗證精確率 Validation accuracy: 0.16
樣本數 Examples seen: 1300
學習率 Learning rate: 0.01 change
學習動量 Momentum: 0.9 change
批次量大小 Batch size: 4 change
權重衰減率 Weight decay: 0.0001 change

save network snapshot as JSON
init network from JSON snapshot

load a pretrained network (achieves ~80% accuracy)

Loss:



clear graph

圖片來源：cs.stanford

利用CNN 做 CIFAR10 影像辨識 - 開始執行網路訓練(1)

CUPOY

Instantiate a Network and Trainer

```
layer_defs = [];  # 定義網路起始
# 建立DATA 輸入層, 維度: 32x32x3
layer_defs.push({type:'input', out_sx:32, out_sy:32, out_depth:3});
# 建立卷積層1,該層將使用16個內核執行卷積，每個內核大小為5x5。
#移動步數為1,輸入將在所有邊上填充2個像素以使輸出Vol具有相同的大小 , 激活函數為 ReLU
layer_defs.push({type:'conv', sx:5, filters:16, stride:1, pad:2, activation:'relu'});
# 建立池化層1,每個池化內核大小為2x2,移動步數為2
layer_defs.push({type:'pool', sx:2, stride:2});
# 建立卷積層2,該層將使用20個內核執行卷積，每個內核大小為5x5。
#移動步數為1,輸入將在所有邊上填充2個像素以使輸出Vol具有相同的大小 , 激活函數為 ReLU
layer_defs.push({type:'conv', sx:5, filters:20, stride:1, pad:2, activation:'relu'});
# 建立池化層2,每個池化內核大小為2x2,移動步數為2
```

利用CNN 做 CIFAR10 影像辨識 - 開始執行網路訓練(2)

CUPOY

Instantiate a Network and Trainer

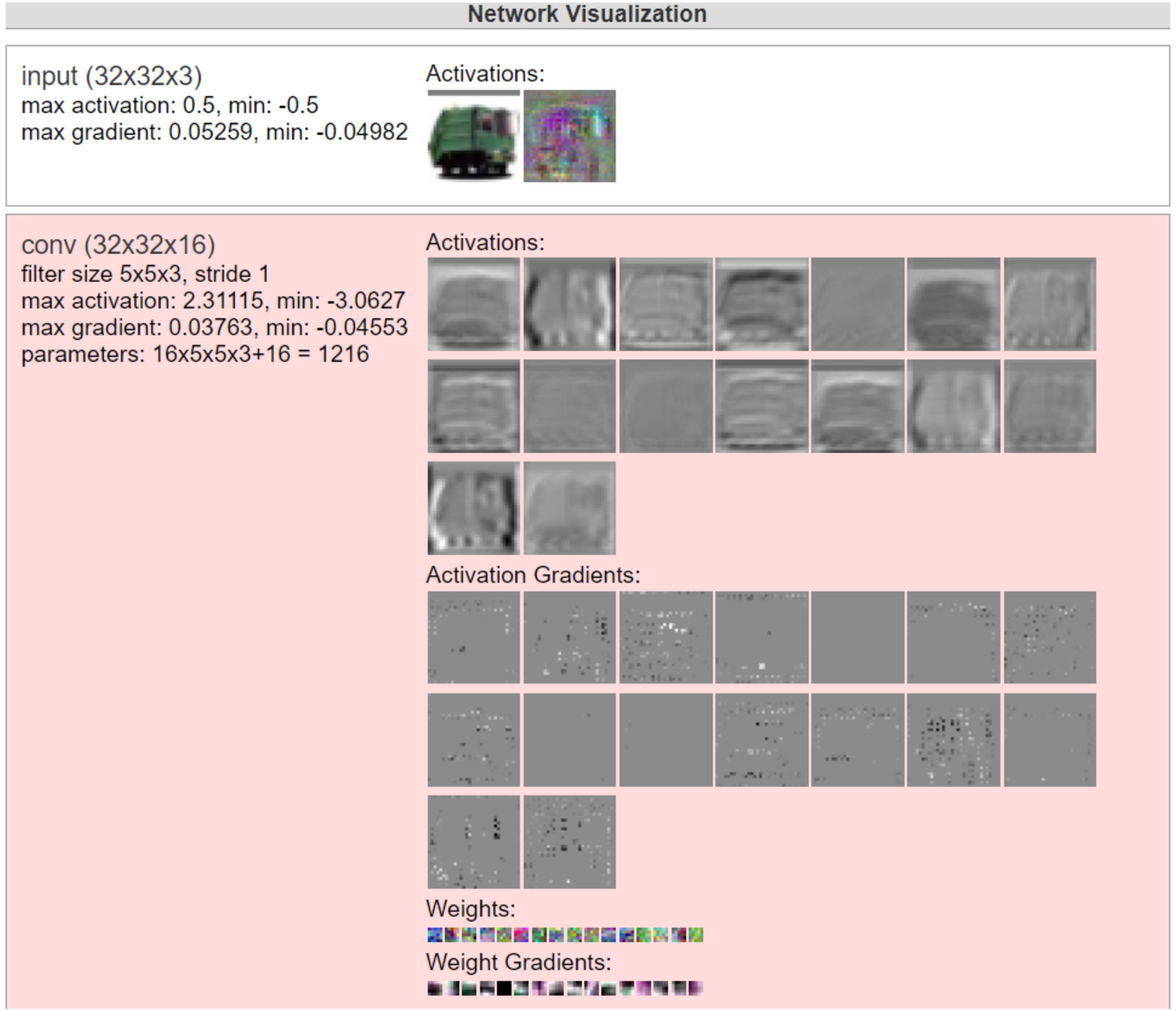
```
layer_defs.push({type:'pool', sx:2, stride:2});
# 建立卷積層3,該層將使用20個內核執行卷積，每個內核大小為5x5。
#移動步數為1,輸入將在所有邊上填充2個像素以使輸出Vol具有相同的大小，啟動函數為 ReLU
layer_defs.push({type:'conv', sx:5, filters:20, stride:1, pad:2, activation:'relu'});
# 建立池化層3,每個池化內核大小為2x2,移動步數為2
layer_defs.push({type:'pool', sx:2, stride:2});
#輸出Vol的大小為1x1x10
layer_defs.push({type:'softmax', num_classes:10});
#指定NET 為一個輸出網路
net = new convnetjs.Net();
#執行並建立網路
net.makeLayers(layer_defs);
#執行網路訓練, 優化器採用adadelta, batch_size=4, l2_decay (l2,每次更新時學習率下降)=0.0001
trainer = new convnetjs.SGDTrainer(net, {method:'adadelta', batch_size:4, l2_decay:0.0001});
```

利用 CNN 做 CIFAR10 影像辨識 – 網路視覺化

CUPOY

第一層啟動後的
視覺呈現

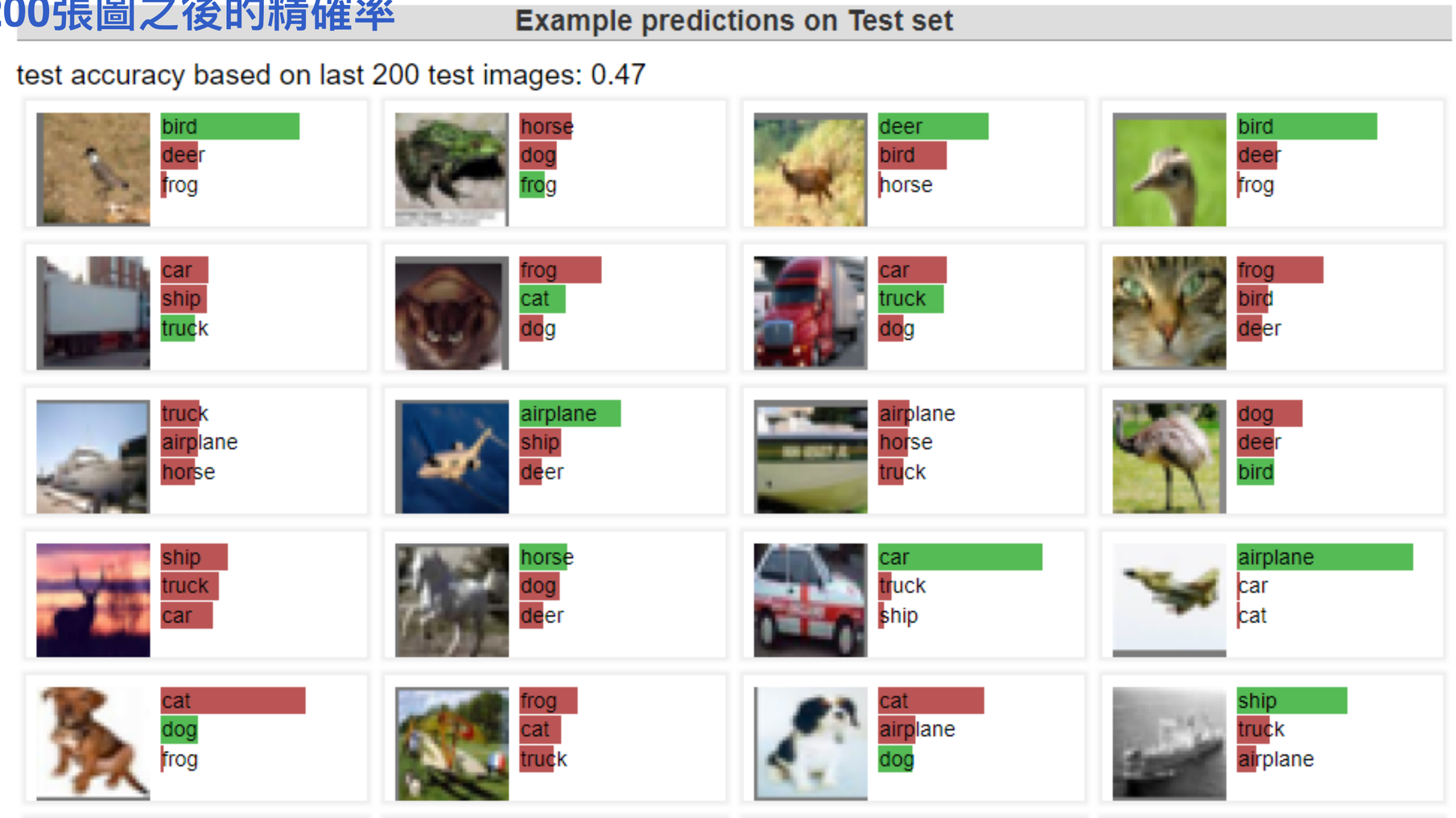
第一層啟動後的特徵圖
視覺呈現



利用 CNN 做 CIFAR10 影像辨識 – 結果預測輸出

CUPOY

測試200張圖之後的精確率



重要知識點複習：相關參數與優化器

CUPOY

- L2_weight decay：在設置上，Weight Decay 是一個 L2 penalty，是對參數取值平方和的懲罰

  (1)取值可以量子化，即存在大量可壓縮空間

  (2)因為 Relu 的存在使得其有界。
- weight decay（權值衰減）的使用最終目的是防止過擬合。所以 weight decay 的作用是調節模型複雜度對損失函數的影響，若 weight decay 很大，則複雜的模型損失函數的值也就大。
- Momentum 是梯度下降法中一種常用的加速技術

延伸閱讀：使用 JSON 保存和加載網路

CUPOY

- **使用 JSON 保存和加載網路**

  簡單地說，使用 toJSON（）和fromJSON（）函數。例如，保存和加載網：

  執行最後一行後，net2 應該與原始網路完全相同

```
# network outputs all of its parameters into json object
# 網路輸出所有相關參數, 放入 json 物件
var json = net.toJSON();

# the entire object is now simply string. You can save this somewhere
# 整個物件現在只是字符串。你可以把它保存在某個地方
var str = JSON.stringify(json);

# 重新建立一個神經網路:
//從字符串中創建json對象(  creates json object out of a string)
var json = JSON.parse(str);
// 創建一個新的網路 (create an empty network)
var net2 = new convnetjs.Net();
// 載入所有參數(load all parameters from JSON)
net2.fromJSON(json);
```

延伸閱讀：使用 JSON 保存和加載網路 - Keras

CUPOY

使用Keras 儲存與加載 網路模型架構

- 保存網路模型但是不包括參數

```
# save as JSON
json_string = model.to_json()

# 使用JSON 建構模型 (model reconstruction from JSON):
from keras.models import model_from_json
model = model_from_json(json_string)
```

延伸閱讀：使用 JSON – 關於優化器

CUPOY

ConvNetJS 網路一樣可以使用優化器：

- ✓ SGD+Momentum
- ✓ Adadelta
- ✓ adagrad.

```
// example SGD+Momentum trainer. Performs a weight update every 10 examples
var trainer = new convnetjs.Trainer(net, {method: 'sgd', learning_rate: 0.01,
                                l2_decay: 0.001, momentum: 0.9, batch_size:
                                10, l1_decay: 0.001});
// example that uses adadelta. Reasonable for beginners.
var trainer = new convnetjs.Trainer(net, {method: 'adadelta', l2_decay: 0.001,
                                batch_size: 10});
// example adagrad.
var trainer = new convnetjs.Trainer(net, {method: 'adagrad', l2_decay: 0.001,
                                l1_decay: 0.001, batch_size: 10});
```