109550017 HW5

Environment details

本次作業皆在Kaggle網站上完成,使用的工具版本如下:

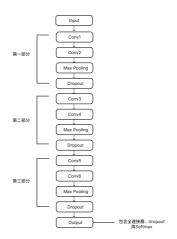
- Python Version- 3.7.12
- torch Version- 1.11.0
- numpy Version- 1.21.6
- cv2 Version- 4.5.4
- tqdm Version- 4.46.0

Implementation details

Model Architecture

模型使用了CNN進行實作,綜觀來看,總共有六層convolution layer,並以每兩層convolution layer為一組,在其後加上max pooling layer與dropout,最後再依輸出數量接上相對應數量的全連接層,每個輸出需經過兩個全連接層。第一層全連接層以ReLU函數activate,ReLU函數會將小於0的值輸出0,大於0的值維持原狀,並經dropout避免overfitting後連接到第二個全連接層,最後輸出。

模型架構如下圖:



CNN由Convolution Layer、Pooling Layer、Activation function組成,並加上BatchNorm做標準化以加速模型的收斂速度。其中透過數層經由ReLU函數activate 的convolution layer擷取特徵,並在第二層、第四層、第六層加上2x2的max pooling layer採樣減少計算量,也用dropout隨機關閉神經元,以避免模型有overfit的 問題。最後用Flatten Layer來把資料降到1維,輸出到全連接層,再透過Softmax regression分類器把輸出層的機率分布總和變為1。

下列為各層與函數介紹:

- 1. Conv:經由ReLU activate的卷積層,用於對圖像掃瞄並開始擷取特徵,會在其中使用kernel當作掃描的工具,並做BatchNorm以標準化,最後輸出結果。
- 2. Max_Pooling:將區域內的最大值作為該區域的代表,減少傳入神經網路的特徵。
- 3. Dropout:在每個部份與全連接層後會接dropout函數,用以避免overfitting,神經元會隨機被關閉,使神經網路在訓練的時候防止特徵之間有合作的關係。隨機關閉神經元後,模型會必須從剩下的不完整網路進行學習,而不是透過全部的神經元。
- 4. 全連接層:全連接層的前層負責進行特徵擷取,而全連接層負責將結果決定出來,藉由將前層學出的所有特徵組合在一起,並進行分類。
- 5. Softmax函數:最後全連接層的輸出接到softmax上,使輸出的機率分布能夠逼近真實的機率分布,將總和變為1。

首先,第一層Convolution Layer的輸入的維度為1*32*32,1為通道數,32為resize過的照片長和寬,stride設為1,padding也設為1,避免靠近圖片邊緣的部分受到影響,kernel_size設為3,輸出則使用了64個通道。第二層則不會增加通道數量,輸入與輸出通道數均與第一層的輸出數量相同,其餘stride、padding、kernel_size的部分均與第一層相同。

其餘二部分均重複上述行為,第三層與第五層行為類似第一層,第二層與第四層行為類似第二層,不同的地方在於每經過2層convolution layer與後接的max pooling layer與dropout後,通道數會在經過每層的時候會被乘以2,而第二與第三維度減半。

Task實作

在三種task上各使用了三個模型,主要差異在於全連接層數量會依照task規定的驗證碼數量而增加。

Data augmentation

將原先訓練資料變為4倍,除了原先的訓練資料外,分別對資料進行水平翻轉、改變色彩(去除藍色)、增加雜訊。

Hyperparameters

model weight link

除了上附連結含的模型weight(通道數、kernel_size、stride、padding)外,我還調整了data loader的batch數量,batch數量為每次由訓練目標中取樣的大小,大的batch數量較利於模型收斂,但也會影響模型泛化的能力,小的batch數量則相反。經過數次試驗後,我發現將batch數量與訓練資量大小的比例控制在0.03時效果最好。epoch數設為100,learning rate為0.001。

Loss function and optimizer

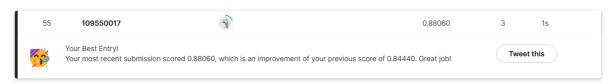
loss function使用了cross entropy計算預測與正確結果的差異,optimizer在試驗過後仍決定使用Adam,Adam的表現較為穩定,準確率也更高。

Used Deep Learning Framework

使用pytorch進行建構

Output

leader board截圖:



Reference

- $\bullet \ \ \, \underline{\text{https://adventuresinmachinelearning.com/convolutional-neural-networks-tutorial-in-pytorch/}\\$
- https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax
- https://ithelp.ithome.com.tw/articles/10187521
- $\bullet \ \ \, \underline{https://github.com/JasonLiTW/simple-railway-captcha-solver}$
- $\bullet \ \ \, \underline{\text{https://www.cupoy.com/qa/club/ai}} \ \, \underline{\text{tw/0000016D6BA22D97000000016375706F795F72656C656173654B5741535354434C5542/000001752A2637F20000C}} \\$
- https://github.com/SIDD58/Captcha-recognition-using-CNN
- https://github.com/chengche6230/Pattern-Recognition-Course/blob/main/Homework5/0613414_HW5.py
- https://github.com/ice-tong/pytorch-captcha
- https://medium.com/手寫筆記/使用-tensorflow-學習-softmax-negression-41a12b619f04
- https://hackmd.io/@allen108108/H1l4zqtp4#Adam-—-常用的-optimizer
- https://chih-sheng-huang821.medium.com/03-pytorch-dataaug-a712a7a7f55e

109550017_HW5 2