1. Convolutional Network
2. forward
   * 首先對圖片x做padding，在4周都加入paading個0。
   * 由於產生的圖片大小會根據padding、filter及stride而有不同，在程式中H\_out及W\_out就是產生圖片的長及寬。F代表新產生圖片的深度，也就是幾個不同的feature。N則是mini batch的大小。
   * 這邊我們會依序做迴圈來計算每個新圖片像素點的值，x\_tmp代表目前圖片的區塊會被filter產生新的像素點上的值。計算方式是將x\_tmp與filter做element wise multiplication，並將其加總，最後再加上一個bias。

Text

Description automatically generated

1. Backward
   * dx : 由於新圖片的每個像素點都是好幾個舊圖片像素點所產生，且每個就像素點都可能產生好幾個像素點，因此其微分的計算方式會是每個產生的心像素的dout[I,j,k,l]去乘以對應的filter[j, :]，並將其加回原本就圖片的位置，其中j代表的是共有幾個不同的filter，因此F層個個filter都會加回原本的位置。最後，由於圖片是有加過padding的，所以會將padding去除。
   * dw : 由於每個filter都會在整張圖片移動，因此其微分會將dout與其所到x處相乘並加總，其計算方式如程式碼所示。
   * db : wx+b對b微分為1，同時共有F個filter所以每個filter對應的db就是將dout除了F維度以外加總。

Text

Description automatically generated

1. Apply filter
   * A screenshot of a computer

     Description automatically generated with medium confidence使用不同的filter，並執行forward可以對圖片做特定的效果

A picture containing text, dog, indoor, staring

Description automatically generated

1. Max Pooling
2. forward
   * Text

     Description automatically generatedmax pooling就是將其所指定的範圍取出最大值當成新圖片的像素點，因此其做法就是在圖的height k與width l中每次跨stride的跨度來取出當前的正方形或長方形，並將裡面最大的值設為output的像素
3. backward

* Text

  Description automatically generateddx : m由於max pooling只取最大值，因此在微分時只有那個最大值是1其他都是0，所以在backward的dx就是最大值的部分保留其他都是0。

1. Three Layer Convolution Layer
2. Init

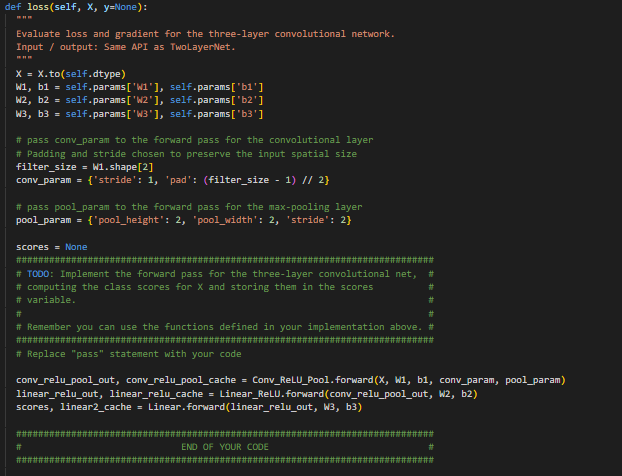
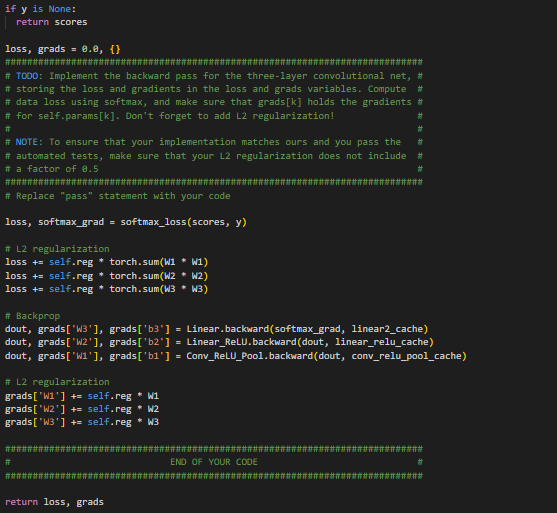
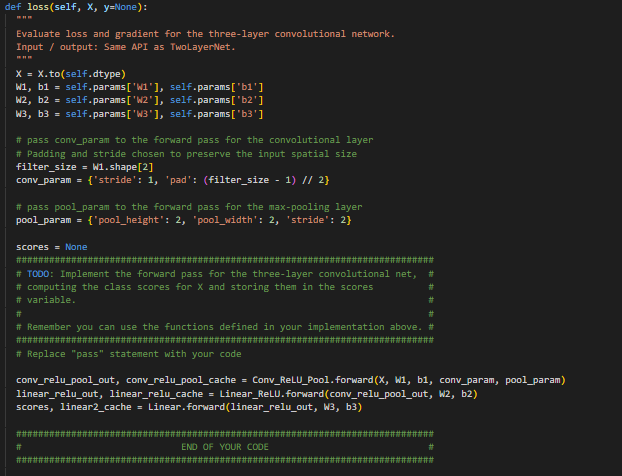
* conv - relu - 2x2 max pool - linear - relu - linear - softmax
* layer 1為size為conv -> relu -> 2x2 max pool，其參數weight會有size為 (F,C,HH,WW ); 參數b為 size F 。(F,)代表有多少個filter、C代表原本input image的深度，HH及WW則代表filter的長及寬。
* layer 2為linear layer，linear - >relu，其參數weight為size(num\_filters \* HH \* WW, hidden\_dim)，其中num\_filters \* HH \* WW代表將前面filter轉為1維來執行linear layer，hidden dimension則是此layer輸出的神經元; 參數b的size則是根據output hidden dimension。
* layer 3 為 linear – softmax，其參數weight為size(hidden\_dim, num\_classes)，hidden dimension是前一個linear layer的output，num\_classes則是output size，在這邊代表幾種分類; 參數b的size則是根據output num\_classes。
* 在weight的部分透過torch.randn產生隨機的值並且乘以一個weight\_scale來做擴張。Bias的部分則是初始化為0。

Text

Description automatically generated

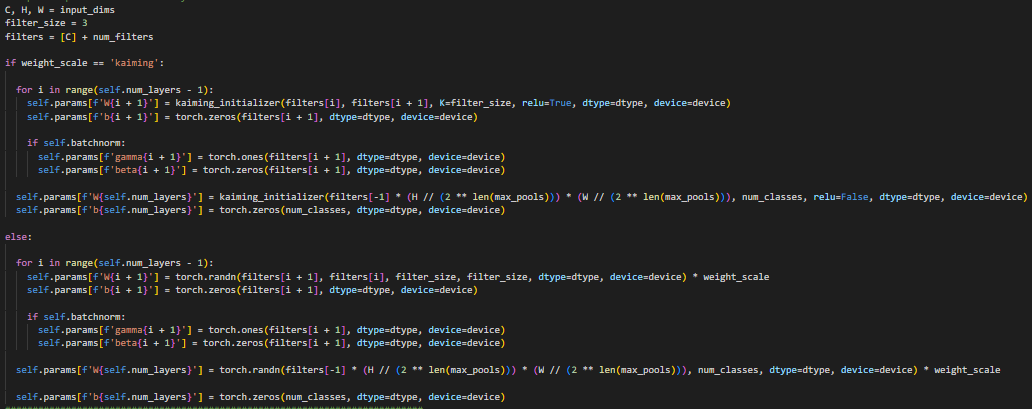
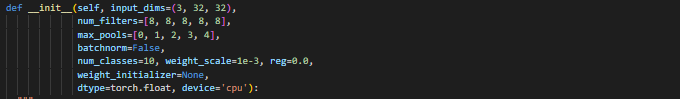
1. Loss

* 透過Loss function，可以對照片執行3 layer convolution network，其會依序執行conv -> relu -> 2x2 max pool -> linear -> relu -> linear -> softmax，最後透過cross entorpy找出其loss。
* 透過使用L2 Regularization來降低network overfit training data的情況，在implentation中我們忽略乘以0.5的部分。
* Back propagation的部分，我們需要由第3 layer 依序做back prop到 layer 1，並計算每層的gradient，其中gradient必須依照L2 regularization進行調整( )



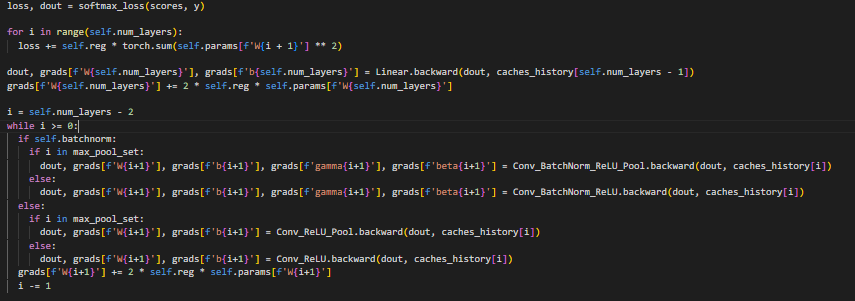
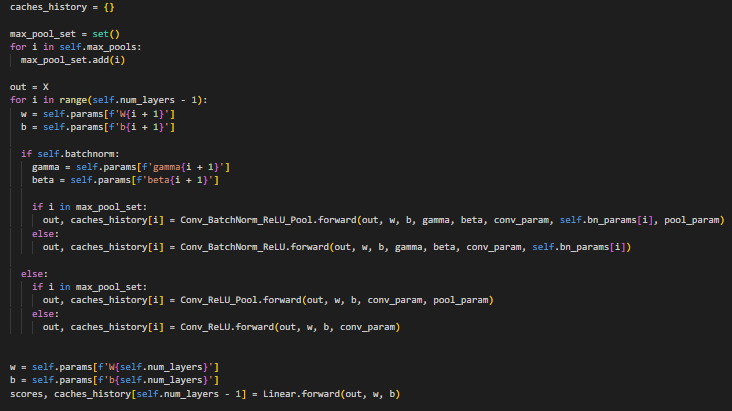
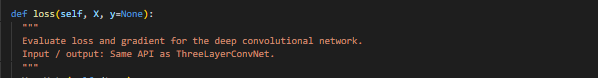
1. DeepConvNet
2. Init

* {conv - [batchnorm?] - relu - [pool?]} \* (L - 1) - linear，此Deep Convolution Network每層都是 Convolution-> batch normalization -> Lelu -> pooling結合起來，其中batch normalization及pooling都是optional的。透過多層再加上一個linear layer形成整個network。
* Filter size固定為3、pooling 固定為2\*2。
* Init這個部分會去初始化要記住的參數，這個部分的參數有分成Weight、Bias、gamma、beta。其中weight的size為(output filters, input filters, filter size, filter size); Bias的size為(output filters, ) ; gamma 與 beta都是在batch normalization才會使用，其size都是(output filters, ) 並且分別初始化為1及0。
* 在最後一層linear layer ，weight 的size為(input dimension, output classes); bias的size為(input diemension, )，其中input dimension會受到使用了多少次的max pool所影響，每使用一次圖片的長寬就少1倍，因此計算方式請參考程式。



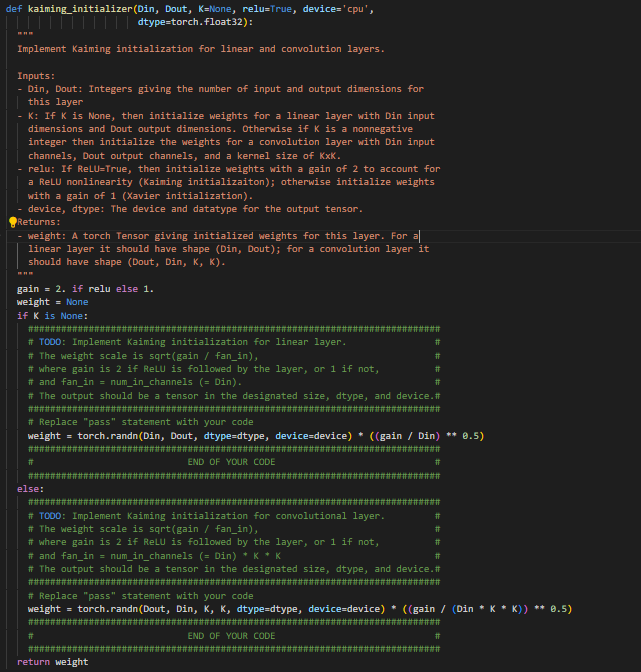
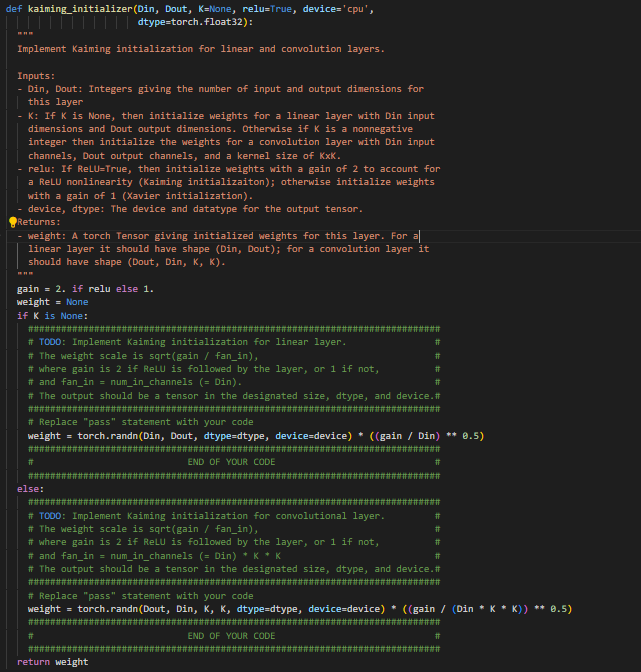
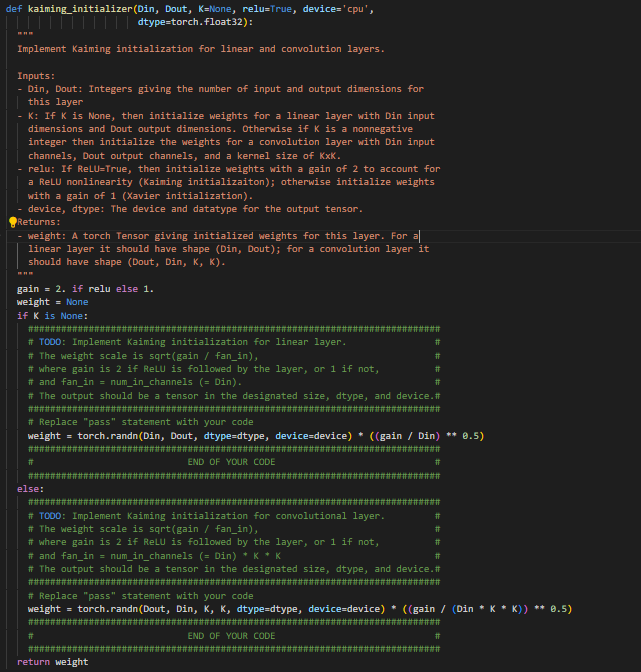
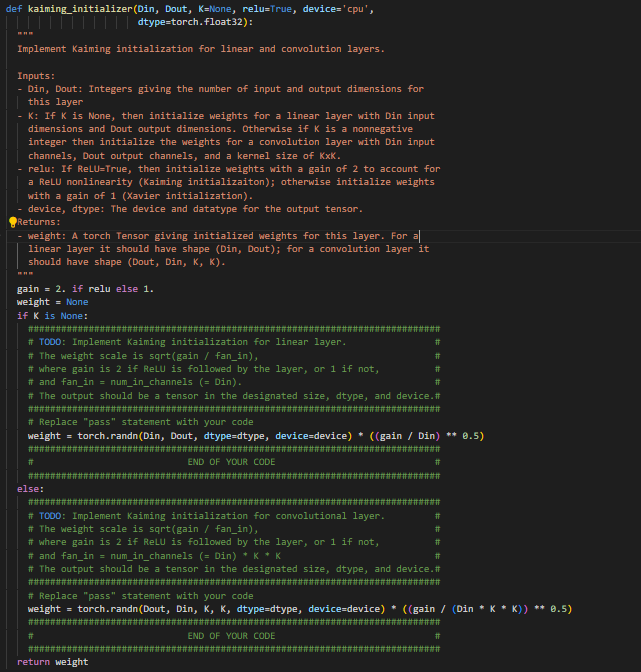
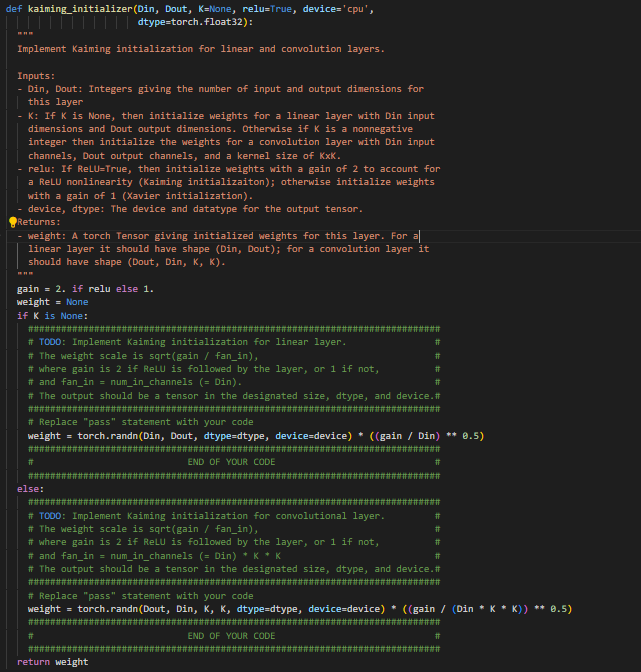
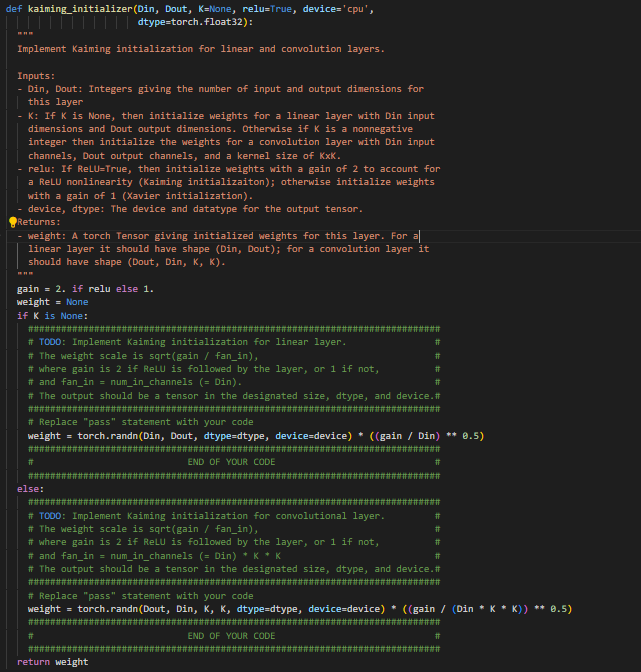
1. loss

* 依據選擇的情況去執行不同的layer，並且將out 不斷地只配給下個layer，並且每個layer都使用屬於該layer的parameter ，例如self.params[W{layer num}]。
* 最後一輪forward則是使用linear.forward並且輸出每個class的score
* 在backward的部分則是反向執行，先從linear layer開始，依序往前執行定義的conv layer直到layer 1，在執行的過程中必須記住其gradient。
* 在一開始使用softmax cross entropy算出loss後，會將此loss做L2 regularization，也就是加上所有weight^2 \* regularization factor。在最後面也會將所有的gradient做L2 regularization，也就是加上2 \* regularization factor \* weight。



1. kaiming\_initializer

* 透過kaiming方法來初始化weight可以有效提升訓練成效。
* 如果沒有K的話代表是執行linear layer的初始化。
* 有K的話代表是convolution layer的出初始化。



1. create\_convolutional\_solver\_instance

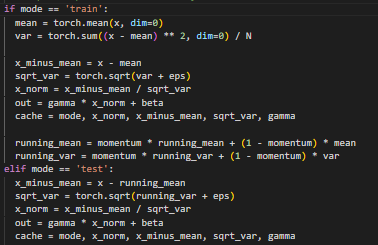
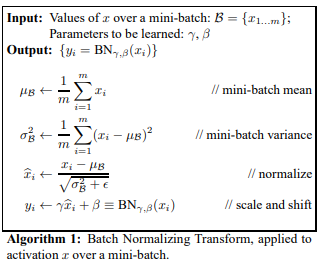
* Text

  Description automatically generatedText

  Description automatically generated建立模型

1. Batch Normalization
2. 為了使neuro network更好的訓練及產生更好的結果，除了可以使用不同的optimization方法，還可以使用batch normalization。主要原因是因為在訓練時，如果input data包含uncorrected features會使得成效較佳。Batch normalization透過預處理資料來decorrelate features來幫助training。
3. forward

* 在training的部分根據論文裡面的公式來計算，將所有的x normalize。running\_mean及running\_var則是應用momentum算出的mean及variance。
* 在testing的部分，則是直接使用running\_mean及running\_var來當作mean及variance來計算。



1. Backward

* <https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html>
* dx的計算方式參考上方連結
* dbeta : 對beta微分為1，所以計算方式就是dout加總。
* dgamma : 對gamma微分為x\_norm，所以計算方式為dout \* x\_norm。

Text

Description automatically generated

1. backward\_alt

* <https://kevinzakka.github.io/2016/09/14/batch_normalization/>
* A screenshot of a computer

  Description automatically generated with medium confidencedbeta及dgamma跟一般backward一樣，dx計算方式請參考上方連結

1. Spatial Batch Normalization
2. Batch Normalization只能用在Linear neuro network，為了使其可以使用在Convolution neuro network，我們必須將其input調整成可以dimension 2，並且是根據Channel去做normalization。
3. Forward

* Text

  Description automatically generated首先會將(N, C, H, W)調整成(N, H, W, C)，在將其轉變為(N\*H\*W, C)，這樣在執行Batch normalization時就會以C為基準去normalize其他dimension。在計算結束後，我們必須將Output轉回原本的模式。

1. Backward

* Text

  Description automatically generated與forward相同，只是變成呼叫backward function來執行

1. Batch Normalization Performance
2. 從途中可以看到在使用batch normalization時，其所產生的準確度明顯比沒有使用的高。

Graphical user interface, chart

Description automatically generated