**Cross Entropy**

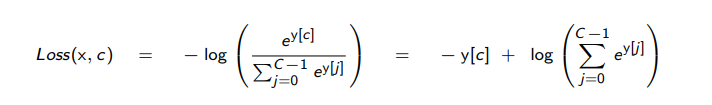
The cross-entropy value shown in Eq. (??) on the previous slide is what is measured as the cross entropy loss by a callable instance of the PyTorch class torch.nn.CrossEntropyLoss that you can access through the link:

<https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>

As the documentation page says, the torch.nn.CrossEntropyLoss function expects to see at output of the final layer of your neural network an unnormalized scores for each class. What that translates into is that in your own code you do not have to worry about translating the output in the final layer of your neural network into numbers that look like probability masses; the torch.nn.CrossEntropyLoss class will take care of that for you.

IMPORTANT: torch.nn.CrossEntropyLoss expects that if C is the total number of classes in your training data, your class labels are integers between 0 and C − 1.

Given the values y[i], i = 0, . . . , C − 1, at the nodes of the output layer, for each image x in the input batch, CrossEntropyLoss calculates



where y is the tensor that represents the values in the output layer of the neural network and c is the index value of the true class for the input image x.

torch.nn.CrossEntropyLoss returns the average of the loss values for all the images in a batch and that’s what is backproped when you call .backward() on it.

A nice thing about the torch.nn.CrossEntropyLoss is that it lets you weight the loss calculations to deal with what is referred to as the class imbalance problem in your training data.

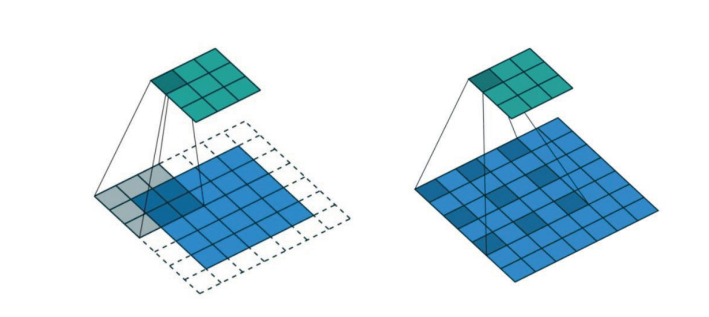
One more thing about torch.nn.CrossEntropyLoss: This criterion combines the torch.nn.LogSoftmax activation function and the torch.nn.NLLLoss loss function in one single class. The LogSoftmax activation function calculates the log-ratio for every node index i in the output layer of the neural network. Subsequently, the NLLLoss loss function returns the negative of one of these values that corresponds to the true label of the input image. The name NLLLoss stands for “Negative Log Likelihood Loss”.

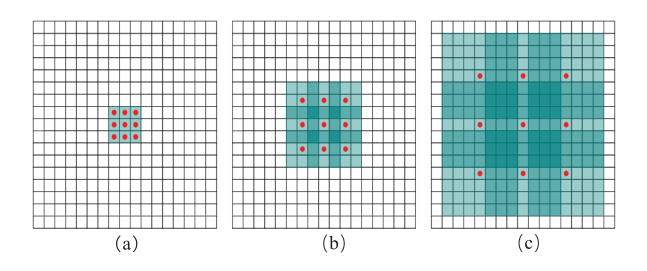
Most neural networks for image classification consist of convolutional layers followed by a small number of fully connected (FC) layers. The number of nodes in the last FC layer equals the number of image classes in your training data. The comment made above implies that when you use the torch.nn.CrossEntropyLoss loss criterion, you do NOT need an activation function for the final layer since the LogSoftmax activation is built into the loss calculation.

**Dilation**

The dilated convolution algorithm, which is widely used for image segmentation, is applied in the image classification field in this paper. In many traditional image classification algorithms, convolution neural network (CNN) plays an important role. However, the classical CNN has the problem of consuming too much computing resources. To solve this problem, first, this paper proposed a dilated CNN model which is built through replacing the convolution kernels of traditional CNN by the dilated convolution kernels, and then, the dilated CNN model is tested on the Mnist handwritten digital recognition data set. Second, to solve the detail loss problem in the dilated CNN model, the hybrid dilated CNN (HDC) is built by stacking dilated convolution kernels with different dilation rates, and then the HDC model is tested on the wide-band remote sensing image data set of earth’s terrain. The results show that under the same environment, compared with the traditional CNN model, the dilated CNN model reduces the training time by 12.99% and improves the training accuracy by 2.86% averagely, compared with the dilated CNN model, the HDC model reduces the training time by 2.02% and improves the training and testing accuracy by 14.15% and 15.35% averagely. Therefore, the dilated CNN and HDC model proposed in this paper can significantly improve the image classification performance.

To cope with more complex situations and achieve higher network accuracy, the depth of CNN is increased by stacking layers in traditional methods. However, the disadvantages are also very prominent. The back propagation of gradient may lead to the vanishing gradient with the increasing of network layers, then the network performance tends to be





saturated or even drops sharply. The dilated CNN model is a solution to the above problems.

The dilated CNN model is formed by using the dilated convolution kernels to instead the traditional convolution kernels. The dilated convolution kernel is shown in the Fig.6, where (a) is a traditional convolution kernel of size 3 ∗ 3, a hole (a point with weight of 0) is inserted around each point in the matrix in (a) and transform into (b), similarly, (c) is a 3-hole dilated convolution kernel. As Fig.6 shows, the receptive field of the convolution kernel in (a) is 3 ∗ 3, in (b) is 7 ∗ 7, and in (c) is 15∗15. The size of the receptive field increases with the increase of the number of inserted hole, however, the number of parameters in (a), (b) and (c) are still the same. Therefore, using such a dilated convolution kernel to process images can make the convolution kernel obtain more information without increasing the computation.

