Q.1) How to tune batch size and number of epochs?

Ans: Batch\_size determines the number of samples in each mini batch. Its maximum is the number of all samples, which makes gradient descent accurate, the loss will decrease towards the minimum if the learning rate is small enough, but iterations are slower. Its minimum is 1, resulting in stochastic gradient descent: Fast but the direction of the gradient step is based only on one example, the loss may jump around. batch\_size allows to adjust between the two extremes: accurate gradient direction and fast iteration. Also, the maximum value for batch\_size may be limited if your model + data set does not fit into the available (GPU) memory.

steps\_per\_epoch the number of batch iterations before a training epoch is considered finished. If you have a training set of fixed size you can ignore it but it may be useful if you have a huge data set or if you are generating random data augmentations on the fly, i.e. if your training set has a (generated) infinite size.

Q.3) Illustrate the working of Gradient descent.

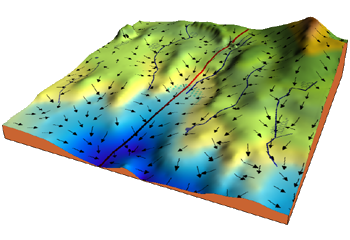
Ans: It is an *iterative* optimization algorithm used in machine learning to find the best results (minima of a curve).

*Gradient* means the *rate* of inclination or declination of a slope.

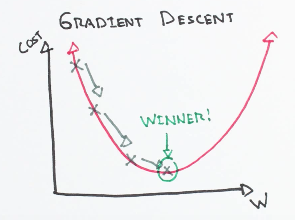
*Descent* means the instance of *descending*.

Working:

Consider the 3-dimensional graph below in the context of a cost function. Our goal is to move from the mountain in the top right corner (high cost) to the dark blue sea in the bottom left (low cost). The arrows represent the direction of steepest descent (negative gradient) from any given point–the direction that decreases the cost function as quickly as possible.



Starting at the top of the mountain, we take our first step downhill in the direction specified by the negative gradient. Next we recalculate the negative gradient (passing in the coordinates of our new point) and take another step in the direction it specifies. We continue this process iteratively until we get to the bottom of our graph, or to a point where we can no longer move downhill–a local minimum.



The size of these steps is called the *learning rate*. With a high learning rate we can cover more ground each step, but we risk overshooting the lowest point since the slope of the hill is constantly changing. With a very low learning rate, we can confidently move in the direction of the negative gradient since we are recalculating it so frequently. A low learning rate is more precise, but calculating the gradient is time-consuming, so it will take us a very long time to get to the bottom.

Now let’s run gradient descent using our new cost function. There are two parameters in our cost function we can control: mm (weight) and bb (bias). Since we need to consider the impact each one has on the final prediction, we need to use partial derivatives. We calculate the partial derivatives of the cost function with respect to each parameter and store the results in a gradient.

Math

Given the cost function:

f(m,b)=1N∑i=1n(yi−(mxi+b))2f(m,b)=1N∑i=1n(yi−(mxi+b))2

The gradient can be calculated as:

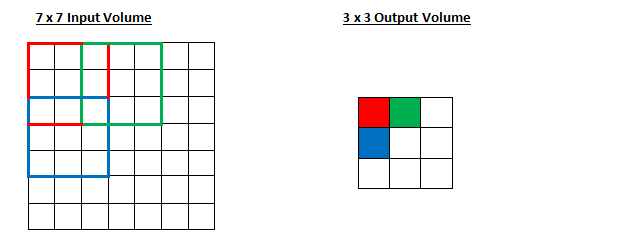
f′(m,b)=[dfdm/dfdb]=[1N∑−2xi(yi−(mxi+b))1N∑−2(yi−(mxi+b))]f′(m,b)=[dfdmdfdb]=[1N∑−2xi(yi−(mxi+b))1N∑−2(yi−(mxi+b))]

To solve for the gradient, we iterate through our data points using our new mm and bb values and compute the partial derivatives. This new gradient tells us the slope of our cost function at our current position (current parameter values) and the direction we should move to update our parameters. The size of our update is controlled by the learning rate.

Q.4) Explain stride and padding in convolution neural network.

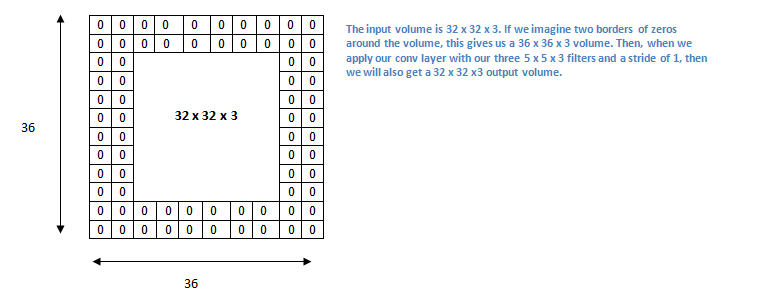
Ans: Now, there are 2 main parameters that we can change to modify the behavior of each layer. After we choose the filter size, we also have to choose the **stride** and the **padding**.

Stride controls how the filter convolves around the input volume. In the example we had in part 1, the filter convolves around the input volume by shifting one unit at a time. The amount by which the filter shifts is the stride. In that case, the stride was implicitly set at 1. Stride is normally set in a way so that the output volume is an integer and not a fraction. Let’s look at an example. Let’s imagine a 7 x 7 input volume, a 3 x 3 filter (Disregard the 3rd dimension for simplicity), and a stride of 1. This is the case that we’re accustomed to.



So, as we can see, the receptive field is shifting by 2 units now and the output volume shrinks as well. Notice that if we tried to set our stride to 3, then we’d have issues with spacing and making sure the receptive fields fit on the input volume. Normally, programmers will increase the stride if they want receptive fields to overlap less and if they want smaller spatial dimensions.

Now, let’s take a look at padding. Before getting into that, let’s think about a scenario. What happens when you apply three 5 x 5 x 3 filters to a 32 x 32 x 3 input volume? The output volume would be 28 x 28 x 3. Notice that the spatial dimensions decrease. As we keep applying conv layers, the size of the volume will decrease faster than we would like. In the early layers of our network, we want to preserve as much information about the original input volume so that we can extract those low level features. Let’s say we want to apply the same conv layer but we want the output volume to remain 32 x 32 x 3. To do this, we can apply a zero padding of size 2 to that layer. Zero padding pads the input volume with zeros around the border. If we think about a zero padding of two, then this would result in a 36 x 36 x 3 input volume.



If you have a stride of 1 and if you set the size of zero padding to

https://adeshpande3.github.io/assets/ZeroPad.png

where K is the filter size, then the input and output volume will always have the same spatial dimensions.

The formula for calculating the output size for any given conv layer is

https://adeshpande3.github.io/assets/Output.png

where O is the output height/length, W is the input height/length, K is the filter size, P is the padding, and S is the stride.