1)After each stride 2 conv,why do we double the number of filters ?

Ans :

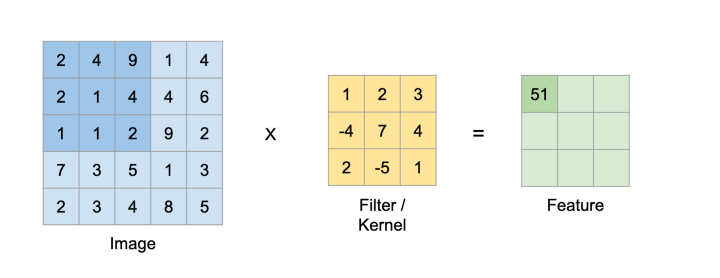
When we use a stride-2 convolution we often increase the number of features because we’re decreasing the number of activations in the activation map by a factor of 4.we don’t want to decrease the capacity of a layer by too much at a time.

There is one bias for each channel. (Sometimes channels are called features or filters when they are not input channels.) The output shape is 64x4x14x14, and this will therefore become the input shape to the next layer. The next layer has 296 parameters. Let’s ignore the batch axis to keep things simple. So for each of 14\*14=196 locations we are multiplying 296-8=288 weights (ignoring the bias for simplicity), so that’s 196\*288=56\_448 multiplications at this layer. The next layer will have 7\*7\*(1168-16)=56\_448 multiplications.here is that our stride-2 convolution halved the grid size from 14x14 to 7x7, and we doubled the number of filters from 8 to 16, resulting in no overall change in the amount of computation. If we left the number of channels the same in each stride-2 layer, the amount of computation being done in the net would get less and less as it gets deeper. But we know that the deeper layers have to compute semantically rich features.

2)why do we use a larger kernel with MNIST(with simple cnn)in the first conv.?

Ans : We use multiple convolution filters or kernels that run over the image and compute a dot product. Each filter extracts different features from the image.

Lets consider a filter of size 3x3 and an image of size 5x5. We perform an element wise multiplication between the image pixel values that match the size of the kernel and the the kernel itself and sum them up. This provides us a single value for the feature cell.

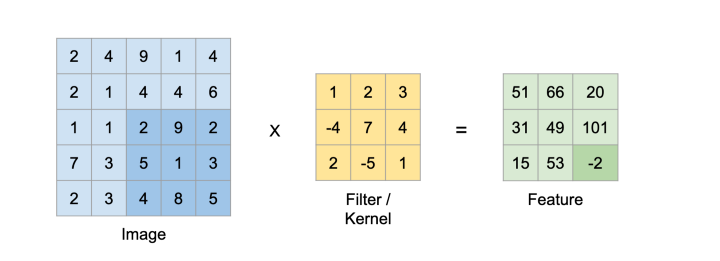


2\*1 + 4\*2 + 9\*3 + 2\*(-4) + 1\*7 + 4\*4 + 1\*2 + 1\*(-5) + 2\*1 = 51

Filter continues to run further on the image and produce new values as shown below.

4\*1 + 9\*2 + 1\*3 + 1\*(-4) + 4\*7 + 4\*4 + 1\*2 + 2\*(-5) + 9\*1 = 66

and so on …



2\*1 + 9\*2 + 2\*3 + 5\*(-4) + 1\*7 + 3\*4 + 4\*2 + 8\*(-5) + 5\*1 = -2

In the above example we are sliding the kernel by 1 pixel. This is called stride. We can have the kernel move by different stride values to extract different kinds of features.

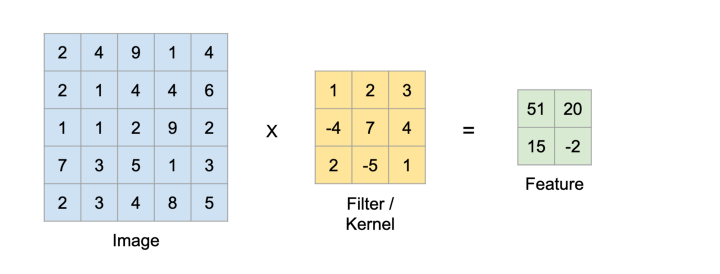
Also the amount of stride we choose affects the size of the feature extracted. The equation to calculate the size of feature for a particular kernel size is as follows:

Feature size = ((Image size − Kernel size) / Stride) + 1

We can put the values for the above example and verify it.

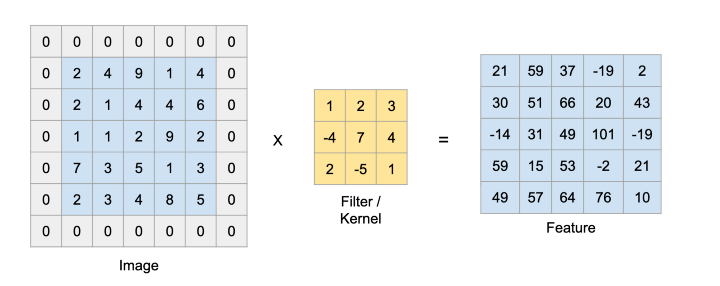
Feature size = ((5 − 3) / 1) + 1 = 3

So with a stride of 2 the kernel of size 3x3 on a image of size 5x5 would only be able to extract a feature of size 2.



You can achieve this by padding the image. Padding is a technique to simply add zeros around the margin of the image to increase it’s dimension. Padding allows us to emphasize the border pixels and in order lose less information.

Here is an example with an input image of size 5x5 which is padded to 7x7 i.e. padding size of 1 and convoluted by a kernel of size 3x3 with stride of 1 resulting in a feature of size 5x5.



The equation to calculate the size of feature for a particular kernel size when considering a padded image is as follows:

Feature size = ((Image size + 2 \* Padding size − Kernel size) / Stride)+1

We can put in the values for the above example and verify it.

Feature size = ((5 + 2 \* 1 − 3) / 1) + 1= 5

For an image with 3 channels i.e. rgb we perform the same operation on all the 3 channels.

A neural network learns those kernel values through back propagation to extract different features of the image. Typically in a convolutional neural network we would have more than 1 kernel at each layer. We can further use those feature maps to perform different tasks like classification, segmentation, object detection etc.

3)what data is saved by activation stats for each layer ?

Ans : An activation record is a contiguous block of storage that manages information required by a single execution of a procedure. When you enter a procedure, you allocate an activation record, and when you exit that procedure, you deallocate it. Basically, it stores the status of the current activation function. So, whenever a function call occurs, then a new activation record is created and it will be pushed onto the top of the stack. It will be in function till the execution of that function. So, once the procedure is completed and it is returned to the calling function, this activation function will be popped out of the stack.

If a procedure is called, an activation record is pushed into the stack, and it is popped when the control returns to the calling function.

Activation Record includes some fields which are –

Return values, parameter list, control links, access links, saved machine status, local data, and temporaries.

Temporaries:

The temporary values, such as those arising in the evaluation of expressions, are stored in the field for temporaries.

Local data:

The field for local data holds data that is local to an execution of a procedure.

Saved Machine States:

The field for Saved Machine Status holds information about the state of the machine just before the procedure is called. This information includes the value of the program counter and machine registers that have to be restored when control returns from the procedure.

Access Link :

It refers to information stored in other activation records that is non-local. The access link is a static link and the main purpose of the access link is to access the data which is not present in the local scope of the activation record. It is a static link.

Control links :

In this case, it refers to an activation record of the caller. They are generally used for links and saved status. It is a dynamic link in nature. When a function calls another function, then the control link points to the activation record of the caller.

Record A contains a control link pointing to the previous record on the stack. Dynamically executed programs are traced by the chain of control links.

Parameter List:

The field for parameters list is used by the calling procedure to supply parameters to the called procedure. We show space for parameters in the activation record, but in practice, parameters are often passed in machine registers for greater efficiency.

Return value:

The field for the return value is used by the called procedure to return a value to the calling procedure. Again in practice, this value is often returned in a register for greater efficiency.

4)how do we get a learners callback after they have completed training ?

Ans : A callback is an object that can perform actions at various stages of training (e.g. at the start or end of an epoch, before or after a single batch, etc). You can use callbacks to: Write TensorBoard logs after every batch of training to monitor your metrics. Periodically save your model to disk. Do early stopping.

5)what are the drawbacks of activations above zero ?

Ans : Zero-centered activation functions ensure that the mean activation value is around zero. This property is important in deep learning because it has been empirically shown that models operating on normalized data whether it be inputs or latent activations enjoy faster convergence.

6)Draw up the benefits and drawbacks of practicing in larger batches.?

Ans : Advantages:

1. Repeated tasks are done fast in batch systems without user interaction.

2. It is most suitable for large organizations however small organizations can also benefit from it.

3. The Batch systems can work offline so it makes less pressure on the processor.

4. The processor utilizes a good time while processing which means it knows which job to process next.

5. The batch systems can handle large repeated work quickly.

Disadvantages:

1. It is hard to debug batch systems.

2. The batch systems are sometimes expensive.

3. If some job takes too much time i.e. if an error occurs in a job then other jobs will wait for an unknown time.

4. Changes to Information regarding orders are delayed by batch processing, increasing the need for lead times if the user is changing the quantity, mode of transport, forwarder, etc.

5. With batch processing, users may be forced to viewing data in both systems to see the most current data, resulting in losing order processing efficiency.

7)why should we avoid starting training with a high learning rate.?

Ans : If your learning rate is set too low, training will progress very slowly as you are making very tiny updates to the weights in your network. However, if your learning rate is set too high, it can cause undesirable divergent behavior in your loss function so we should avoid high learning rate.

8)what are the pros of studying with a high rate of learning.?

Ans :a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train.

9)why do we want to end the training with a low learning rate.?

Ans : This allows large weight changes in the beginning of the learning process and small changes or fine-tuning towards the end of the learning process.Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train.so we want to end training with low learning rate.