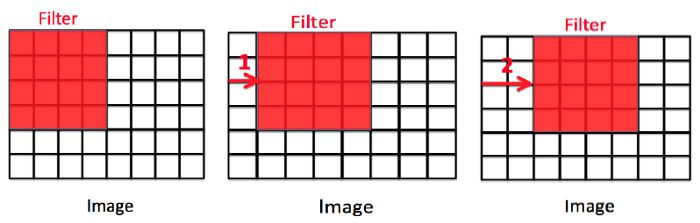
MACHINE LEARNING LAB 7

Hendrick Cardoso 21BDA20

What is Stride, Padding & Pooling? Explain with an example.

**Strides**

When the array is created, the pixels are shifted over to the input matrix. The number of pixels turning to the input matrix is known as the strides. When the number of strides is 1, we move the filters to 1 pixel at a time. Similarly, when the number of strides is 2, we carry the filters to 2 pixels, and so on. They are essential because they control the convolution of the filter against the input, i.e., Strides are responsible for regulating the features that could be missed while flattening the image. They denote the number of steps we are moving in each convolution. The following figure shows how the convolution would work.

****

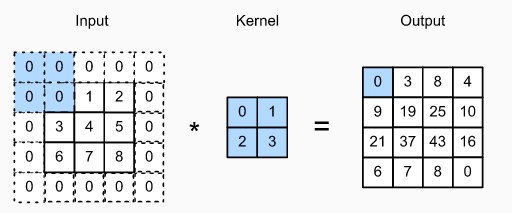
In the first matrix, the stride = 0, second image: stride=2, and the third image: stride=2. The size of the output image is calculated by:

[{(n+2p-f+1)/s}+1][{(n+2p-f+1)/s}]

**Padding**

The padding plays a vital role in creating CNN. After the convolution operation, the original size of the image is shrunk. Also, in the image classification task, there are multiple convolution layers after which our original image is shrunk after every step, which we don’t want. Secondly, when the kernel moves over the original image, it passes through the middle layer more times than the edge layers, due to which there occurs an overlap.

To overcome this problem, a new concept was introduced named padding. It is an additional layer that can add to the borders of an image while preserving the size of the original picture.



So, if an n x n matrix is convolved with an ff matrix with a padding p, then the size of the output image will be:

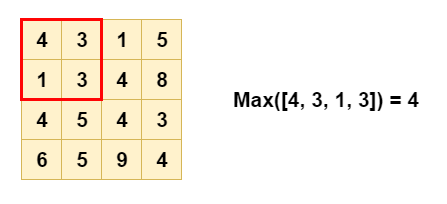
(n+2p-f+1) x (n+2p-f+1)

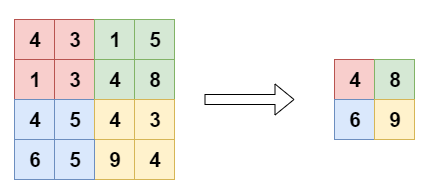
**Pooling**

The pooling layer is another building block of a CNN and plays a vital role in pre-processing an image. In the pre-process, the image size shrinks by reducing the number of parameters if the image is too large. When the picture is shrunk, the pixel density is also reduced; the downscaled image is obtained from the previous layers. Basically, its function is to progressively reduce the spatial size of the image to reduce the network complexity and computational cost. Spatial pooling is also known as down sampling or sub sampling that reduces the dimensionality of each map but retains the essential features. A rectified linear activation function, or ReLU, is applied to each value in the feature map. Relu is a simple and effective nonlinearity that does not change the values in the feature map but is present because later subsequent pooling layers are added. Pooling is added after the nonlinearity is applied to the feature maps. There are three types of spatial pooling:

**1. Max Pooling**

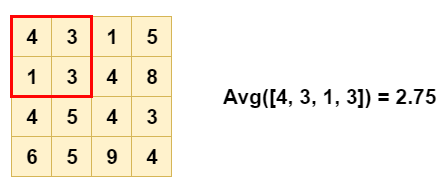
Max pooling is a rule to take the maximum of a region and help to proceed with the most crucial features from the image. It is a sample-based process that transfers continuous functions into discrete counterparts. Its primary objective is to downscale an input by reducing its dimensionality and making assumptions about features contained in the sub-region that were rejected.

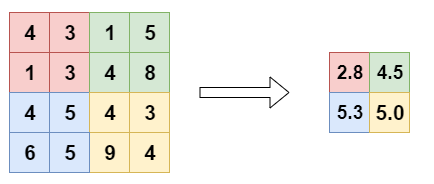




**2. Average Pooling**

It is different from Max Pooling; it retains information about the lesser essential features. It simply downscales by dividing the input matrix into rectangular regions and calculating the average values of each area.





**3. Sum Pooling**

It is similar to Max pooling, but instead of calculating the maximum value, we calculate the mean of each sub-region.

What is overfitting? How to overcome overfitting in an ML model?

It is a common pitfall in deep algorithm algorithms in which a model tries to fit the[training data](https://www.v7labs.com/blog/quality-training-data-for-machine-learning-guide)entirely and ends up memorizing the data patterns and the noise and random fluctuations. These models fail to generalize and perform well in the case of unseen data scenarios, defeating the model's purpose.

When can overfitting occur?

The high variance of the model performance is an indicator of an overfitting problem.

The training time of the model or its architectural complexity may cause the model to overfit. If the model trains for too long on the training data or is too complex, it learns the noise or irrelevant information within the dataset.

Overfitting happens when:

1. The data used for training is not cleaned and contains garbage values. The model captures the noise in the training data and fails to generalize the model's learning.
2. The model has a high variance.
3. The training data size is not enough, and the model trains on the limited training data for several epochs.
4. The architecture of the model has several neural layers stacked together. Deep neural networks are complex and require a significant amount of time to train, and often lead to overfitting the training set.

How to prevent it?

1. **Train with more data**

With the increase in the training data, the crucial features to be extracted become prominent. The model can recognize the relationship between the input attributes and the output variable. The only assumption in this method is that the data to be fed into the model should be clean; otherwise, it would worsen the problem of overfitting.

1. **Data augmentation**

An alternative method to training with more data is data augmentation, which is less expensive and safer than the previous method. Data augmentation makes a sample data look slightly different every time the model processes it.

1. **Addition of noise to the input data**

Another similar option as data augmentation is adding noise to the input and output data. Adding noise to the input makes the model stable without affecting data quality and privacy while adding noise to the output makes the data more diverse. Noise addition should be done in limit so that it does not make the data incorrect or too different.

1. **Feature selection**

Every model has several parameters or features depending upon the number of layers, number of neurons, etc.  The model can detect many redundant features or features determinable from other features leading to unnecessary complexity. We very well know that the more complex the model, the higher the chances of the model to overfit.

1. **Cross-validation**

Cross-validation is a robust measure to prevent overfitting. The complete dataset is split into parts. In standard K-fold cross-validation, we need to partition the data into k folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining holdout fold as the test set. This method allows us to tune the hyper parameters of the neural network or machine learning model and test it using completely unseen data.

1. **Simplify data**

Till now, we have come across model complexity to be one of the top reasons for overfitting. The data simplification method is used to reduce overfitting by decreasing the complexity of the model to make it simple enough that it does not overfit. Some of the procedures include pruning a decision tree, reducing the number of parameters in a neural network, and using dropout on a neutral network.

1. **Regularization**

If overfitting occurs when a model is too complex, reducing the number of features makes sense. Regularization methods like Lasso, L1 can be beneficial if we do not know which features to remove from our model. Regularization applies a "penalty" to the input parameters with the larger coefficients, which subsequently limits the model's variance

1. **Ensembling**

It is a machine learning technique that combines several base models to produce one optimal predictive model. In Ensemble learning,  the predictions are aggregated to identify the most popular result. Well-known ensemble methods include bagging and boosting, which prevents overfitting as an ensemble model is made from the aggregation of multiple models.

1. **Early stopping**

This method aims to pause the model's training before memorizing noise and random fluctuations from the data. There can be a risk that the model stops training too soon, leading to underfitting. One has to come to an optimum time/iterations the model should train.

1. **Adding dropout layers**

Large weights in a neural network signify a more complex network. Probabilistically dropping out nodes in the network is a simple and effective method to prevent overfitting. In regularization, some number of layer outputs are randomly ignored or “*dropped out*” to reduce the complexity of the model.