Machine Learning 1 Assignment

Name: Aaran D'Lima **Reg. No.:21BDA23**

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

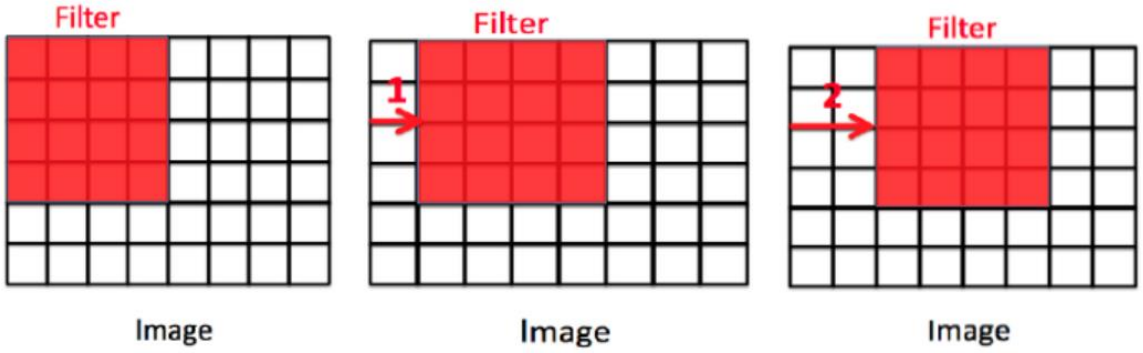
Strides

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:

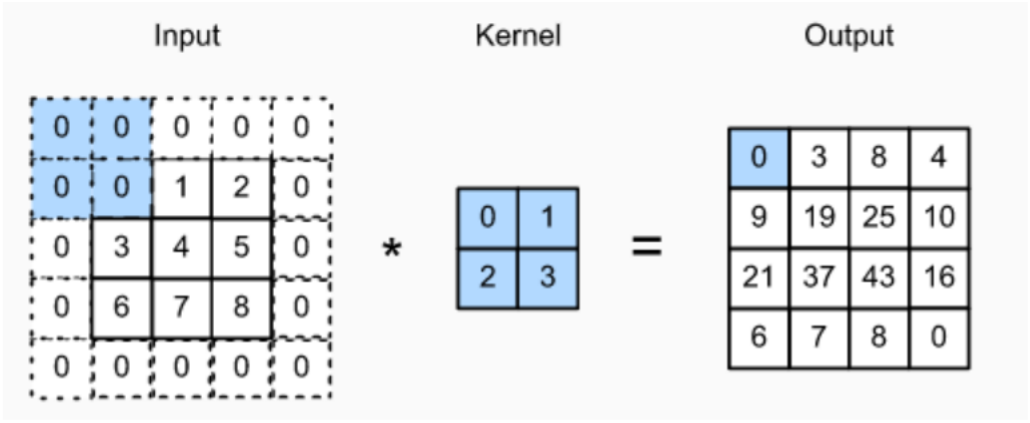
$$[\{(n+2p-f+1)/s\}+1][\{(n+2p-f+1)/s\}]$$

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. For example:



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:

$$(n+2p-f+1) \times (n+2p-f+1)$$

Pooling

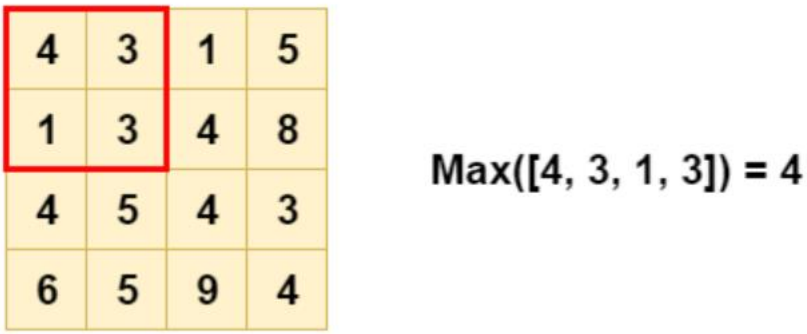
The pooling layer is another building block of a CNN and plays a vital role in pre-processing an image. In the pre-process, the image size shrinks by reducing the number of parameters if the image is too large.

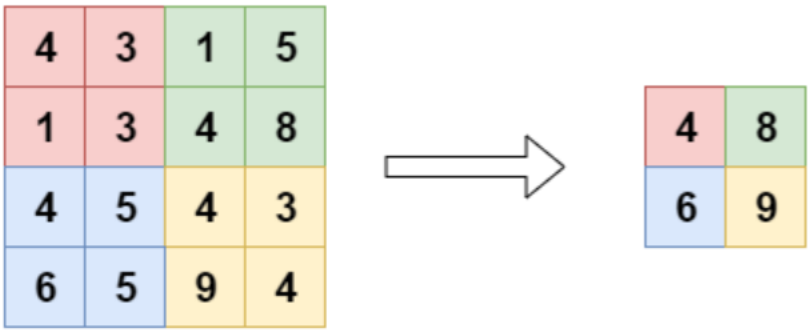
When the picture is shrunk, the pixel density is also reduced, the downscaled image is obtained from the previous layers.

Basically, its function is to progressively reduce the spatial size of the image to reduce the network complexity and computational cost. Pooling is added after the nonlinearity is applied to the feature maps. There are three types of spatial pooling:

1. Max Pooling

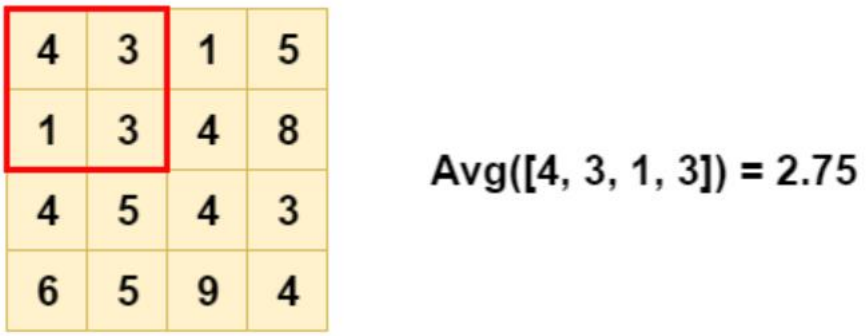
Max pooling is a rule to take the maximum of a region and help to proceed with the most crucial features from the image. It is a sample-based process that transfers continuous functions into discrete counterparts. Its primary objective is to downscale an input by reducing its dimensionality and making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

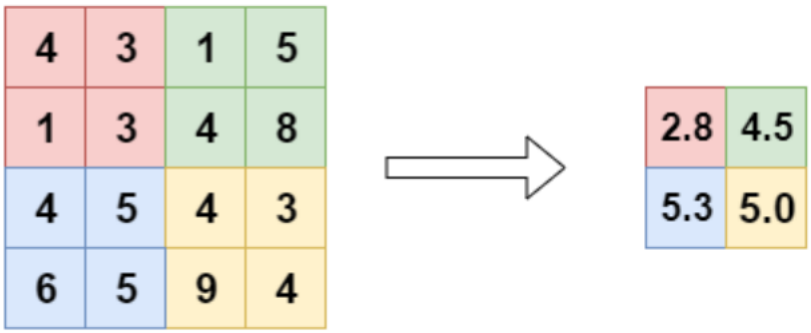




2. Average Pooling

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.

Reference: https://www.codingninjas.com/codestudio/library/convolution-layer-padding-stride-and-pooling-in-cnn

Q) What is over fitting? How to overcome over fitting in ML model?

A statistical model is said to be over-fitted when we feed it a lot more data than necessary. To make it relatable, imagine trying to fit into oversized apparel. When a model fits more data than it actually needs, it starts catching the noisy data and inaccurate values in the data. As a result, the efficiency and accuracy of the model decrease.

Examples of Overfitting

Let's say we want to predict if a student will land a job interview based on her resume. Now, assume we train a model from a dataset of 10,000 resumes and their outcomes.

Next, we try the model out on the original dataset, and it predicts outcomes with 99% accuracy… wow! But now comes the bad news. When we run the model on a new ("unseen") dataset of resumes, we only get 50% accuracy. Our model doesn't generalize well from our training data to unseen data. This is known as overfitting, and it's a common problem in machine learning and data science.

How to Prevent Overfitting

1. Cross-Validation

One of the most powerful features to avoid/prevent overfitting is cross-validation. The idea behind this is to use the initial training data to generate mini train-test-splits, and then use these splits to tune your model.

In a standard k-fold validation, the data is partitioned into k-subsets also known as folds. After this, the algorithm is trained iteratively on k-1 folds while using the remaining folds as the test set, also known as holdout fold.

The cross-validation helps us to tune the hyperparameters with only the original training set. It basically keeps the test set separately as a true unseen data set for selecting the final model. Hence, avoiding overfitting altogether.

2. Training with More Data

This technique might not work every time, as we have also discussed in the example above, where training with a significant amount of population helps the model. It basically helps the model in identifying the signal better.

But in some cases, the increased data can also mean feeding more noise to the model. When we are training the model with more data, we have to make sure the data is clean and free from randomness and inconsistencies.

3. Removing Features

Although some algorithms have an automatic selection of features. For a significant number of those who do not have a built-in feature selection, we can manually remove a few irrelevant features from the input features to improve the generalization.

One way to do it is by deriving a conclusion as to how a feature fits into the model. It is quite similar to debugging the code line-by-line.

In case if a feature is unable to explain the relevancy in the model, we can simply identify those features. We can even use a few feature selection heuristics for a good starting point.

4. Early Stopping

When the model is training, you can actually measure how well the model performs based on each iteration. We can do this until a point when the iterations improve the model's performance. After this, the model overfits the training data as the generalization weakens after each iteration.

So basically, early stopping means stopping the training process before the model passes the point where the model begins to overfit the training data. This technique is mostly used in deep learning.

5. Regularization

It basically means, artificially forcing your model to be simpler by using a broader range of techniques. It totally depends on the type of learner that we are using. For example, we can prune a decision tree, use a dropout on a neural network or add a penalty parameter to the cost function in regression.

Quite often, regularization is a hyperparameter as well. It means it can also be tuned through cross-validation.

6. Ensembling

This technique basically combines predictions from different Machine Learning models. Two of the most common methods for ensembling are listed below:

- Bagging attempts to reduce the chance overfitting the models
- Boosting attempts to improve the predictive flexibility of simpler models

Even though they are both ensemble methods, the approach totally starts from opposite directions. Bagging uses complex base models and tries to smooth out their predictions while boosting uses simple base models and tries to boost its aggregate complexity.

References: https://www.edureka.co/blog/overfitting-in-machine-learning/