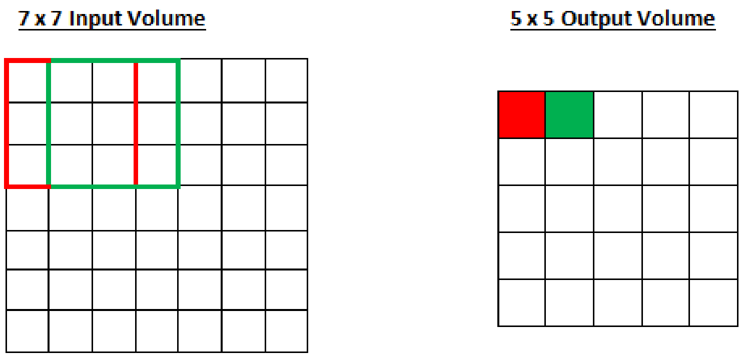
LAB- 07

CHRISTINE BIJU JACOB

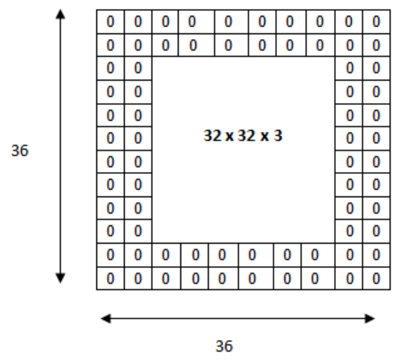
21BDA06

Question 2) What is Stride, Padding & Pooling? Explain with an example.

Stride is a component of convolutional neural networks, or neural networks tuned for the compression of images and video data. Stride is a parameter of the neural network's filter that modifies the amount of movement over the image or video. For example, if a neural network's stride is set to 1, the filter will move one pixel, or unit, at a time. The size of the filter affects the encoded output volume, so stride is often set to a whole integer, rather than a fraction or decimal.

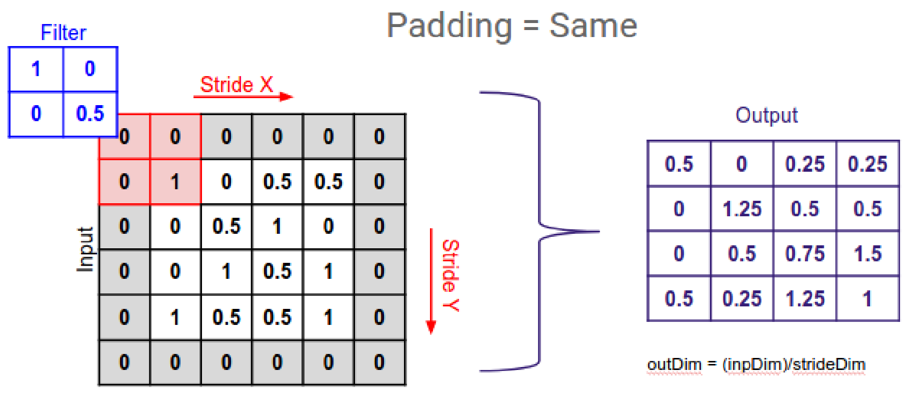


Imagine a convolutional neural network is taking an image and analyzing the content. If the filter size is 3x3 pixels, the contained nine pixels will be converted down to 1 pixel in the output layer. Naturally, as the stride, or movement, is increased, the resulting output will be smaller. Stride is a parameter that works in conjunction with padding, the feature that adds blank, or empty pixels to the frame of the image to allow for a minimized reduction of size in the output layer. Roughly, it is a way of increasing the size of an image, to counteract the fact that stride reduces the size. Padding and stride are the foundational parameters of any convolutional neural network.



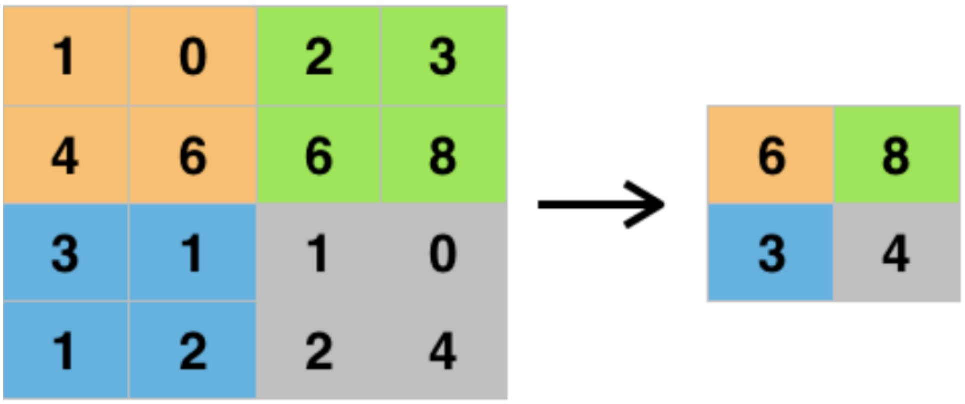
The input volume is 32 x 32 x 3. If we imagine two borders of zeros around the volume, this gives us a 36 x 36 x 3 volume. Then, when we apply our conv layer with our three 5 x 5 x 3 filters and a stride of 1, then we will also get a 32 x 32 x3 output volume.

Padding is a term relevant to convolutional neural networks as it refers to the amount of pixels added to an image when it is being processed by the kernel of a CNN. For example, if the padding in a CNN is set to zero, then every pixel value that is added will be of value zero. If, however, the zero padding is set to one, there will be a one pixel border added to the image with a pixel value of zero.



Padding works by extending the area of which a convolutional neural network processes an image. The kernel is the neural networks filter which moves across the image, scanning each pixel and converting the data into a smaller, or sometimes larger, format. In order to assist the kernel with processing the image, padding is added to the frame of the image to allow for more space for the kernel to cover the image. Adding padding to an image processed by a CNN allows for more accurate analysis of images.

Pooling is a feature commonly imbibed into Convolutional Neural Network (CNN) architectures. The main idea behind a pooling layer is to “accumulate” features from maps generated by convolving a filter over an image. Formally, its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. The most common form of pooling is max pooling.



Max pooling is done to in part to help over-fitting by providing an abstracted form of the representation. As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation. Max pooling is done by applying a max filter to (usually) non-overlapping subregions of the initial representation. The other forms of pooling are: average, general.

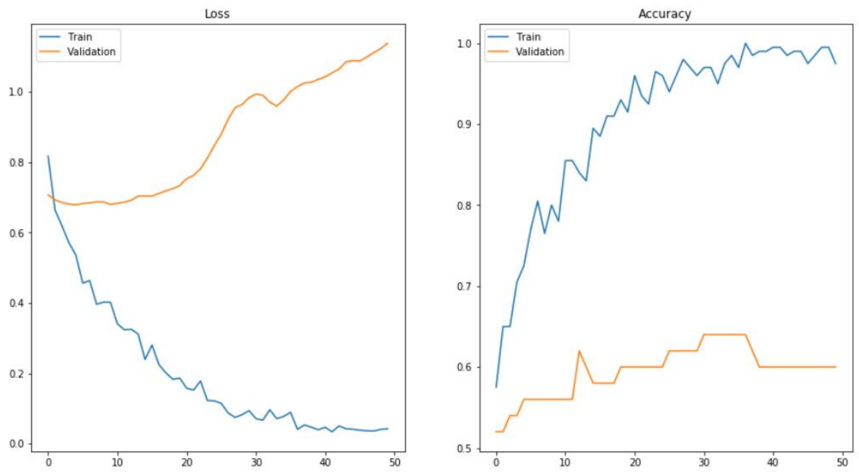
Question 4) What is overfitting? How to overcome overfitting in an ML model?

It is a common pitfall in deep learning algorithms in which a model tries to fit the training data 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.

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.



Here are some techniques to avoid overfitting:

- **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.

- **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.

- **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.

- **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.

- **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 hyperparameters of the neural network or machine learning model and test it using completely unseen data.

- **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.

- **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.

- **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.

- **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.

- **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.