**Classification of Fashion-MNIST with a**

**simple CNN in Keras:**

**Convolution Neural Networks[CNN] :**

Convolution Neural Network is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets has the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

**Convolution Operations:**

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset, similar to how we would

A close up of a map

Description automatically generated

Fig:1 Convolution Neural Network Architecture

As mentioned in the above figure the entire architecture consists of input images passing through several convolutional layers(convoluted with feature map) in order to reduce the size of convoluted matrices by Appling transform with single unit between convolutional matrices and feature map to get reduced size matrices.

A close up of a map

Description automatically generated

Fig:2

## **Pooling Layer:**

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to**decrease the computational power required to process the data** through dimensionality reduction. Furthermore, it is useful for **extracting dominant features** which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

* **Max Pooling**returns the**maximum value**from the portion of the imagecovered by the Kernel.
* The Convolutional Layer and the Pooling Layer together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low levels details even further, but at the cost of more computational power.
* After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

**Padding:**

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.

* **Same Padding**:When we augment the 5x5x1 image into a 6x6x1 image and then apply the 3x3x1 kernel over it, we find that the convolved matrix turns out to be of dimensions 5x5x1.
* **Valid Padding**:if we perform the same operation without padding, we are presented with a matrix which has dimensions of the Kernel (3x3x1) itself.

A close up of text on a white background

Description automatically generated

## Fig:3 Classification — Fully Connected Layer (FC Layer)

we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

**Activation Function:**

* To understand the work of activation in CNNs, we must first understand the basic functionality of the CNN. The Convolutional Neural Networks are designed to capture some features from images starting with basic features at the initial level and very high-level features at the deeper levels. By high level I mean the features which are more generalized for a class. This generalization would not be possible with a linear mapping as in that case a high level of abstraction/generalization would not be possible. Hence, to map a class of images into a manifold of feature vector, we need activation, without it, it would be difficult to generalize as pictures in a class can have to much intra-class variations.
* Activation Layer is an activation function that decides the final value of a neuron. Suppose a cell value should be 1 ideally, however it has a value of 0.85, since you can never achieve a probability of 1 in CNN thus, we apply an activation function. Ex if cell value is greater than 0.7 make 1 else make it 0. In this way one can easily achieve an image with sharp features. You can visit here
* For every neural network we have to apply activation function to find out which node should be fired, in other terms to match the output nodes we have to apply different nonlinearities by using these activation function on each layer. In Convolutional neural network we apply mostly RELU(not compulsory we can use different activation function) activation function which replace the all negative values to 0 and remains same with the positive values.

**Rectifier Activation Function:**

Activation function decides, whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

* In Convolutional neural network we apply mostly RELU(not compulsory we can use different activation function) activation function which replace the all negative values to 0 and remains same with the positive values.
* To use stochastic gradient descent with [backpropagation of errors](https://machinelearningmastery.com/implement-backpropagation-algorithm-scratch-python/) to train deep neural networks, an activation function is needed that looks and acts like a linear function.
* To neurons with this nonlinearity as Rectified Linear Units (ReLUs). Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units.

**Kernel /Filter:**

* Kernels are used in convolutional layers to extract features. They basically are filters that you apply on a small region of the image.
* Number of filters and filter size are two hyper-parameters, you can tune them during the training. Each dataset requires a different setting of hyper-parameters
* A filter is represented by a vector of weights with which we convolve the input.
* In the context of CNN, a filter is a set of learnable weights which are learned using the backpropagation algorithm. When you convolve this filter across the corresponding input, you are basically trying to find out the similarity between the stored template and different locations in the input.
* In a CNN the weights (in the convolutional layers) are a small matrix (often 3x3) which is dot produced with each pixel to produce a new pixel thus acting as image filters. The new images produced by each neuron/filter in a layer are then combined and then passed as the inputs to every neuron in the next layer and so on until the end of the network is reached (there is often a single dense layer at the end to turn the image output of the final convolutional layer into the numerical class prediction we are typically training it to produce)

**Filter size:**

* Filter size refers to the dimensions of the filter/kernel in the ConvNet.
* To say it informally, the filter size is how many neighbor information you can see when processing the current layer. When the filter size is 3\*3, that means each neuron can see its left, right, upper, down, upper left, upper right, lower left, lower right, as a total of 8 neighbor information.
* 3\*3 is the currently most widely used filter size. This is because 3\*3 is usually the smallest filter size to do image processing

**How can I decide Filter size?**

* You should initially use fewer filters and gradually increase and monitor the error rate to see how it is varying.
* It is always a tradeoff between speed and accuracy. But there is several to-do`s and never-do`s.
* **Kernel size:** smaller + deeper = better. Deeper - better up to some level. Wider - better, but at some level there is an accuracy saturation, and computations grow up quadratically.

**Fashion-MNIST:**

* Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples.
* Each example is a 28x28 grayscale image, associated with a label from 10 classes.
* Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms.
* It shares the same image size and structure of training and testing splits.

**JUPYTER NOTEBOOK:**

A screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generated

A picture containing clock

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

**A picture containing knife, table

Description automatically generated**

A screenshot of a cell phone

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a social media post

Description automatically generatedA screenshot of a cell phone

Description automatically generated