**Keras**

1. **Introduction**

🡪 Deep learning is one of the major subfield of machine learning framework. Machine learning is the study of design of algorithms, inspired from the model of human brain. Deep learning is becoming more popular in data science fields like robotics, artificial intelligence(AI), audio & video recognition and image recognition. Artificial neural network is the core of deep learning methodologies. Deep learning is supported by various libraries such as Theano, TensorFlow, Caffe, Mxnet etc., Keras is one of the most powerful and easy to use python library, which is built on top of popular deep learning libraries like TensorFlow, Theano, etc., for creating deep learning models.

**🡺Overview Of Keras**

🡪 Keras runs on top of open source machine libraries like TensorFlow, Theano or Cognitive Toolkit (CNTK). Theano is a python library used for fast numerical computation tasks. TensorFlow is the most famous symbolic math library used for creating neural networks and deep learning models. TensorFlow is very flexible and the primary benefit is distributed computing. CNTK is deep learning framework developed by Microsoft. It uses libraries such as Python, C#, C++ or standalone machine learning toolkits. Theano and TensorFlow are very powerful libraries but difficult to understand for creating neural networks.

🡪 Keras is based on minimal structure that provides a clean and easy way to create deep learning models based on TensorFlow or Theano. Keras is designed to quickly define deep learning models. Well, Keras is an optimal choice for deep learning applications.

**🡺Features**

🡪 Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

* Consistent, simple and extensible API.
* Minimal structure - easy to achieve the result without any frills.
* It supports multiple platforms and backends.
* It is user friendly framework which runs on both CPU and GPU.
* Highly scalability of computation.

🡺Benefits

🡪 Keras is highly powerful and dynamic framework and comes up with the following advantages:

* Larger community support.
* Easy to test
* Keras neural networks are written in Python which makes things simpler.
* Keras supports both convolution and recurrent networks.
* Deep learning models are discrete components, so that, you can combine into many ways.

🡺Keras depends on the following libraries:

* + Numpy
  + Pandas
  + Scikit-learn
  + Matplolib
  + Scipy
  + Seaborn

🡪 Spyder is an IDE for executing python applications. Let us install this IDE in our conda environment using the below command



1. **Overview of Deep Learning**

🡪 Deep learning is an evolving subfield of machine learning. Deep learning involves analyzing the input in layer by layer manner, where each layer progressively extracts higher level information about the input.

🡪Let us take a simple scenario of analyzing an image. Let us assume that your input image is divided up into a rectangular grid of pixels. Now, the first layer abstracts the pixels. The second layer understands the edges in the image. The Next layer constructs nodes from the edges. Then, the next would find branches from the nodes. Finally, the output layer will detect the full object. Here, the feature extraction process goes from the output of one layer into the input of the next subsequent layer.

🡪By using this approach, we can process huge amount of features, which makes deep learning a very powerful tool. Deep learning algorithms are also useful for the analysis of unstructured data. Let us go through the basics of deep learning in this chapter.

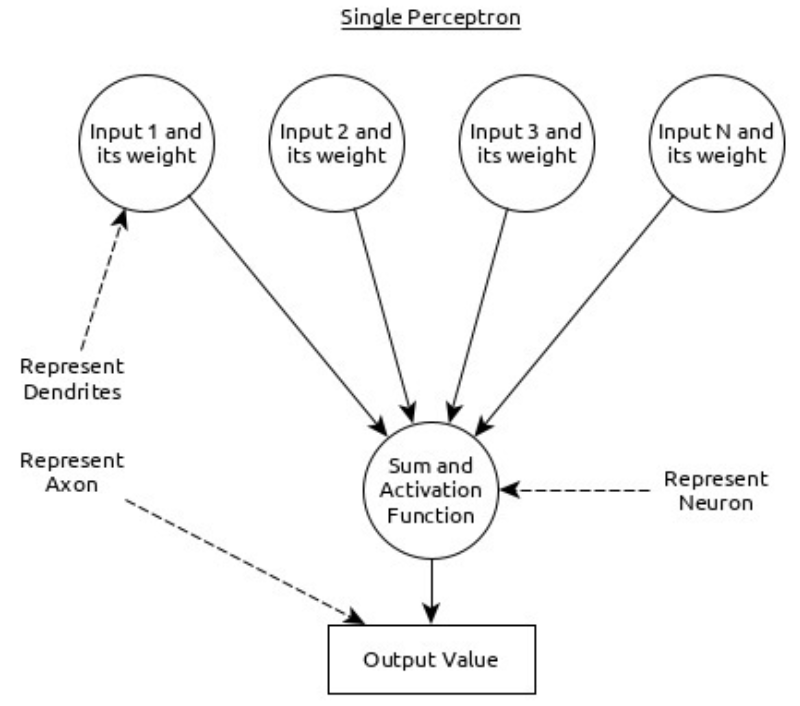
**🡺Artificial Neural Networks**

🡪 **The most popular and primary approach of deep learning is using “Artificial neural network” (ANN)**. They are inspired from the model of human brain, which is the most complex organ of our body. The human brain is made up of more than 90 billion tiny cells called “Neurons”. Neurons are inter-connected through nerve fiber called “axons” and “Dendrites”. The main role of axon is to transmit information from one neuron to another to which it is connected.

🡪Similarly, the main role of dendrites is to receive the information being transmitted by the axons of another neuron to which it is connected. Each neuron processes a small information and then passes the result to another neuron and this process continues. This is the basic method used by our human brain to process huge about of information like speech, visual, etc., and extract useful information from it.

🡪Based on this model, ANNs are made up of multiple nodes which is similar to neurons. Nodes are tightly interconnected and organized into different hidden layers. The input layer receives the input data and the data goes through one or more hidden layers sequentially and finally the output layer predict something useful about the input data. For example, the input may be an image and the output may be the thing identified in the image, say a “Cat”.

🡪A single neuron (called as perception is ANN) can be represented as below:



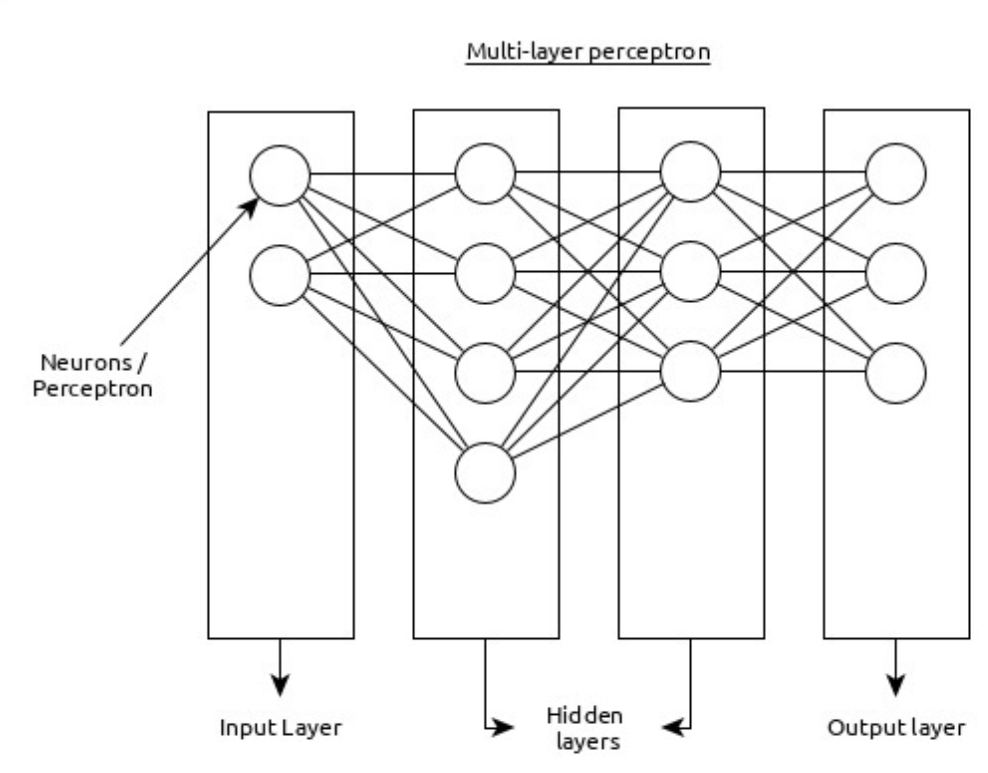
🡪 Here,

* Multiple input along with weight represents dendrites.
* Sum of input along with activation function represents neurons. **Sum** actually mean computed value of all inputs and activation function represent a function, which modify the **Sum** value into 0, 1 or 0 to 1.
* Actual output represent axon and the output will be received by neuron in next layer.

🡪Let us understand different types of the neural networks in this section.

* 1. **Multi-Layer Perception**

🡪Multi-Layer perceptron is the simplest form of ANN. It consists of a single input layer, one or more hidden layer and finally an output layer. A layer consists of a collection of perceptron. Input layer is basically one or more features of the input data. Every hidden layer consists of one or more neurons and process certain aspect of the feature and send the processed information into the next hidden layer. The output layer process receives the data from last hidden layer and finally output the result.

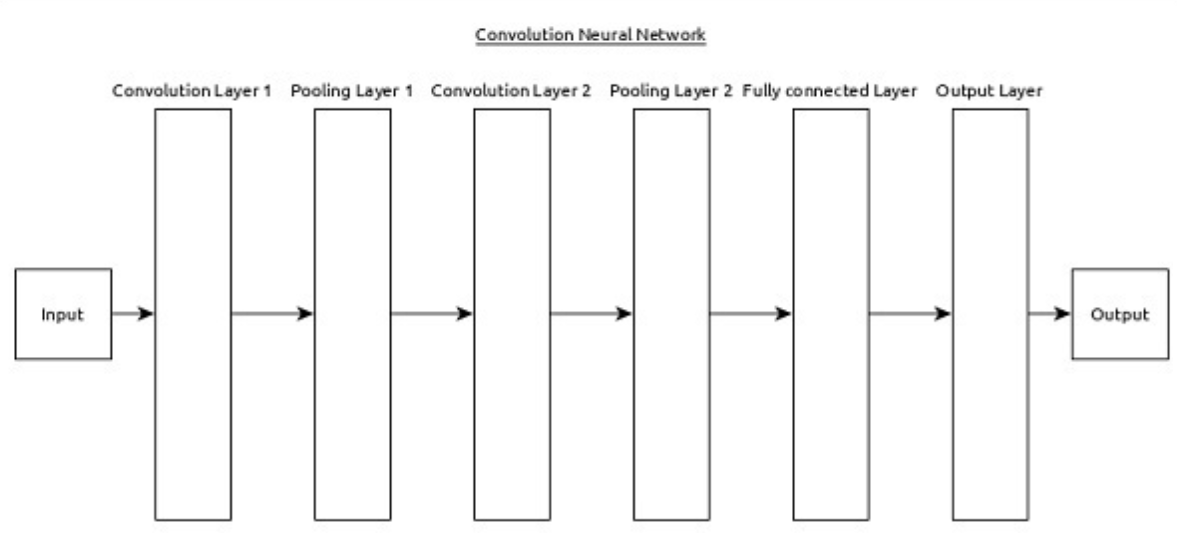


* 1. **Convolutional Neural Network (CNN)**

🡪Convolutional neural network is one of the most popular ANN. It is widely used in the fields of image and video recognition. It is based on the concept of convolution, a mathematical concept. It is almost similar to multi-layer perceptron except it contains series of convolution layer and pooling layer before the fully connected hidden neuron layer. It has three important layers as shown below:

* **Convolution layer** − It is the primary building block and perform computational tasks based on convolution function.
* **Pooling layer** − It is arranged next to convolution layer and is used to reduce the size of inputs by removing unnecessary information so computation can be performed faster.
* **Fully connected layer** − It is arranged to next to series of convolution and pooling layer and classify input into various categories.

🡪A simple CNN can be represented as below:



🡪Here,

* 2 series of convolution and pooling layer is used and it receives and process the input (e.g. image)
* A single fully connected layer is used and it is used to output the data (e.g. classification of image)
  1. **Recurrent Neural Network (RNN)**

🡪Recurrent Neural Networks (RNN) are useful to address the flaw in other ANN models. Well, **Most of the ANN doesn’t remember the steps from previous situations and learned to make decisions based on context in training**. **Meanwhile, RNN stores the past information and all its decisions are taken from what it has learnt from the past**.

🡪This approach is mainly useful in image classification. Sometimes, we may need to look into the future to fix the past. In this case bidirectional RNN is helpful to learn from the past and predict the future. For example, we have handwritten samples in multiple inputs. Suppose, we have confusion in one input then we need to check again other inputs to recognize the correct context which takes the decision from the past.

1. **Workflow of ANN**

🡪Let us first understand the different phases of the deep learning and than, learn how keras helps in the process of deep learning.

* 1. **Collect required data**

🡪Deep learning requires lot of input data to successfully learn and predict the result. So, first collect as much data as possible

* 1. **Analyze Data**

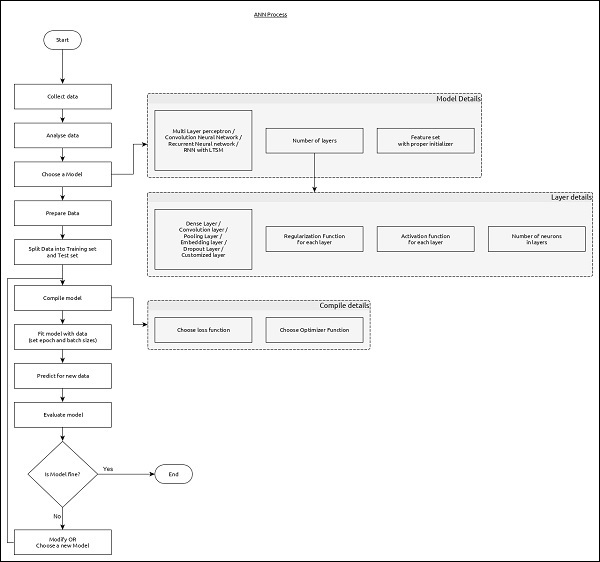
🡪Analyze the data and acquire a good understanding of the data. The better understanding of the data is required to select the correct ANN algorithm.

* 1. **Choose an algorithm (model)**

🡪Choose an algorithm, which will best fit for the type of learning process (e.g image classification, text processing, etc.,) and the available input data. Algorithm is represented by Model in Keras. Algorithm includes one or more layers. Each layers in ANN can be represented by Keras Layer in Keras.

* + - **Prepare data** − Process, filter and select only the required information from the data.
    - **Split data** − Split the data into training and test data set. Test data will be used to evaluate the prediction of the algorithm / Model (once the machine learn) and to cross check the efficiency of the learning process.
    - **Compile the model** − Compile the algorithm / model, so that, it can be used further to learn by training and finally do to prediction. This step requires us to choose loss function and Optimizer. loss function and Optimizer are used in learning phase to find the error (deviation from actual output) and do optimization so that the error will be minimized.
    - **Fit the model** − The actual learning process will be done in this phase using the training data set.
    - **Predict result for unknown value** − Predict the output for the unknown input data (other than existing training and test data)
    - **Evaluate model** − Evaluate the model by predicting the output for test data and cross-comparing the prediction with actual result of the test data.
    - **Freeze, Modify or choose new algorithm** − Check whether the evaluation of the model is successful. If yes, save the algorithm for future prediction purpose. If not, then modify or choose new algorithm / model and finally, again train, predict and evaluate the model. Repeat the process until the best algorithm (model) is found.

🡪The above steps can be represented by below flow chart:



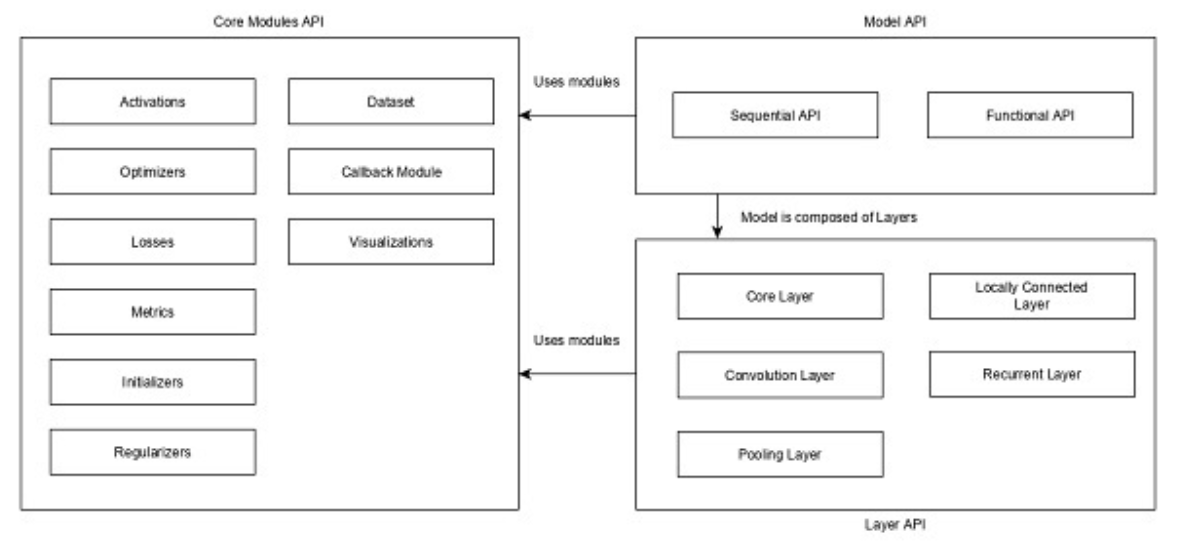
1. **Architecture of keras**

**🡪Keras API can be divided into three main categories:**

* + **Model**
  + **Layer**
  + **Core Modules**

🡪 In Keras, every ANN is represented by Keras Models. **In turn, every Keras Model is composition of Keras Layers and represents ANN layers like input, hidden layer, output layers, convolution layer, pooling layer, etc**., Keras model and layer access Keras modules for activation function, loss function, regularization function, etc., Using Keras model, Keras Layer, and Keras modules, any ANN algorithm (CNN, RNN, etc.,) can be represented in a simple and efficient manner.

🡪The following diagram depicts the relationship between model, layer and core modules



🡪 Let us see the overview of Keras models, Keras layers and Keras modules.

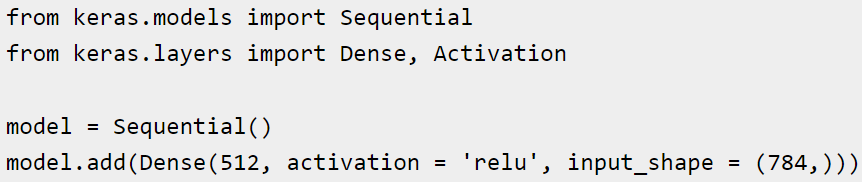
🡺Model

🡪Keras models are of two types as mentioned below:

* 1. **Sequential Model**

🡪Sequential model is basically a linear composition of keras layers. Sequential model is easy, minimal as well as has the ability to represent nearly all available neural networks.

🡪A simple sequential mode is as follow:



🡪 Where,

* **Line 1** imports **Sequential** model from Keras models
* **Line 2** imports **Dense** layer and **Activation** module
* **Line 4** create a new sequential model using **Sequential API**
* **Line 5** adds a dense layer (Dense API) with **relu** activation (using Activation module) function.

🡪 Sequential model exposes Model class to create customized models as well. We can use sub-classing concept to create our own complex model.

* 1. **Funcional API**

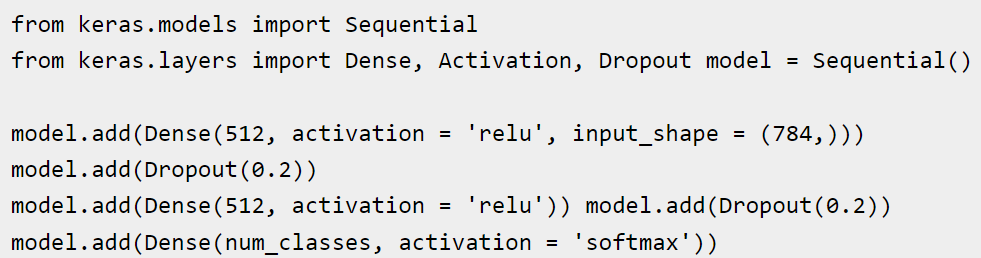
🡪Functional APIs are used to make the complex models

**🡺Layer**

🡪 Each Keras layer in the Keras model represent the corresponding layer (input layer, hidden layer and output layer) in the actual proposed neural network model. Keras provides a lot of pre-build layers so that any complex neural network can be easily created. Some of the important Keras layers are specified below,

* + **Core Layers**
  + **Convolution Layers**
  + **Pooling Layers**
  + **Recurrent Layers**

🡪A simple python code to represent a neural network model using sequential mode is as follow:



🡪 Where,

* + **Line 1** imports **Sequential** model from Keras models
  + **Line 2** imports **Dense** layer and **Activation** module
  + **Line 4** create a new sequential model using **Sequential** API
  + **Line 5** adds a dense layer (Dense API) with **relu** activation (using Activation module) function.
  + **Line 6** adds a dropout layer (Dropout API) to handle over-fitting.
  + **Line 7** adds another dense layer (Dense API) with **relu** activation (using Activation module) function.
  + **Line 8** adds another dropout layer (Dropout API) to handle over-fitting.
  + **Line 9** adds final dense layer (Dense API) with **softmax** activation (using Activation module) function.

🡪 Keras also provides options to create our own customized layers. Customized layer can be created by sub-classing the **Keras.Layer** class and it is similar to sub-classing Keras models.

**🡺Core Modules**

🡪 Keras also provides a lot of built-in neural network related functions to properly create the Keras model and Keras layers. Some of the function are as follows:

1. **Activation module** – Activation function is an important concept in ANN and activation modules provides many activation functions like softmax, relu, etc.
2. **Loss module** – Loss module provides loss functions like mean\_squared\_error, mean\_absolute\_error, poisson, etc
3. **Optimizer module** – Optimizer module provides optimizer function like adam, sgd, etc.
4. **Regularizers** – Regularizer module provides functions like L1 regularization, L2 regularization, etc.
5. **Keras -Modules**

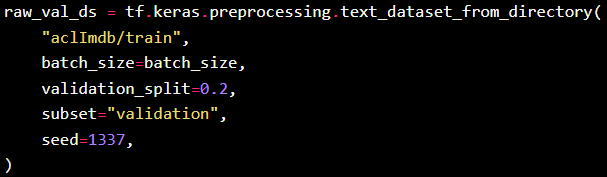
🡪As we learned earlier, Keras modules contains pre-defined classes, functions and variables which are useful for deep learning algorithm. Let us learn the modules provided by Keras in this chapter.

🡺Available Modules:

1. **What is Validation\_split?**

validation\_split: Float between 0 and 1. Fraction of the training data to be used as validation data. **The model will set apart this fraction of the training data, will not train on it, and will evaluate the loss and any model metrics on this data at the end of each epoch**.

🡪Example:



1. What is batch\_size ?

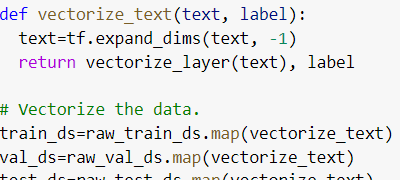
Batch size is a term used in machine learning and refers to the number of training examples utilized in one iteration. The batch size can be one of three options:

1. **batch mode**: where the batch size is equal to the total dataset thus making the iteration and epoch value equivalent.
2. **mini-batch mode**: where the batch size is greater than one but less than the total dataset size. Usually, a number that can be divided into the total dataset size.
3. **stochastic mode**: where the batch size is equal to one. Therefore the gradient and the neural network parameters are updated after each sample.
4. The seed() method is used to initialize the random number generator.

🡪The random number generator needs a number to start with (a seed value), to be able to generate a random number.

🡪By default the random number generator uses the current system time.

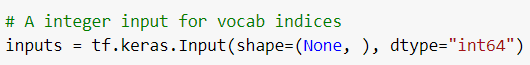
1. **Eager execution** is a powerful execution environment that evaluates operations immediately. It does not build graphs, and the operations return actual values instead of computational graphs to run later. With Eager execution, TensorFlow calculates the values of tensors as they occur in your code.
2. In programming language theory, **lazy evaluation**, or call-by-need, is an evaluation strategy which delays the evaluation of an expression until its value is needed (non-strict evaluation) and which also avoids repeated evaluations (sharing).
3. The expand\_dims() function in NumPy is used **to expand the shape of an input array that is passed to it**. This operation is done in such a way that when a new axis is inserted, it appears in the axis position of the resulting expanded array shape.



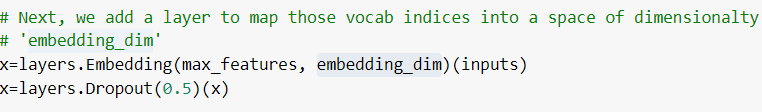
1. Caching is an optimization technique that consists in keeping recently (or frequently) used data in a memory location that has cheap and fast access for repeated queries.

🡪Due to multiple reasons, accessing the data from its original source can be expensive and caching appears as a solution that alleviates this problem.

1. prefetch() function is used to **produce a dataset that prefetches the specified elements from this given dataset**. Parameters: This function accepts a parameter which is illustrated below: bufferSize: It is an integer value that specifies the number of elements to be prefetched.
2. Python input() function is **used to take user input**. By default, it returns the user input in form of a string.



1. **Embedding layer** enables us to convert **each word into a fixed length vector of defined size. The resultant vector is a dense one with having real values instead of just 0’s and 1’s**. The fixed length of word vectors helps us to represent words in a better way along with reduced dimensions.



1. (**More to found on it**)The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/(1 - rate) such that the sum over all inputs is unchanged.
2. **Conv1d layer is often used in pattern recognition model and extract the feature from the vectors**.

🡪If there is no pre-trained model before initialization, it is required to configure network model related parameters. Two methods to create a new layer. Arguments are required.

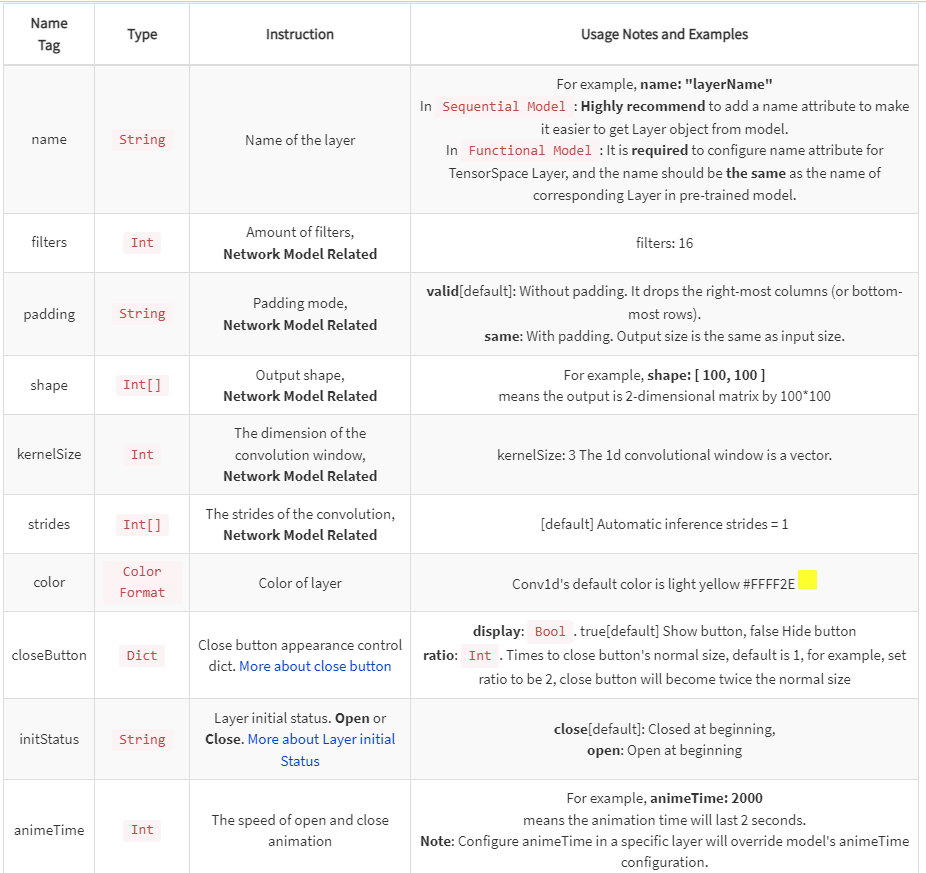
* 1. **Method 1**: Use **filters**, **kernelSize** and **strides**



* 1. **Method 2**: Use **shape**, the 2-dimension is the amount of filters



**🡺Arguments:**

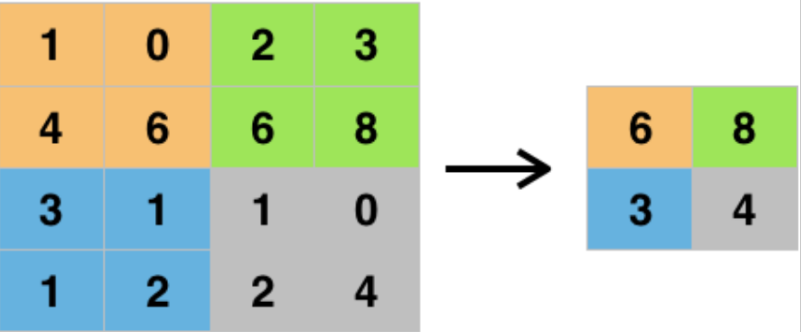


**🡺Example:**

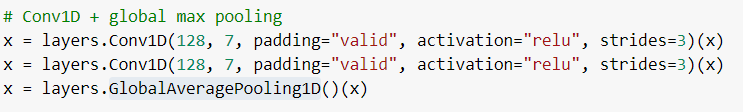


1. **What is Max Pooling?**

🡪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.



1. **Dense laye**r is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.



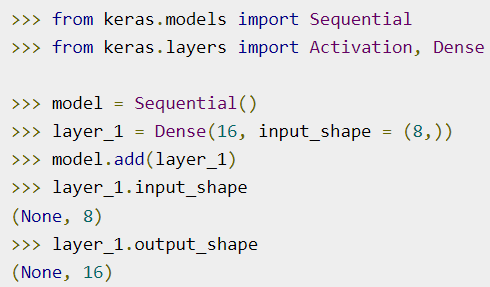
🡪where,

* + - **input** represents the input data
    - **kernel** represents the weight data
    - **dot** represents numpy dot product of all input and its corresponding output
    - **bias** represents a biased value used in machine learning to optimize the model
    - **activation** represents the activation function

🡪We can also get the input and output shape of the layer by help of the :

* 1. <layer-name>.input\_shape
  2. <layer-name>.output\_shape

🡪Example:



🡪The arguments supported by the **Dense layers** is as follows:

* **units** represents the number of units and it affects the output layer.
* **activation** represents the activation function
* **use\_bias** represents weather the layers use bias vectors
* **kernel\_initializer** represents the initializer to be used for kernel.
* **bias\_initializer** represents the initializer to be used for the bias vector.
* **kernel\_regularizer** represents the regularizer function to be applied to the kernel weights matrix.
* **bias\_regularizer** represents the regularizer function to be applied to the bias vector.
* **activity\_regularizer** represents the regularizer function tp be applied to the output of the layer.
* **kernel\_constraint** represent constraint function to be applied to the kernel weights matrix.
* **bias\_constraint** represent constraint function to be applied to the bias vector.