# 1, CNN.

- Convolutional Neural Networks (CNN, or ConvNet) are a type of multi-layer neural network that is meant to discern visual patterns from pixel images. In CNN, ‘convolution’ is referred to as the mathematical function. It’s a type of linear operation in which you can multiply two functions to create a third function that expresses how one function’s shape can be changed by the other. In simple terms, two images that are represented in the form of two matrices, are multiplied to provide an output that is used to extract information from the image.

- In this project, instead of using matrix of pixels as input to the network like in image processing, we will use word vector, which having embed size equal to 100 as the input channels number.

- We will use CNN 1d and CNN 3d with skipgram and cbow word embedding using the structure below:

+ **Convolutional Layer** (CONV): They are the foundation of CNN, and they are in charge of executing convolution operations. The Kernel/Filter is the component in this layer that performs the convolution operation (matrix). We reshaped embed matrix as *(batch\_size=32, embed\_size=100, 150)* for CNN 1d and as *(batch\_size=32, embed\_size=100, 6, 5, 5)* for CNN 3d. For both CNN 1d and 3d we will use 2x2 kernel, with ‘valid’ padding (no zero padding) and stride=1. Both CNN have output channels = 100.

+ **Fully Connected Layer** (FC): The fully connected layer (FC) works with a flattened input, which means that each input is coupled to every neuron. For CNN 1d we can use the output of the CONV layer to put it in a max method to get the max likely appear words vector and then put it as FC layer input, but for CNN 3d we have to reshape the output of the CONV layer and then put it as max method input. We will use 2 FC layers, the first one will use relu activation function and the second one will use softmax activation function for each word in the word vector classifying. For CNN 3d, the first layer will have 160 hidden neural and the second layer will have 64 hidden neural. For CNN 1d, the first layer will have 149 hidden neural and the second layer will have 64 hidden neural.

- For pretrained embed matrix, we will use the structure below:

+ **Convolutional Layer** (CONV): We reshaped embed matrix as *(batch\_size=32, embed\_size=100, 600)* for CNN 1d and as *(batch\_size=32, embed\_size=100, 6, 10, 10)* for CNN 3d. For both CNN 1d and 3d we will also use 2x2 kernel, with ‘valid’ padding(no zero padding) and stride=1. CNN 1d have ouput channels = 100, and for CNN 3d is 200.

+ **Fully Connected Layer** (FC): We will use 2 FC layers, the first one will use relu activation function and the second one will use softmax activation function for each word in the word vector classifying. For CNN 3d, the first layer will have 810 hidden neural and the second layer will have 64 hidden neural. For CNN 1d, the first layer will have 600 hidden neural and the second layer will have 64 hidden neural. The result of FC layer is 3 neural for sentiments levels (Positive, Neutral, Negative) classification.

# 2, RNN(LSTM)

- ****Recurrent Neural Networks(RNN)**** are a type of Neural Network where the output from the previous step is fed as input to the current step. One of the special features of RNN is that it has the ability to connect previous information with the current task. It makes RNN extremely useful in ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition,… However, in long-term dependencies, RNN learning capability is fall off due to derivative cancellation. We will use LSTM - a special RNN structure for our problem.

- LSTM makes a special architecture of RNN capable of learning long-term dependencies introduced by Hochreiter & Schmidhuber (1997). This architecture has been popular and widely used to this day. LSTM has proved to overcome many limitations of the previous RNN in terms of derivative cancellation. However, their structure is somewhat more complex, although the main idea of RNN is the replication of architectures in the form of chains.

- In our project, we will use both bidirectional and non-bidirectional LSTM with the structure below:

+ **LSTM layer:** Both bidirectional and non-bidirectional LSTM will have 1 recurrent layer, the number of expected features in the input is 100 (embed size), The number of features in the hidden state is  64.

+ **FC layer:** Before the output of LSTM layer is feed to the FC layer, it will be feed to a average pooling layer and a max pooling layer and then these two matrices will be concatenated into a matrix and then will be feed to the first FC layer having the hidden size equal to 64\*4 for bidirectional and 64\*2 for non-bidirectional with the selu activation function. And then the result will be feed to a drop out layer(drop\_rate=0.3) and then to the second FC layer with the input size equal 64\*2 for bidirectional and 64 for non-bidirectional with the softmax activation function with the output size equal to 3 for sentiment levels(Positive, Neutral, Negative) classification.

-