Assignment -12

Title: Implementation of CNN using Tensor flow/Pytorch

Problem Statement: Tensor Flow/Pytorch implementation of CNN

Objective: To implement CNN using Tensor flow/Pytorch

Software Required:

Tensor flow/ Pytorch

Theory:
A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm that 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 have the ability to learn these filters/characteristics.
How do convolutional neural networks work?

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- FuJJy-connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

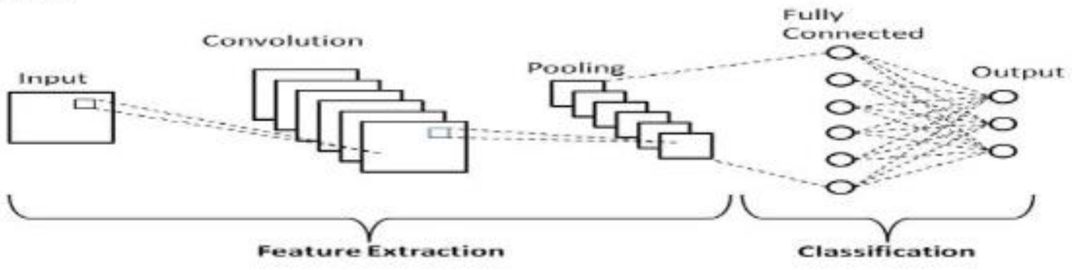
Advantages of Convolutional Neural Networks (CNNs):

- Good at detecting patterns and features in images, videos and audio signals.
- Robust to translation, rotation and scaling invariance.
- End-to-end training, no need for manual feature extraction.
- Can handle large amounts of data and achieve high accuracy.

Disadvantages of Convolutional Neural Networks (CNNs):

- Computationally expensive to train and require a lot of memory.
- Can be prone to overfitting if not enough data or proper regularization is used.
- Requires large amount of labeled data.
  Interpretability is limited, it's hard to understand what the network has learned.

Diagram:



Algorithm:

Step 1- Include the necessary modules for Tensor Flow and the data set modules, which are needed to compute the CNN model.

Step 2 - Declare a function called **run_cnn(),** which includes various parameters and optimization variables with declaration of data placeholders. These optimization variables will declare the training pattern.

Step 3 - In this step, we will declare the training data placeholders with input parameters. This is the flattened image data that is drawn from **mnist.train.nextbatch().**
We can reshape the tensor according to our requirements.

Step 4 - Now it is important to create some convolutional layers -

Step 5 - Let us flatten the output ready for the fully connected output stage. We can set up some weights and bias values for this layer, then activate with ReLU.

Step 6 - Another layer with specific softmax activations with the required optimizer defines the accuracy assessment, which makes the setup of initialization operator.

Step 7 - We should set up recording variables. This adds up a summary to store the accuracy of data.

Conclusion:

We have successfully implemented CNN using Tensor flow/Pytorch

Questions:

1. Explain Automation language translation in ANN.
2. Explain CNN Models.
3. Explain real time examples of pattern classification.
4. Explain real time examples of Neocognitron.
5. Explain real time examples of NET Talk.