Topic: Convolution Neural Network (CNN)

Instructions

Please share your answers filled inline in the word document. Submit Python code and R code files wherever applicable.

Please ensure you update all the details:

Name: Nukala Ayyappa Bharthwaj

Batch Id: DSWDMCOS 21012022

Topic: Convolution Neural Network.

1. Business Problem

1.1. Objective

1.2. Constraints (if any)

2. Work on each feature of the dataset to create a data dictionary as displayed in the below image:

Name of Feature	Description	Type	Relevance
ID	Customer ID	Quantitative, Nominal	Irrelevant, ID does not provide useful information

2.1 Make a table as shown above and provide information about the features such as its Data type and its relevance to the model building, if not relevant provide reasons and provide description of the feature.

Using Python code perform:

3. Data Pre-processing

3.1 Data Cleaning, Feature Engineering, etc.

3.2 Outlier Imputation if applicable

4. Model Building

5.1 Build the convolution neural network model

5.2 Train and Test the data

5.3 Briefly explain the model output in the documentation

5. Share the benefits/impact of the solution - how or in what way the business (client) gets benefit from the solution provided

6. Use Tensorflow for this assignment depending on your system configuration either tensorflowgpu or tensorflowcpu version.

Note:

The assignment should be submitted in the following format:

- **Python code**
- **Code Modularization should be maintained**
- **Documentation of the model building (elaborating on steps mentioned above)**

Problem Statement: -

1. Build a CNN model on CIFAR-10 dataset by applying few regularization techniques like drop out and data augmentation

Sol:

CNN model for CIFAR-10 data set is done using python and the same is attached.

2. Find out the differences between Convnet filter and the Maxpool layers

Sol:

Convolutional neural networks – CNNs or convnets for short – are at the heart of deep learning, emerging in recent years as the most prominent strain of neural networks in research. They have revolutionized computer vision, achieving state-of-the-art results in many fundamental tasks, as well as making strong progress in natural language processing, computer audition, reinforcement learning, and many other areas. Convnets have been widely deployed by tech companies for many of the new services and features we see today. They have numerous and diverse applications, including:

- detecting and labeling objects, locations, and people in images

- converting speech into text and synthesizing audio of natural sounds
- describing images and videos with natural language
- tracking roads and navigating around obstacles in autonomous vehicles
- analyzing videogame screens to guide autonomous agents playing them
- “hallucinating” images, sounds, and text with generative models

Although convnets have been around since the 1980s (at least in their current form) and have their roots in earlier neuroscience research, they’ve only recently achieved fame in the wider scientific community for a series of remarkable successes in important scientific problems across multiple domains. They extend neural networks primarily by introducing a new kind of layer, designed to improve the network’s ability to cope with variations in position, scale, and viewpoint. Additionally, they have become increasingly deep, containing upwards of dozens or even hundreds of layers, forming hierarchically compositional models of images, sounds, as well as game boards and other spatial data structures.

Because of their success at vision-oriented tasks, they have been adopted by creative technologists and interaction designers, allowing their installations not only to detect movement, but to actively identify, describe, and track objects in physical spaces. They were also the driving force behind Deepdream and style transfer, the neural applications which first caught the attention of new media artists.

The next few chapters will focus on convnets and their applications, with this one formulating them and how they work, the next one describing their properties, and subsequent chapters focusing on their creative and artistic applications.

Max Pooling is a downsampling strategy in Convolutional Neural Networks. Please see the following figure for a more comprehensive understanding. Here in the figure, we show the operation upon the pixel space. Alternatively we can do a similar operation on some other mathematical space. Also, one can change the operation of taking ‘Max’ to something else, say taking an ‘Average’ (This is what is done in Average Pooling).

Generally, for pedagogical purposes, the depiction of max pooling is made for non overlapping regions. This sometimes leads to a conjecture that max pooling is usually performed without overlaps. However, in reality, this notion is mostly not followed. In almost all of the famous CNN architectures, max pooling has been performed with overlapping regions. [Kernel Size, Stride] - AlexNet = [3x3, 2]; GoogleNet = [3x3, 2] , [3x3, 1]; VGG_CNN_S = [3x3,3], [2x2,2]; VGG_CNN_M and variants = [3x3, 2]; VGG_CNN_F = [3x3, 2]. We have thus shown in the figure all max pooling variants across the famous CNN architectures ([3x3,3] is similar in nature to [2x2,2]).

One can Google these configurations or refer to deploy files in BVLC Caffe !!

The pooling overlaps are in fact necessary in CNNs. As was pointed out by Hinton that without overlaps, pooling operation may lose important information regarding the location of the object.

3. If the input of an image is 64x64x3 which has been convolved by 10 5x5 filters with stride 1 and padding 2.

a. How many activation maps are obtained?

Sol: The size of the output CNN layer is calculated as "*input_size-(filter_size-1)*" = (64)-(5-1)=60

b. What is the size of the activation maps?

Sol: *input_size + 2 * padding_size-(filter_size-1) = (64+2*2)-(5-1)=64*

c. How many parameters are calculated?

Sol: (n*m*k+1)*f = (64*60*64+1)*5

4. During training, I get into overfitting issues. What are the different techniques will you apply to overcome this issue and why?

Sol: Overfitting is the phenomenon where the model performs well on the training set and gives out less accuracy when tested on the test dataset. This is rectified by performing regularization techniques. Some of the reguralization techniques are listed below: