Image Captioning

Name - Avadhoot Jadhav
Roll Number - 210050027

28th June 2023

Module 1

Machine learning model system design involves the careful construction of a framework that facilitates the development, training, and testing of machine learning models. The system consists of:

1. Data preprocessing: Normalization/scaling of data, Removing of outliers, dealing with missing values, etc.

2. Feature extraction: Converting the given data in numerical type. for example, One hot encoding of categorical variables.

3. Model Selection: Based on the type of data given and task we have to choose appropriate machine learning algorithm. Some machine learning algorithm includes classification, linear regression, clustering, logistic regression, etc. Also we should choose appropriate model architecture, such as neural network, decision tree, random forests, etc.

4. Model Training: While training, we decide loss and accuracy functions. Based on the the loss and accuracy graphs we can check if model is overfitting or underfitting. These problems can be removed by hypertuning.

5. Model Evaluation:The performance of model is checked using well-defined accuracy metrics. Some of them are accuracy score, precision, recall, and F1-score. These are used to evaluate the model's predictive capabilities.

Additionally, regularization techniques plys a vital role in enhancing the model's performance and ensuring its applicability to the required task.

Module 2: Convolutional Neural Networks

The convolutional neural network is a type of artificial neural network that is used for image recognition and classification. We input the image matrix and then calculate the loss by using some appropriate loss function. Then we minimize this loss using backpropagation and gradient descent.

Convolution Layer

Convolutional Neutral Networks operate over volumes, and a Convolution Layer is the core building block of the network. It consists of numerous filters in the layer that act on a given input volume. We say the layer as convolutional as it is related to convolution of two signals, which includes elementwise multiplication and sum of a filter and the image, that is

$$f[x,y] * g[x,y] = \sum_{n1=-\infty}^{\infty} \sum_{n2=-\infty}^{\infty} f[n1,n2] \cdot g[x-n1, y-n2]$$

Consider an input volume of an image of size $W_1 \times H_1 \times D_1$ that a Conv layer accepts. The Conv layer has essentially 4 hyperparameters :

- Number of filters K
- Spatial extent of filters F - height and width dimension of filter
- Stride S - the amount by which filter slides over the image volume
- Amount of zero padding P - done in general to preserve size spatially

Each filter in the layer is of size $F \times F \times D_1$. Notice that the *Depth* dimension of input volume and filter are same, i.e. Filters always extend the full depth of the input volume. We **Convolve** the filter with the image, that is "slide over the image spatially, computing dot products". We get a number as a result of dot product between a small $F \times F \times D_1$ chunk of input image and filter (that is, a $F * F * D_1$ dimensional dot product + bias; mathematically, $w^T x + b$). After sliding a single filter, we get an Activation Map of some height (say H_2) & width (say W_2) depending on other factors, and depth as 1. Since we had K filters, after applying all of them, we get a K activation maps, each of size $W_2 \times H_2 \times 1$. We stack them together to get a "new image" of size $W_2 \times H_2 \times K$. Thus, the convolution layer has re-represented the initial image in terms of activations of filters on it.

$$W_1 \times H_1 \times D_1 \xrightarrow[\text{Convolution Layer}]{} W_2 \times H_2 \times K$$

Padding

Notice that applying a filter of small dimension compared to input volume size repeatedly shrinks the volume spatially and that too very fast, which is not good. We generally try to produce the output from a Convolutional layer to be of same spatial size for representation and convenience reasons.
Therefore, it is a common practice to **zero pad** the border so as to preserve the size. This simply means padding with zeroes at borders just to maintain size.
In general, it is common to see Conv layers with stride $S = 1$, filters of size $F \times F$, and zero-padding with $\frac{F-1}{2}$.
So, a convolution layer -

- produces volume of size $W_2 \times H_2 \times D_2$, where
  - $W_2 = \frac{W_1 - F + 2P}{S} + 1$
  - $H_2 = \frac{H_1 - F + 2P}{S} + 1$
  - $D_2 = K$
- introduces $F \cdot F \cdot D_1$ weights per filter, so a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases with parameter sharing
- creates the d^{th} depth slice of output volume as a result of valid convolution of d^{th} filter over input volume.
- uses common settings for convenience such as using powers of 2 for K, odd natural numbers for F and accordingly choosing S and P that fits.

It is a Neuron with local connectivity that takes x_i, w_i's from neighbouring axons, calculates their convolution with a function f, and transfers $f(\sum_i w_i x_i + b)$ to the output axon.
Each activation map produced is a $W1 \times H1$ sheet of neuron outputs -

1. Each connected to a small region in input
2. All of them share parameters

So, a "$F \times F$ filter" $\rightarrow$ "$F \times F$ receptive field for each neuron". For example, with a 5 filters of $5 \times 5 \times 3$ on a input volume of size $32 \times 32 \times 3$, the Conv layer consists of neurons arranged in a 3-D grid of size $28 \times 28 \times 5$. There will be 5 different neurons all looking at the same region in input volume.

Pooling Layer

A layer that makes the representations smaller and more manageable. It operates over each activation map independently. It accepts a volume of size $W_1 \times H_1 \times D_1$ as input. A pooling layer has 2 hyperparameters -

- the spacial extent of filters F
- the stride S

And it reduces the size by producing a volume of size $W_2 \times H_2 \times D_2$, where -

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1$$

$$D_2 = D_1$$

Note that it introduces zero parameters as it computes a fixed function of input and also, it is not common to use zero-padding for pooling layers.Generally, some common settings use $F = 2, S = 2$ or $F = 3, S = 2$.
There are different types of poolings. One of them used common is **Max Pooling** - take maximum of the elements with a filter and stride for obtaining a reduced size volume. There is Average pooling also similarly.

Activation Layer

Activation layers are used to increase the complexity of the model by adding non-linearity. Basically, given a matrix activation layer transform all elements of the matrix by using a function called the activation function. Some of the common activation functions used are ReLu, sigmoid, tanh, etc.

Fully Connected Layer (FC Layer)

It is another layer in CNN that contains neurons which connect to entire input volume, as in an ordinary Neural Network. It is the layer which is present at very end of CNN. This layer has neurons for computing scores and are fully connected.
Convolutional neural networks are used for image classification and recognition in computer vision as they are good at detecting patterns and features.

Module 3: Recurrent Neural Networks

Recurrent Neural Networks are used for sequential data. That is when the output of current data depends on previous data. On studying the implementation of RNN we understand that the backpropagation in RNN leads to long multiplication of gradients. This leads to the vanishing and exploding gradient problem in RNN. The exploding gradient problem is controlled by gradient clipping. The vanishing gradient problem can be solved using LSTM.

LSTM

Long Short-Term Memory Networks is a deep learning, sequential neural network that helps to remember information. It is a special type of Recurrent Neural Network which is capable of handling the vanishing gradient problem faced by traditional RNN.
Just like a simple RNN, an LSTM also has a hidden state where H_{t-1} represents the hidden state of the previous timestamp and H_t is the hidden state of the current timestamp. In addition to that, LSTM also has a cell state represented by C_{t-1} and C_t for the previous and current timestamps, respectively. Here the hidden state is known as short term memory, and the cell state is known as Long term memory. The architecture of LSTM consists of the following gates:

1. Forget Gate: Depending on the previous hidden state and current input forget gate decides how much of the previous Long term memory to remember. It gives a binary output which is then multiplied with previous long term memory to remember/ forget info.

2. Input gate: Input gate operates on the same signals as the forget gate, but here the objective is to decide which new information is going to enter the state of LSTM.

3. Output gate: Depends on both the current Long term memory and current short term memory.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} sigm \\ sigm \\ sigm \\ tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot tanh(c_t^l)$$

Preprocessing in NLP

Preprocessing of Text data includes following:

1. Removing punctuations

2. Removing URLs

3. Lower casing: Since we want same words with different cases to be considered same.

4. Tokenization: Splitting the text in smaller units.

5. Removing Stop words: Some words occur very frequntly in text. They do not add any value to analysis and hence are removed.
6. Stemming: Removing the suffix of the words.
7. Lemmatization: Removing the suffix of the word such that it does not lose it's meaning.