Velagapudi Ramakrishna Siddhartha Engineering College

M.Tech Degree Examination, November,2021

II Semester DataScience

**19ITDS2014B- Deep Learning**

Max. Marks :60

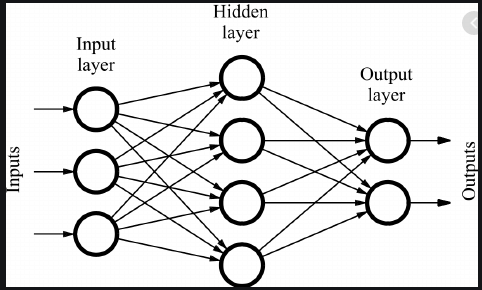
Answer one question from each Unit

**UNIT-I**

1.a. Interpret the significance of activation functions in feed forward neural network. **7M**

Brief explanation of Feed forward neural networks -**4M**

Need for activation function -**3M**



* Input Layer: The bottom layer of the network pulls in the input data.
* Output Layer: The top layer of neurons (output nodes) computes our final answer.
* Hidden layer: The middle layer of neurons in between input and output layers are called the *hidden layers.*

A neuron takes in some number of inputs, x1, x2, . . . ,xn, each of which is multiplied by a specific weight, w1,w2, . . . ,wn. They are summed together to produce the logit.logit also includes a bias, which is a constant. The logit is then passed through a function f which is called an activation function to produce the output y = f (z). The purpose of an activation function is to add some kind of non-linear property to the function.

Without the activation functions, the neural network could perform only linear mappings from inputs **x**to the outputs **y**. So, the feed forward network with only linear neurons can be expressed as a network with no hidden layers. In order to learn complex relationships, we need neurons which use activation functions that introduce nonlinearities in their computations.

b. Compare and contrast stochastic and mini batch gradient descent approaches. **8M**

Discussion on what is gradient descent 3M

Discussion on stochastic and mini batch similarities, differences 5M

* During training, we will give the neural net a large number of training examples and iteratively modify the weights to minimize the errors we make on the training examples.
* we want to minimize the square error over all of the training examples that we encounter

the closer *E is to 0, the better our model is.*



* our goal will be to select our parameter vector *θ (the values for all the weights* in our model) such that *E is as close to 0 as possible*

To find the values of the weights that minimizes the error function, gradient descent algorithm will be used.

* *batch gradient descent.*

*The* idea behind batch gradient descent is that we use our entire dataset to compute the error surface and then follow the gradient to take the path of steepest descent.

* In Stochastic Gradient Descent (sometimes also referred to as *iterative* or *on-line* gradient descent) we update the weights after each training sample

Disadvantages of batch and stochastic

Batch- It is difficult for datasets that don't fit in memory. Batch gradient descent also doesn't allow us to update our model *online*, i.e. with new examples on-the-fly

Stochastic- SGD will keep overshooting, complicates convergence to the exact minimum and make gradient descent take a significant amount of time

Mini batch gradient descent

* Mini-batch gradient descent takes the best of both worlds and performs an update for every mini-batch of n training examples. Instead of summing over all the examples in the dataset, we sum over the examples in the current mini batch.

**(OR)**

2.a. Outline the limitations of traditional computer programs 7M

Explaining traditional computer programs and where they are suitable 3M

Explanation of limitations 4M

Traditional computer programs are designed to be very good at two things:

1. performing arithmetic really fast and

2) explicitly following a list of instructions

So, Traditional computer programs can do heavy financial number crunching. But it is difficult to write a programs like automatically read someone’s handwriting .

* *For example, It difficult to write* rules that can differentiate one digit from another



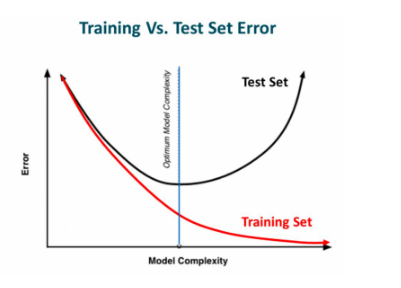
It is also difficult to write programs for Object recognition, Speech comprehension, automated translation etc. using traditional computer programming.

b. Demonstrate overfitting in Deep neural networks 8M

Explanation of overfitting 4M

Techniques for overfitting 4M

* Overfitting refers to the phenomenon where a neural network models the training data very well but fails when it sees new data from the same problem domain.



One of the main reasons for the network to overfit is if the size of the training dataset is small. Over fitting can also be caused by noise in the training data that the neural network picks up during training and learns it as an underlying concept of the data. Avoiding over fitting can improve model’s performance.

Methods to avoid overfitting include

* *regularization (L2 and L1)*
* *Max norm constraints*
* *Dropout*

**UNIT-II**

3.a. Explain convolution operation with a suitable example on a 5x5 image with a 3x3 kernel with a stride 2 and zero padding 8M

Any example with Convolution 4M

Stride 2M

Zero padding 2M

b.Make use of convolution neural networks to recognize hand written digits. 7M

CNN layers explanation 3M

Any cnn model building 4M

A CNN model consists of three primary layers: Convolutional Layer, Pooling layer(s), and fully connected layer.

(1) Convolutional Layer: This layer extracts high-level input features from input data and passes those features to the next layer in the form of feature maps.

(2) Pooling Layer: It is used to reduce the dimensions of data by applying pooling on the feature map to generate new feature maps with reduced dimensions. PL takes either maximum or average in the old feature map within a given stride.

3) Fully-Connected Layer: Finally, the task of classification is done by the FC layer. Probability scores are calculated for each class label by a popular activation function called the softmax function.

Loading the Dataset

Processing the Dataset

Creating and Training a CNN

Making Predictions

**(OR)**

4.a. Justify the importance of filters and feature maps in convolution networks 7M

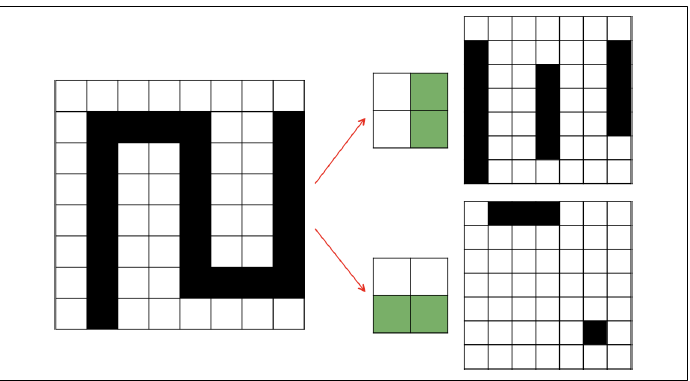
Explanation of filterand feature map-- 3M

Explanation of filters and feature maps in CNN-4M

The three elements that enter into the convolution operation

Input image  
Feature detector  
Feature map

A filter is essentially a feature detector



* For example, to detect vertical lines, we would use the feature detector on the top, slide it across the entirety of the image, and at every step check if we have a match.
* This result is our *feature map,* This operation is called a convolution. We take a filter and we multiply it over the entire area of an input image.
* If we express this operation as neurons in a network, layers of neurons in a feed forward neural net represent either the original image or a feature map.
* Filters represent combinations of connections (one such combination is highlighted in Figure) that get replicated across the entirety of the input.
* The output layer is the feature map generated by this filter. A neuron in the feature map is activated if the filter contributing to its activity detected an appropriate feature at the position in the previous layer
* if kth feature map in layer m is mk , corresponding filter is denoted by the values of its weights W, assuming bias bk ,we can mathematically express the feature map as follows:



b. Examine the accelerating training with batch normalization with a suitable example 8M

what is batch normalization , explanation 6M

Example graph -2M

Training of feed-forward and convolutional neural networks can be accelerated using a technique

called*batch normalization*

Normalization of image inputs helps out the training process by making it more robust to variations. Batch normalization takes this a step further by normalizing inputs to every layer in our neural network. Specifically, we modify the architecture of our network to include operations that:

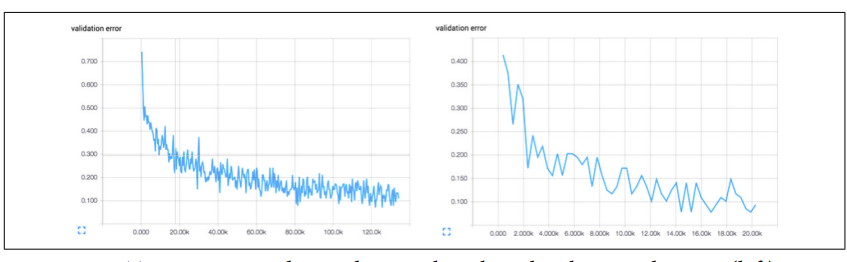
1. Grab the vector of logits incoming to a layer before they pass through the nonlinearity

2. Normalize each component of the vector of logits across all examples of the minibatch

by subtracting the mean and dividing by the standard

3. Given normalized inputs **x**̂, use an affine transform to restore representational

power with two vectors of (trainable) parameters: γ**x**̂ + β

****

Without batch normalization, cracking the 90% accuracy threshold may require over 80,000 minibatches. On the other hand, with batch normalization, crossing the same threshold only requires slightly over 14,000 minibatches.

**UNIT-III**

5.a. Explain the motivation behind autoencoders 5M

Explanation 4M

Block diagram 1M

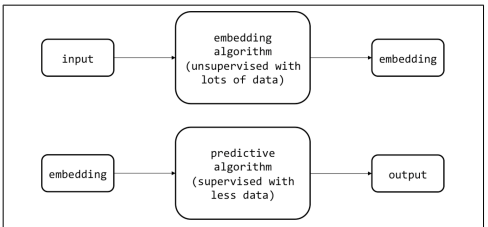
In Deep Learning, the larger our input vector, the larger our model. Large models with lots of parameters are expressive, but they’re also increasingly data hungry.

This means that without sufficiently large volumes of training data, we will likely overfit. Convolutional architectures help us cope with the curse of dimensionality by reducing the number of parameters in our models without necessarily diminishing expressiveness.

But, convolutional networks still require large amounts of labeled training data. And for many problems, labeled data is scarce and expensive to generate.

Autoencoders are effective learning models in situations where labeled data is scarce but wild, unlabeled data is plentiful.

They learn embeddings, or low-dimensional representations, in an unsupervised fashion They use generated embeddings to solve learning problems using smaller models that require less data.



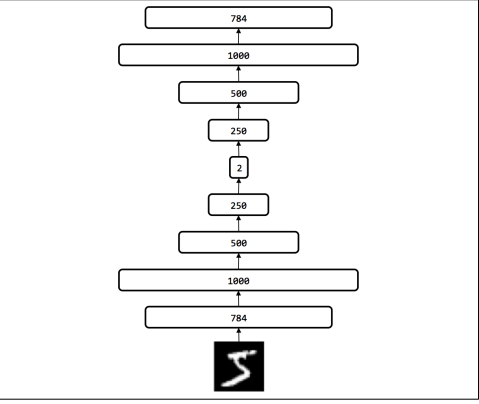
b. Develop an autoencoder in tensorflow using MNSIT 10M

Experimental setup for autoencoder diagram -3M

Encoder, decoder explanation - 2M

Code -5M

Experimental setup for dimensionality reduction with autoencoder



**(OR)**

6.a. Discuss the importance of denoisy autoencoders to improve the ability of autoencoder to generate embeddings 7M

Human visions ability to identify noisy image –1M

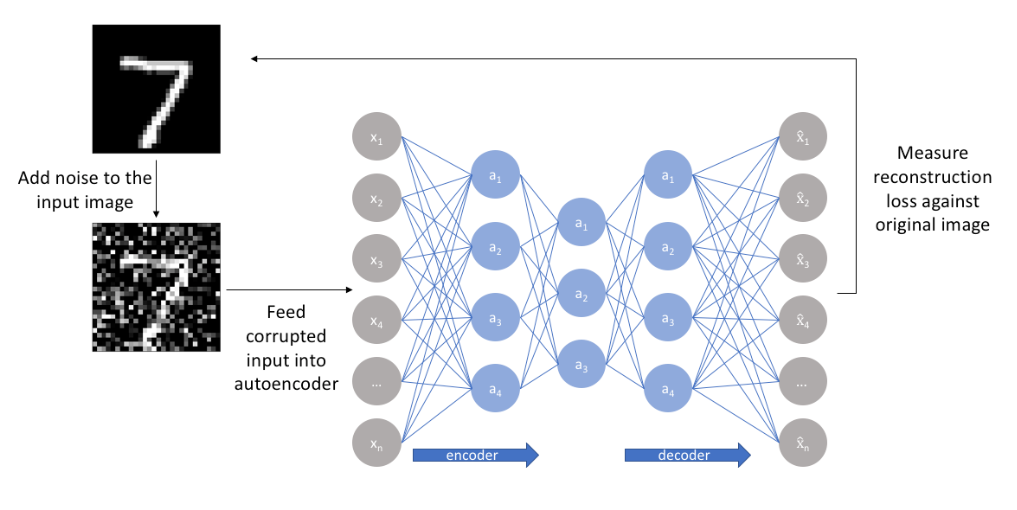
Denoisy autoencoder diagram -2M

Explanation -4M

Even though a corrupted version of a digit hits our retina, our brain is still able to reproduce the set of activations (i.e., the code or embedding) that we normally would use to represent the image of that digit. Denoisy autoencoders are also made to generate embeddings that are resistant to noise . We corrupt some fixed percentage of the pixels in the input image by setting them to zero. Given an original input X, let’s call the corrupted version C(X).

The input to the encoder network is the corrupted C( X ) instead of X. In other words, the autoencoder is forced to learn a Embedding and Representation Learning code for each input that is resistant to the corruption mechanism and is able to interpolate through the missing information to recreate the original, uncorrupted image.

.



**UNIT- IV**

7.a.Explain unfolding computational graphs in detail 7M

Dynamic system -3M

Unfolding any rnn -4M

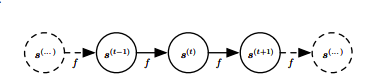
A computational graph is a way to formalize the structure of a set of computations, such as those involved in mapping inputs and parameters to outputs and loss.

For example, consider the classical form of a dynamical system:



where s (t) is called the state of the system. This equation is recurrent because the definition of s at time t refers back to the same definition at time t − 1.

For a finite number of time steps τ , the graph can be unfolded by applying the definition τ − 1 times



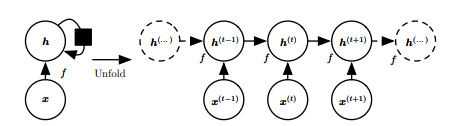
A dynamical system driven by an external signal x (t) can be represented as



where the state s(t) contains information about the whole past sequence. Many recurrent neural networks use the following equation or a similar equation to define the values of their hidden units. To indicate that the state is the hidden units of the network, we now rewrite above equation using the variable h to represent the state,



This equation can be illustrated with:



This diagram represents a recurrent network with no outputs. The black square in Circuit diagram indicates a delay of a single time step. The same network seen as an unfolded computational graph, where each node is now associated with one particular time instance.

b. Elaborate on optimization for long term dependencies 8M

Exploding and vanishing gradients - 2M

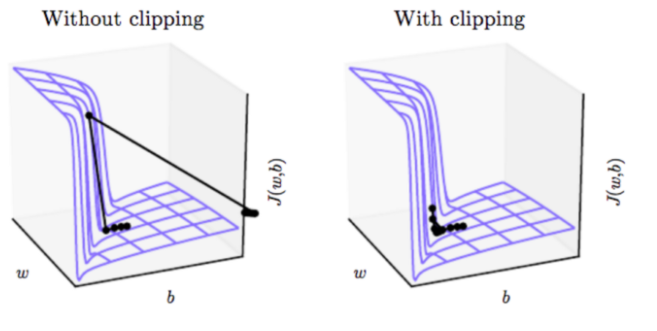
Optimization techniques - 6M

RNN architecturecauses two problems in training: exploding gradients and vanishing gradients.

Gradient clipping is a technique that tackles exploding gradients. The idea of gradient clipping is very simple: If the gradient gets too large, we rescale it to keep it small. More precisely, if ‖**g**‖ ≥ c, then

**g** ↤ c · **g**/‖**g**‖

where c is a hyperparameter, **g**is the gradient, and ‖**g**‖ is the norm of **g**. Since **g**/‖**g**‖ is a unit vector, after rescaling the new **g**will have norm c.



One approach to tackle vanishing gradients is LSTM and other self-loops and gating mechanisms

Another idea is to regularize or constrain the parameters so as to encourage “information flow by using a regularizer.

**(OR)**

8.a.Identify the importance of recurrent neural networks in sequence modeling 5M

Explanation 5M

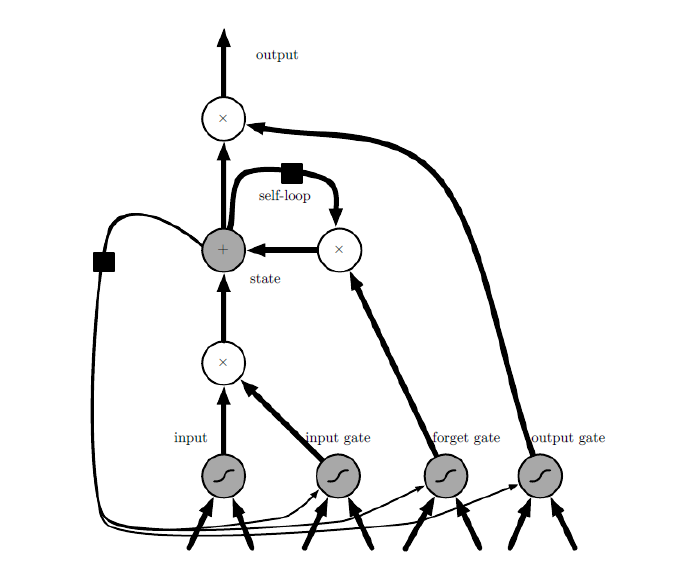
RNN is a neural network that is specialized for processing a sequence of values *x*(1)*, . . . , x*(*τ*) .

* RNN takes advantage of one of the early ideas found in machine learning and statistical models sharing parameters across different parts of a model.
* Parameter sharing makes it possible to extend and apply the model to examples of different forms (different lengths, here) and generalize across them
* Such sharing is particularly important when a specific piece of information can occur at multiple positions within the sequence. For example, consider the two sentences “I went to Nepal in 2009” and “In 2009, I went to Nepal.
* A traditional fully connected feed forward network would have separate parameters for each input feature, so it would need to learn all the rules of the language separately at each position in the sentence.
* RNNs operate on a sequence that contains vectors x (t) with the time step index t ranging from 1 to τ and shares the same weights across several time steps.
* the network may have connections that go backward in time, provided that the entire sequence is observed before it is provided to the network

b. Discuss about LSTM and GRU in detail 10M

Diagram- 3M

Explanation of LSTM gates -7M



A typical LSTM network is comprised of different memory blocks called cells  
There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called gates.

### Input Gate: The input gate is responsible for the addition of information to the cell state.

### ­­­­­Forget Gate: A forget gate is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things

Output gate: The job of selecting useful information from the current cell state and showing it out as an output is done via the output gate.