Atharva Nevase

Assignment: 1

- **Aim:** Study of Deep-Learning packages: Tensorflow, Keras and PyTorch. Document the distinct features functionality of the package.

- **Objective:** Introduction to various deep learning tools & how to use them.

- **Software used:** Windows/Linux, Tensorflow, Keras

- **Theory:**

Tensorflow:-

Tensorflow is an open-source software library released in 2015 by Google to make it easier for developers to design, build & train deep learning models.

Tensorflow originates as internal library that Google developers use to build models in-house, & we expect additional functionality to be added to the open-source version as it is tested & vetted in the internal flavour.

- On a high-level, Tensorflow is a python library that allows users to express arbitary computation as a graph of data flows. Nodes in this graph represent mathematical operations, whereas edges represent data that is communicated from one node to another.

Data in tensor-flow is represented as tensors, which are multi-dimensional arrays.

Keras:

Keras is an open-source deep learning
for Python. It has been developed by an
Keras is based on minimal structure that
clean & easy way to create deep learning m
based on tensorflow. Keras is designed to
deep-learning models. Well. Keras is an option
choice for deep learning applications.

Features:

Keras leverages various optimization techniques
high-level neural network API easier & m
1) Consistent, simple & extensible API
2) Minimal structure - easy to achieve the result.
3) Supports multiplatform & backends.

Benefits:

Keras is highly powerful & dynamic framewo
1. Larger community support.
2. Easy for test
3. Keras NN are written in python which m
Keras support both convolution & recurrent.

Pytorch:

i) Pytorch is an optimized tensorflow library
learning using GPU & CPU. Pytorch is optimize
based on python & torch.
ii) Pytorch is favored over other Deep Learning

- Main 2 features of PyTorch are:

1. Tensor Computation (similar to numpy) with strong GPU acceleration support.
2. Automatic differentiation for creating & training deep neural network.

- Common PyTorch Modules:

Modules are used to represent neural networks:

1) Autograd:

The autograd module in PyTorch's automatic differentiation engine that helps to compute the gradient in the forward pass in quick time. Auto-grad generate a directed on-cyclic graph where the leaves are the input sen tensors while the roots are the output tensors.

2) Optim:

The optim module is the package with pre-written algorithms for optimizers that can be used to build neural networks.

- Conclusion:

We learnt about the various deep learning training tools.

Atharva Nevase

Assignment: 02

- Aim: Implementing Feedforward neural network with Keras & Tensorflow.

a) Import the necessary packages.
b) Load training & testing data.
c) Define network architecture
d) Train model using SGD.
e) Evaluate the network
f) Plot the training loss and accuracy.

- Objective: To learn how to develop a feed forward network & how to optimize it for better performance

- Infrastructure: Computer / Laptop

- Software used: Jupyter Notebook / Google Colab.

- Theory:

Feed forward Neural Network:

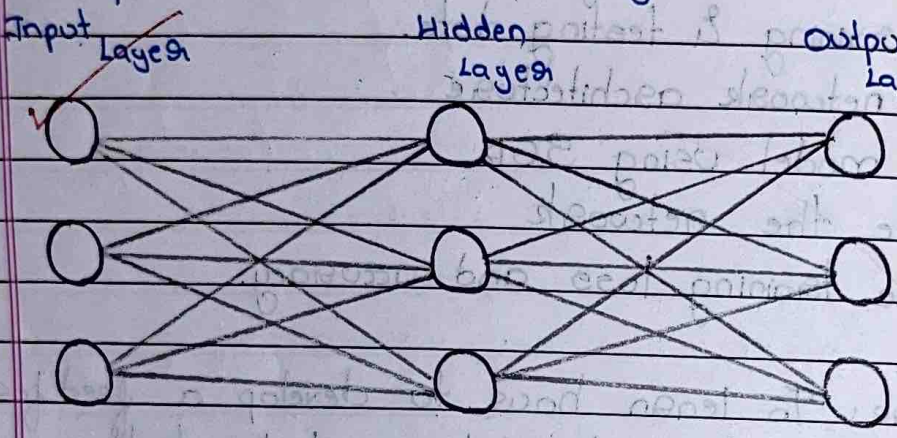
Deep feed forward often called feed-forward, neural network or multi-layer perceptron's. The goal of the feed-forward network is to approximate some function. There are no feedback connection in which outputs of the model are feedback into itself

- Structure of feed-forward neural network:

1) Input layer: It is where the user accepts the layer for the neural network.

2) Hidden layer: This is where the computation for the predictions are done.

3) Output layer: The output from the Hidden la is provided at output layer.



- The nodes are connected with the help of The edges are represented as W_{ij} where represent the node where the edge end The nodes compute the output for the nex layer by summation of the product of the input associated with the node edge, which is do an activation function to decide wheth should fire or not for the input.

- SGD:

Stochastic Gradient Descent & It variants a probably the most used optimization algorithm machine learning in general & deep learning general particular. A crucial parameters for SGD algorithm is learning rate.

- MNIST:

The MNIST data set of handwritten digits has a training set of 70,000 examples and each row of the matrix corresponds to a 28 x 28 image. The unique values of the responsible variable y range from 0 to 9.

- Implementation:

1. Import the necessary libraries.
2. Load the dataset from libraries or from outside
3. Build the feed-forward neural network using keras.
4. Train the model with the dataset & use SGD as optimizer.
5. Plot the loss and accuracy function.

- Conclusion:

We developed a feed-forward neural network hand written digit recognisation.

Page No.
Date

Deep Learning
Assignment : 3

. Aim : Build the image classification model by dividing the model into 4 stages:
a) Loading the image data
b) Defining model's architecture
c) Training the model
d) Estimate model's performance

. Objectives : To learn about CNN & how to develop a CNN for image recognition.

. Infrastructure : Computer / Laptop

. Software Used : Jupyter Notebook / Google Colab.

. Theory :

1. CNN : Convolution networks, also known as convolution neural networks or CNN's are a special kind of neural networks for processing data that has a known grid-like topology.

The name convolutional neural networks indicate that the network employs a mathematical operation called convolution. Convolution is a special kind of linear operation. Convolution operations are simply neural networks that use convolution in place of general matrix multiplication in atleast one of their layers.

Page No.
Date

- There are 2 main parts of a CNN architec
1) A convolution tool seperates and identifies t various features of that image for analysis i process called as Feature Extraction
2. This CNN model of feature extraction aim the number of features present in a data New features which summarizes the existing contained in an original set of features.

- Convolution Layers:
i) There are three types of layers that make CNN, which are convolution layers, pooling l fully connected layers.
ii) In addition to these three layers there are types of important parameters. which are dea layer activation function.

2) Pooling layers: The primary aim of this laye is to decrease the size of the convolved fe map to reduce the computational costs. This is performed by decreasing the connections layers & independently operates on each featu map.

3) Fully-connected layer:
The fully-connected layer consists of weights along with neuron & is used to connect t neurons between 2 different layers. These l are usually placed before the output layer form the last few layers of an CNN archi

Page No.
Date

4) Activation function :

They are used to learn & approximate any continous & complex relationship between variables of network. In simple words, it decides which information of the model should fire in forward direction & which ones should not at end of network.

· Implementation :

1. Load necessary libraries
2. Import dataset from respective library.
3. Design neural network architecture & mention the number of layers, etc.
4) To train the model with imported dataset.
5) Evaluate performance of model.

· Conclusion :

We learnt how to build & train a CNN to ideal images.

Page No
Date

Deep Learning
Assignment : 4

- Aim: Use autoencoders to implement anamoly detection. Build the model by using :-
a) Import required libraries.
b) Upload dataset
c) Encode convert it into latent representation.
d) Decoder network convert back to original input.
e) Compile the model with optimizer.

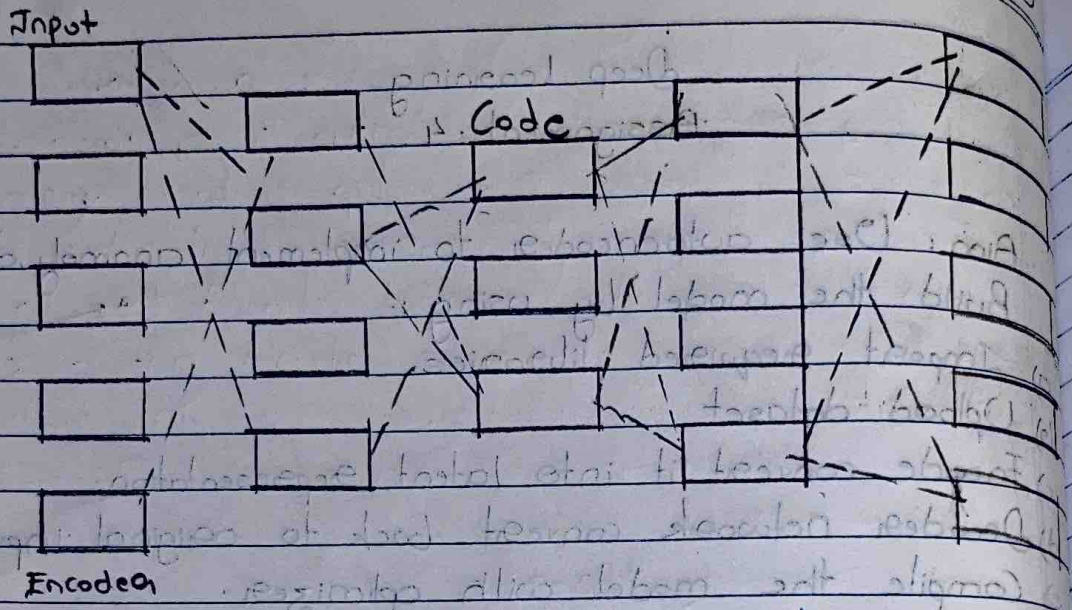
- Objective: To learn about autoencoders & developing autoencoders for anamoly detection.

- Infrastructure: Computer / Laptop

- Software used: Jupyter Notebooks

- Theory:

Autoencoders: Autoencoders are ANN capable of learning efficient representation of input data, called codings without any supervision. These codings have typically a much lower dimensionality than the input data making autoencoders useful for dimensionality reduction & compression.
Autoencoders act as powerful feature detectors can be used for unsupervised learning of deep neural networks. Similarly, they are capable of randomly generating new data. For ex. you can train an autoencoder on a couple of faces & it would be able to generate new faces.



- An autoencoder has 3 components.

1) Encoder: It compresses the input into a lo representation. It compaises the input in a red dimension.

2) Code: The compressed input which is fed to reconstructing the original input later.

3) Decoder: It decodes the encoded output into Code, back to original input.

The layers between encoder & decoder is as the Bottleneck. This is a well-defined to decide which aspects of data should rele infomation.

- Parameters used for training autoencoders a

1) Code size: It is the number of nodes in coding layer. Smaller size results in mor compression of results.

2) Number of layers: Autoencoders can be deep as you like, you need to decide how much layers autoencoders could have.

3) Number of nodes per layer: No. of nodes per layer decreases with each subsequent layer of encoder & increases with each layer in decoder.

4) Loss Function: We either use MSE (Mean Squared Error) & Binary cross entropy as loss function.

. Implementation:

1) Load necessary libraries.

2) Import the dataset from respective library.

3) Shape the data as per your need.

4) Encode the input data to latent representations.

5) Decode the output of encoder to convert it back to original input.

6) Use models with optimizers, loss & Evaluation Met

. Conclusion:

We learnt how to detect anamdy using auto encoders.

Page No.
Date

Assignment: 5
Deep Learning

- Aim: Implement the CBOW stages:
a) Data preparation
b) Generate training data
c) Train model
d) Output

- Objective: To learn and understand CBOW.

- Infrastructure: Computer / Laptop / VM

- Software Used: Jupyter Notebook / Google Colab, Tensorflow, Keras.

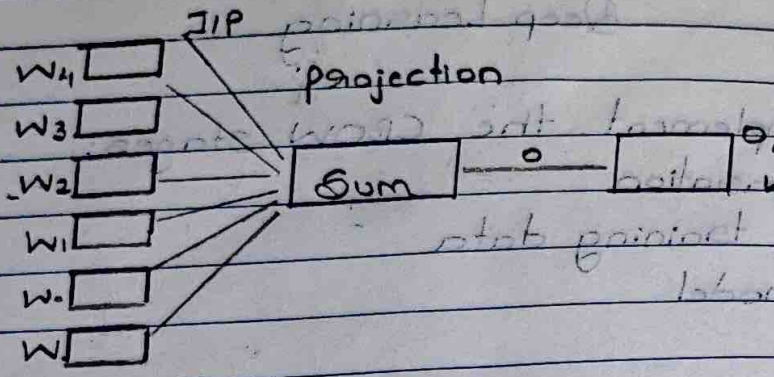
- Theory:

The CBOW model tries to understand the context of the words and takes time as i/p it then tries to predict words that are contextually accurate. Let us consider an example for understanding this consider the sentence, it's a pleasant day and the word 'pleasant' goes up to neural network.

We are trying to predict word 'day' here. We will use one hot encoding for i/p words and measure the error rates with one hot encoded target word. Doing this will help us predict output based on the word with least error.

Page No.
Date

Model Architecture:



The CBOW model architecture is as shown above. The model tries to predict the target words by trying to understand the content of surrounding words considering the same sentence as above 'It's a placement day'. The model converts this sentence into word pairs inform. With these words pairs, the model tries to predict the target words considered content words.

If we have 4 content words which are used for predicting one target words with i/p layer will be in the form of four 1×W ip vectors. Finally 1×N output from hidden layer enters the sum layer where element wise summation is performed and o/p is obtained.

Implementation of CBOW model:

1) Import the libraries and read dataset.
2) For the implementation
3) Generate a function that create window sizes and pairs of target words.

Page No.
Date

. Build neural network on sample data.

. Conclusion:

We saw a CBOW model and how it works. This can be used for text recognition, speech to text conversion, etc.

Page No.
Date

Assignment-6
Deep Learning

- Aim: Object detection using Transfer Learning CNN architecture.

a) Load pre-trained CNN model
b) Freeze parameters
c) Add custom classifier
d) Train classifier layers on training data available.
e) Fine tune hyper parameters and unfreeze more layers as needed.

- Objective: To load pre-trained model and improve performance by transfer learning.

- Infrastructure: Computer / Laptop / VM

- Software Used: Juyptar Notebook / Google Colab, Tensorflow, Keras.

- Theory:

Refers to process where a model trained on one problem is used in some way on a second related problem. Transfer learning has benefit of decreasing the training time for neural network and results in low generalization error. The weights reused layer maybe used as starting point for training process and adapted.

- How to use pre-trained models?

- classifier: directly classifies new images.
- standalone feature extractor = pre-process image
- Integrated feature extractor = Layers are of pre-trained model are trained in concert with n model.
- Weight initialization: some portion of model is ini- ated into new model and layers are trained with new model.

- It may not be clear to which usage of pre-tr- ed model may yield the best results on your new computer vision task.

- Ways to fine-tune the model:

1) Feature extraction: We can use pre-trained mo as feature extraction mechanism. We can remo the output layer and then use entire network as fixed feature extractor for new data-set.

2) Use architecture of pre-trained model: What we can do is we keep the weights randomly and train the model, while we re-train high lay

- Building Deep-Learning Based Object Detection mo Training a performing deep learning model for object detection takes a lot data and computin power to facilitate the dev. we can use trans

learning by fining tuning models. We can create a central dictionary to store these configuration parameters, including setting up different paths, installing relevant libraries, and downloading pre-trained models.

Conclusion:

We conclude this that how to develop a model for specific application with help of transfer learning architecture in Deep Learning.