Assignment - 4

Use autoencoders to implement anomaly detection

Aim:

Use autoencoders to implement anomaly detection
Build the model using -

a) Import required libraries
b) Upload / access dataset
c) Encoder converts it into latent representation
d) Decoder networks convert it back to original input
e) Compile models with optimizer, loss and evaluation metrics.

Theory:

Auto encoder

Auto encoder is an unsupervised Artificial Neural Network that attempts to encode the data by compressing it into the lower dimensions and then decoding that the data to reconstruct the original input. The bottleneck layer holds the compressed representation of the input data. The number of hidden units in the code is called code size.

Applications of auto encoders.
↳ Dimensionality reduction
↳ Anomaly detection
↳ Image denoising
↳ Image compression

~~Dimentio~~
Dimensionality reduction

There methods are based on the assumption that dimension of data is artificially inflated and its itrinsic dimension is much lower.

Anomaly detection

It is the process of finding abnormalities in data. Abnormal data is defined as the ones that deviate significantly from general behaviour of the data.

Image denoising

Today, auto encoders are very good at denoising of images. When image get corrupted or there is a bit of noise in it, we call this image as a noisy image To obtain proper information about content of image, we want image denoising.

Image compression

for this, it is pretty difficult for an auto encoder to do better than basic algorithms, and by being only specific for a particular type of images, we can prove this wrong. Thus, this data specific property of auto encoders makes it impractical for compression of real world data.

Feature Extraction

Encoding part of auto encoders helps to learn important hidden features present in the input data, in the proces to reduce the reconstruction error

Image Generation

There is a type of auto encoder, named Variational Auto encoder, this type of auto encoders are Generative model used to generate images.

The use is -

↳ Generate new characters

↳ Generate fake human images.

Assignment - 5

Implement the continuous Bag of Words (CBOW)

Aim:

ⓐ Data preparation
ⓑ Generate training data
ⓒ Train model
ⓓ Output

Theory:

Word2Vec is considered one of the biggest breakthroughs in the development of natural language processing.

Each unique word in your data is assigned to a vector and these vectors vary in dimensions depending on the length of the word.

The model has two different architectures to create word embeddings –

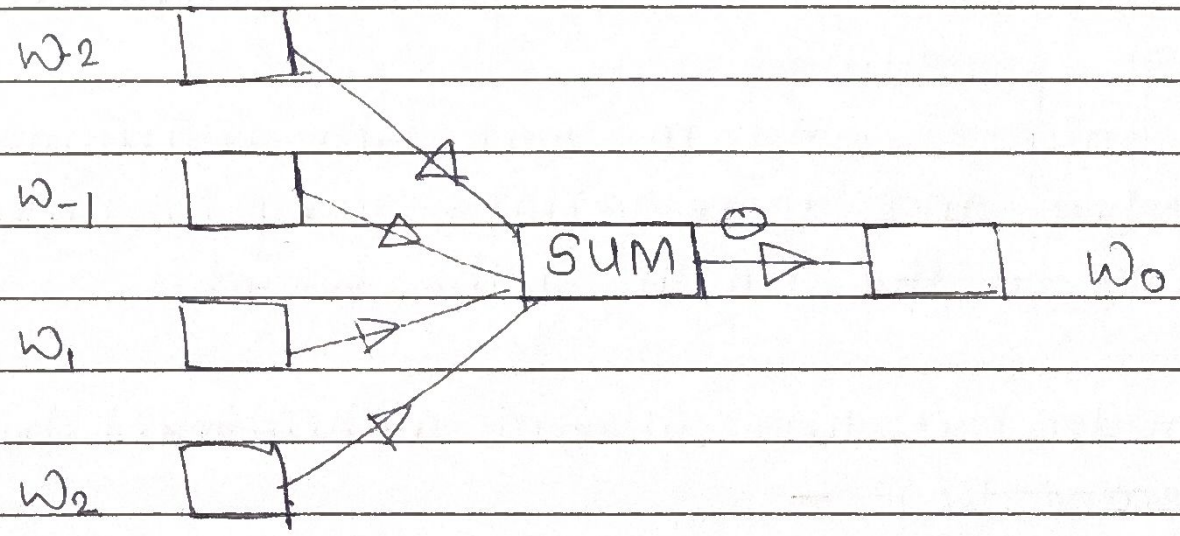
1. Continuous bag of words (CBOW)
2. Skip gram model

The CBOW model tries to understand the context of words and takes this as input. It then tries to predict words that are contextually accurate.

for eg – in sentence "It is a pleasant day", if we set the window size to 2, the word pairs become : ([it, a], is), ([is, pleasant], a), ([a, day], pleasant)

Each context word is represented as a one hot encoded vector, passed through hidden layer and summed element wise before the final activation predicts target word.

The output shows the words that are most similar to word virus along with sequence or degree of similarity. The words like symptoms and incubation are contextually very accurate with the word virus which proves CBOW model succesfully understands the context of data.



input projection output

Assignment - 6
Object Detection

Aim:

Object detection using transfer learning of CNN architectures

Objectives:

a. Load in pretrained CNN model trained on a large dataset
b. Freeze parameters in model's lower convolutional layers.
c. Add custom classifier
d. Train classifier layer
e. fine tune hyperparameters and unfreeze more layers as needed.

Theory:

(a.) load in pre trained CNN model trained on a large dataset

Artificial neural networks such as RNN, GANs and autoencoders.

The learning rate is used in training on the pretrained models.

learning rate gradually increases from 0.00002 to 0.002 at ~~each~~ epoch 100 and linear decreases until reaching 0.0002 at epoch 800.

Size of the data set ↑	train the model from scratch	Fine tune the pretrained pretrain pretrained model
	Fine tune the lower layers of the pretrained model	Fine tune the output of dense layers of model

Data Similarity ——————→

(b) Freeze parameters in model's lower convolutional layer.

Freezing a layer prevents its weights from being modified. This technique is often used in transfer learning where the base model

setting trainable = false for freezing the for f layer model.add (conv2D(64, (3,3), trainable == false)).

(c) Add custom classifier with several layer of trainable parameters of model

We have compelete information about the date of development and trainable parameters counts of If you're not familiar with the MINIST dataset, its a collection of 0-9 digits as images. The images are grayscale and thus each image can be represented within an input shape of 28 x 28 x 1.

(d) Train classifier layer on training data available for task

↳ The ~~weights~~ weights in those layers to be re initialized base.model.trainable = false

↳ Add new trainable layers that will turn old features into predictions on the new dataset

↳ This is important because pre trained model is loaded without the final output layer.

↳ A final dense layer, with units corresponding to the number of output expected by your model.

(e) Fine tune hyper parameters and unfreeze more layers as needed.

~~The optimal hyperparameters, let use~~

↳ learning rate, batch size, momentum and high weight decay. These hyperparmeters act as

the learning rate is high, then training may not converge or even diverge.

Steps for hyperparameter tuning—

1. Select the right type of model

2. Review the list of parameters of the model build the HP space

3. finding the method for searching the hyperperameter space

4. Applying cross validation scheme approach

5. Assess the model score to evaluate the model.