Savitribai Phule Pune University, Pune

Final Year Information Technology (2019 Course)

414447: Lab Practice IV

Teaching Scheme:	Credit Scheme:	Examination Scheme:
Practical (PR):02 hrs/week	01 credits	PR: 25 Marks TW: 25 Marks

Prerequisites: Python programming language

Course Objectives:

The objective of the course is

1. To be able to formulate deep learning problems corresponding to different applications.
2. To be able to apply deep learning algorithms to solve problems of moderate complexity.
3. To apply the algorithms to a real-world problem, optimize the models learned and report on the expected accuracy that can be achieved by applying the models.

Course Outcomes:

On completion of the course, students will be able to-

CO1. Learn and Use various Deep Learning tools and packages.

CO2. Build and train a deep Neural Network models for use in various applications.

CO3. Apply Deep Learning techniques like CNN, RNN Auto encoders to solve real word Problems.

CO4. Evaluate the performance of the model build using Deep Learning.

Guidelines for Instructor's Manual

The faculty member should prepare the laboratory manual for all the experiments, and it should be made available to students and laboratory instructor/assistant

Guidelines for Student's Lab Journal

1. Students should submit term work in the form of a handwritten journal based on a specified list of assignments.
2. Practical Examination will be based on the term work.
3. Candidate is expected to know the theory involved in the experiment.
4. The practical examination should be conducted if and only if the journal of the candidate is complete in all respects.

Guidelines for Lab /TW Assessment

1. Examiners will assess the term work based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of implemented assignment, attendance etc.
2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. Appropriate knowledge of usage of software and hardware related to the respective laboratory should be checked by the concerned faculty member.

Guidelines for Laboratory Conduction

As a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers of the program in a journal may be avoided. There must be hand-written write-ups for every assignment in the journal. The DVD/CD containing student's programs should be attached to the journal by every student and the same to be maintained by the department/lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Practical Examination

1. During practical assessment, maximum weightage should be given to satisfactory implementation of the problem statement.
2. Student's understanding of the fundamentals, effective and efficient implementation can be evaluated by asking relevant questions based on implementation of experiments he/she has carried out.

List of Laboratory Assignments

Mapping of course outcomes for Group A assignments: CO1, CO2,CO3,CO4

1. Study of Deep learning Packages: Tensorflow, Keras, Theano and PyTorch. Document the distinct features and functionality of the packages.

Note: Use a suitable dataset for the implementation of following assignments.

2. Implementing Feedforward neural networks with Keras and TensorFlow
    a. Import the necessary packages
    b. Load the training and testing data (MNIST/CIFAR10)
    c. Define the network architecture using Keras
    d. Train the model using SGD
    e. Evaluate the network
    f. Plot the training loss and accuracy

3. Build the image classification model by dividing the model into following 4 stages:
    a. Loading and preprocessing the image data
    b. Defining the model's architecture
    c. Training the model
    d. Estimating the model's performance

4. Use Autoencoder to implement anomaly detection. Build the model by using:
    a. Import required libraries
    b. Upload / access the dataset
    c. Encoder converts it into latent representation
    d. Decoder networks convert it back to the original input
    e. Compile the models with Optimizer, Loss, and Evaluation Metrics

5. Implement the Continuous Bag of Words (CBOW) Model. Stages can be:
    a. Data preparation
    b. Generate training data
    c. Train model
    d. Output

6. Object detection using Transfer Learning of CNN architectures

a. Load in a pre-trained CNN model trained on a large dataset b. Freeze parameters (weights) in model's lower convolutional layers c. Add custom classifier with several layers of trainable parameters to model d. Train classifier layers on training data available for task e. Fine-tune hyper parameters and unfreeze more layers as needed
Reference Books:
1. Hands-On Deep Learning Algorithms with Python: Master Deep Learning Algorithms with Extensive Math by Implementing Them Using TensorFlow 2. Python Deep Learning, 2nd Edition by Ivan Vasilev , Daniel Slater , GianmarioSpacagna,, Peter Roelants, Valentino Zocca 3. Natural Language Processing with Python Quick Start Guide by Mirant Kasliwal
Virtual Laboratory:
SPIT's Virtual Labs for AI and Deep Learning: https://vlab.spit.ac.in/ai/

Aim :-

Study of Deep learning packages.

Objective :-

① Tensorflow
② Keras
③ theuno
④ PyTourch.

1. *Tensorflow :-

Tensor is a generalization of vector and Matrices of potentially higher dimension array. of data with varing dimensions and rank's that are fed as input to the neural Network are called tensor.

- even they appear in multidimensional arrays.
- Whe the data stored in tensor and fed into the neural network, the output we get is as show below.

- There are some terms associated with tensors that we need to familiarize ourselves with:

• Dimension :-

Dimension is the size of the array elements. Below you can take a look at various type of dimensions:

Data in the form of Array is Fed as i/p to the N/w.

3	5	4
2	6	7
8	1	3
2	5	9
1	4	2

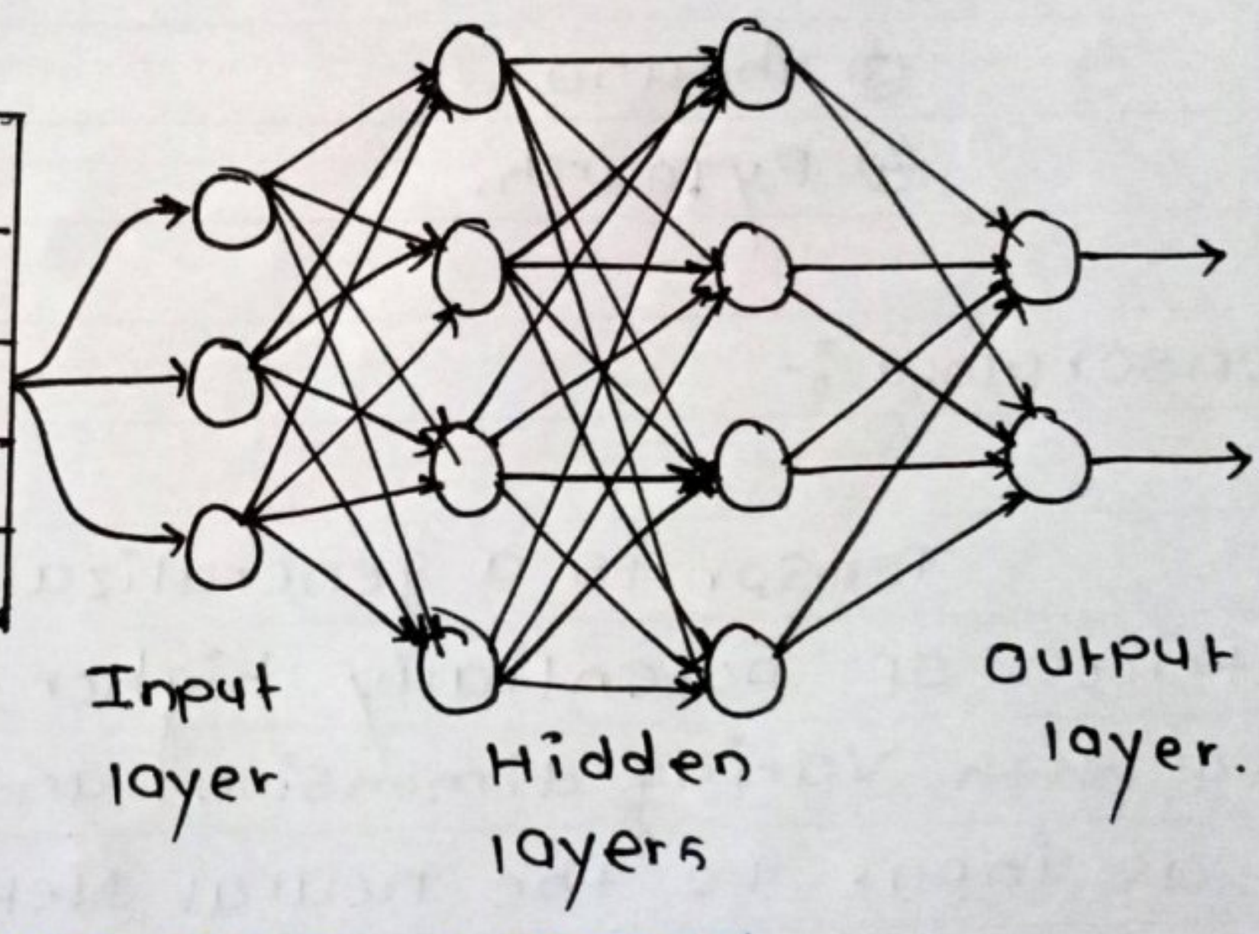
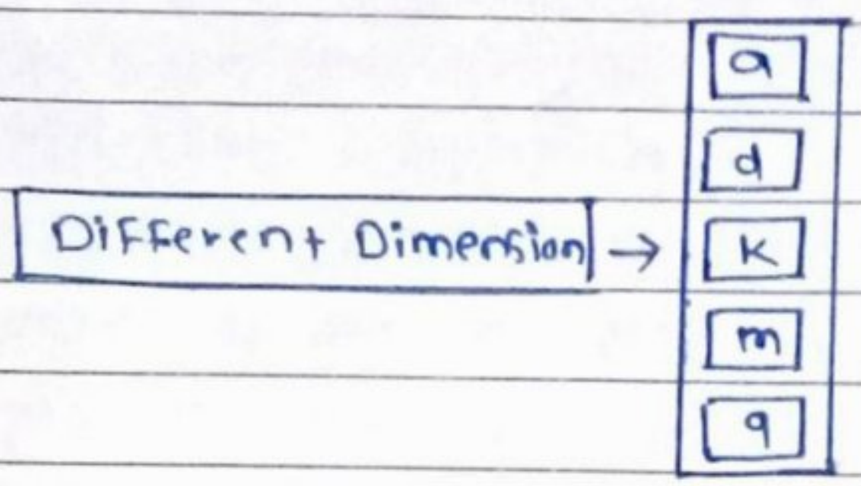
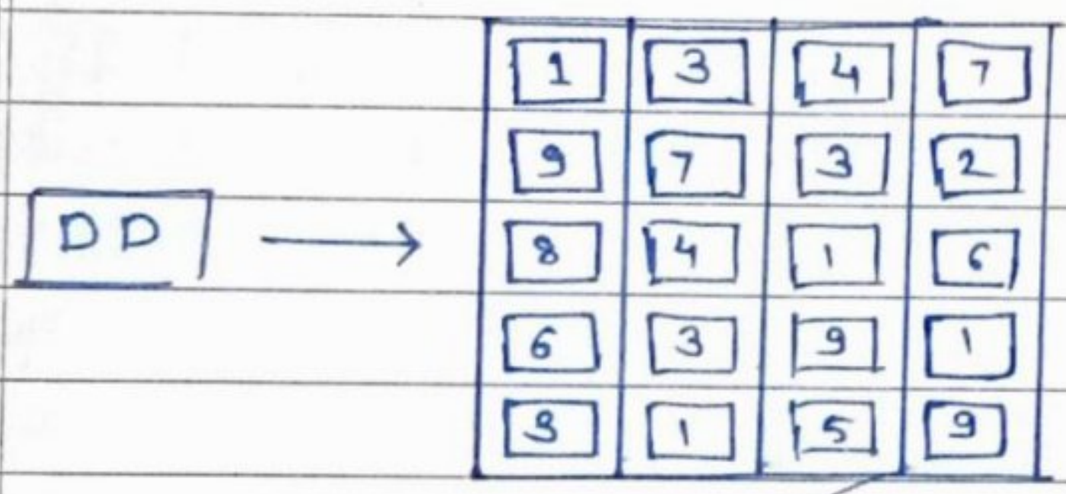


Fig: Neural Network Tensor.





- Algorithm supported by Tensorflow

1. classification - tf.estimator.LinearClassifier
2. Lenear regration - tf.estimator.LinearRegressor.
3. Boosted tree classification - tf.estimator.Boosted Trees Classifier.
4. Booster tree regression - tf.estimator, DNN Linear CombinedClassifier.
5. Deeplearning classification. tf.estimator.DNNClasifier
6. Deeplearning classifi Wipe & Deep - tf.estimator.DNN lener combined classifier

- Tensorflow however data can be stored and manipulated using three different programming element

1. Constant's
2. Variables.
3. Placeholder's.

Example :-

"Hello World" program in Tensorflow.

```
hello = tf.constant ('Hello')
type (hello)
tensorflow. python. framework.Ops.Tensor
World = tf. constant ('world')
result = hello + World.
result
<tf. Tensor 'add:0' Shape=() dtype: string
type (result)
tensorflow.python.framework.ops.Tensor.
With tf.Session() as sess:
    result = sess.run (hello + world)
result
b 'HelloWorld'
```

2. Kera's :-

Kera's is a deep learning API Written in Python, running on top of the machine learning platform transorflow.

- Kera's is
  - Simple
  - Flexible
  - powerful.

- Tensorflow2 to is an end to end opensource Machine learning platform.
  You can think of it as an infrastructure layer for differentiable programming.

- Here is the sequential model:

```
from tensorflow.keras.models import sequential
model = sequential()
```

- Evaluate your test loss and metrics in one line

```
loss_and_matrics = model.evaluate(x_test, batch_size
                                  = 128).
```

- generate prediction new data.

```
classes = model.predict(x_test, batch_size=128)
```

3. Theano :-

- Theano is a python library for fast numerical computation that can be run on the CPU or GPU.

- It is a key foundation library for Deep learning in python that you can use directly to create Deep learning models or Wrapper libararies that greatly simplify the process.

Example :-

```
import theno
from theano import tensor
a = tensor.dscalar()
b = tensor.dscalar()
c = a+b
f = theano.function([a,b],c)
assert 4.0 == f(1.5,2.5)
```

4. PyTorch :-

Pytorch is the premier open-source deep learning framework developed and maintained by facebook.

- Pytorch deep learning model life-cycle

- A model has life cycle and this very simple knowledge provides the backbone for

both modeling a dataset and understanding the pytorch API.

- The five steps in life-cycle :
  1. Prepare the data.
  2. Define the model
  3. Train the model
  4. Evaluate the model.
  5. Make prediction.

Step 1: prepare the data :-
- the 1st step is to load and prepare your data.
- Neural Network models requere numerical input data and numerical output data.
- For example the constractor of your dataset object can load your data file (e.g a csv file) You can override the_lens() function that can be used to get the lenght of dataset.

Step 2: Define the model :-
- the 2nd step is to define model.
- Activation functions can also be define as layer. such as ReLUs, Softmax, and Sigmoid
- Common examples include the xavier and He weight initialization sceme

```
1....
2 xavier_uniform_(self.layer.weight)
```

step 4. evaluate the model.

- once the model is fit, it can be evaluated on the test dataset
- this can be achived by using the Dataloader for the test dataset and collecting the predictions for the test set the comparing the prediction expected.
- perform matrics.

```
1. ...
2. For i, (inputs, targets) in enumerate
3.     # evdluate the model on test set. (test_dl)
4. yhat = model(inputs)
5. ...
```

step 5; Make predictions

Conclusion :-

Wee have study deeplearning

Tensorflow, keras, Theano, pytourch

10/10 24/8/2022

Experiment No-02.

Aim :-

implementing fo feedforword neural Network with Keras and Tensorflow

Objective :-

a. import the necessary package's
b. Load the training and testing data (MNIST).
c. Define the network architecture using keras.
d. Train the model using SGD.
E Evaluate the Network.
F. Plot the training less and accuracy.

Theory :-

- To Demostrate how you can implement ~~feedforword multi-layer network and apploy them to the~~ MNIST ~~and~~ CIFAR-10 Datasets. Feed Simple neural Networks using the keras library.
- Obtain baseline standard Nural Network which we will e later compare to convolution Nural Network

MNIST :-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
get_ipython().run_line_magic('matplotlib',
'inline')
```

MNIST :-

- MNIST stand "Modified National Institute of Standard and Technology"
- It is a Dataset of 70,000 handwrite image each image is of 20 x 20 pixels.
  i.e about 784 features.
  eachfeatyrs. each represent only one pixel intensiy i.e from 0 (White) to 255 (black).
- these Database is further divided into 60,000 training and 10,000 testing image.

```
# import dataset and split into train and test data
mnist = tf, keres, dataset.mnist
(X_train, y_tarin), (X_test, y_test) = mnist.load_
 data ().
```

```
# shape of training dataset 60,000, images
  having 28x28 size x_train.shape.
```

Network Architecture using keras :-

- Tensorflow is an opensource set of libraries for creating and working with Neural Network, such as those used in ML. and DL project.

Creating the model :-

- The ReLU function is one of the most popular activation function's.
- It Stand for "reflected linear unit" mathamatically this function is define as

- $y = max(0, x)$ The ReLU function returns "0" if the input is Nagative and is linear if.
- The input is positive.
- the softmax is another activation function.

```
model = keras.sequantial([
keras.layers.flatten(input_shape = (28,28)),
keras.layers.Dense(128, Activation 'relu'),
keras, layers, Dense(10, activation = 'softmax')
])
```

Graph # Train the model

```
history= model.fit(x_train, y_train, validation.data
(x_test, y-test), epochs =10)
```

Evaluate the model / Network:-

```
test_loss, test_acc = model.evaluate(x_test, y_test)
print("loss = %., 3f" % test_loss)
print("Accuracy = % 3f"%. test_acc)
```

- prediction of the data
- plot graph for Accuracy and loss

```
get_ipython().run_line_magic('pinfo2', 'history.
history')
history.history.keys()
```

graph representing the model's accuracy

in [23]:

```
plt.plot (history.history ['loss'] )
plt.plot(history.history ['val_loss'])
plt.title ('model loss')
plt.xlabel ('epoch')
plt.legend (['Train', 'validation'], loc = 'upper left')
plt.show()
```

graph represent's the model's loss

In [24]:

```
plt.plot (history.history['accuracy'] )
plt.plot(history.history ['val_accuracy' ])
plt.plot (history.history['loss']).
plt.title ['Training loss and accuracy')
plt.ylabel ('accuracy loss')
plt xlabel ('epoch')
plt-legend ( ['accuracy', 'val_accuracy', 'loss', 'val_
                loss']).
plt.show()
```

Conclusion :-

With above code we can see. that throughput with epochs, Our model accuracy increase and our model loss decreasses, that is good since our model gains confidence with its prediction.

Date :-

Experiment No- 03

Page No.
Date

Aim :-

Build the Image classification model by dividing the model into following 4 stages:.

a. Loading and preprocessing the data image data.
b. Training the model c. define the model.
d. Estimating the model performance

Objectives :-

a. Loading and processing the image data.
b. Training the model.
c. Define the model architecture
d. estimating the model performance.

Theory :-

a. Loading and processing the image data :-

1. First, you will use high-level keras preprocessing utilities (such as tf.keras.utils.image_dataset_from_directory) and layers such as tf.keras.layers. Rescaling to read a directory of image on disk.

2. Next, you will write your own input pipline from scratch using tf.data.

3. fainally you will download a dataset from the large catalog available in Tensorflow dataset.

```
import numpy as np
import os
import PIL
import PIL.Image
import tensorflow as tf
import tensorflow_dataset as tfds.

print (tf.__version__)
```

** Output **

2.9.1

Thes Exper Expirement use a dataset of several thousend photos of flowers.
each directory contains images of that type of flower
Example:

```
roses = list (data_dir.glob ('roses/*'))
PIL.Image.open (str(roses[0]))
```

Load data Let's load thes image off to disk using the helpful
tf.keras.utils.image_dataset_from_directory
utility.
You can find the class names in the class_name attribute on these dataset

b. Training the model.

1. Training a model simply means learning (determining) good values for all the weights. and the bias from labeled example
2. in supervised learning a machine learning algorithm build the model by examining many examples and attempting to find a model that minimizes loss;

3. There are 6 steps:-

Step 1 : Setup a google cloud Account

Step 2 : Create a project

Step 3 : Deploy Deep learning virtual machine.

Step 4 : Access Jupyter Notebook GUI

Step 5 : Add GPUs to virtual machine

Step 6 : Change virtual machine Configuration.

c. Define the model architecture :-

- some of the architectures like google net has involved over a period of time with the introduction

Digram:-

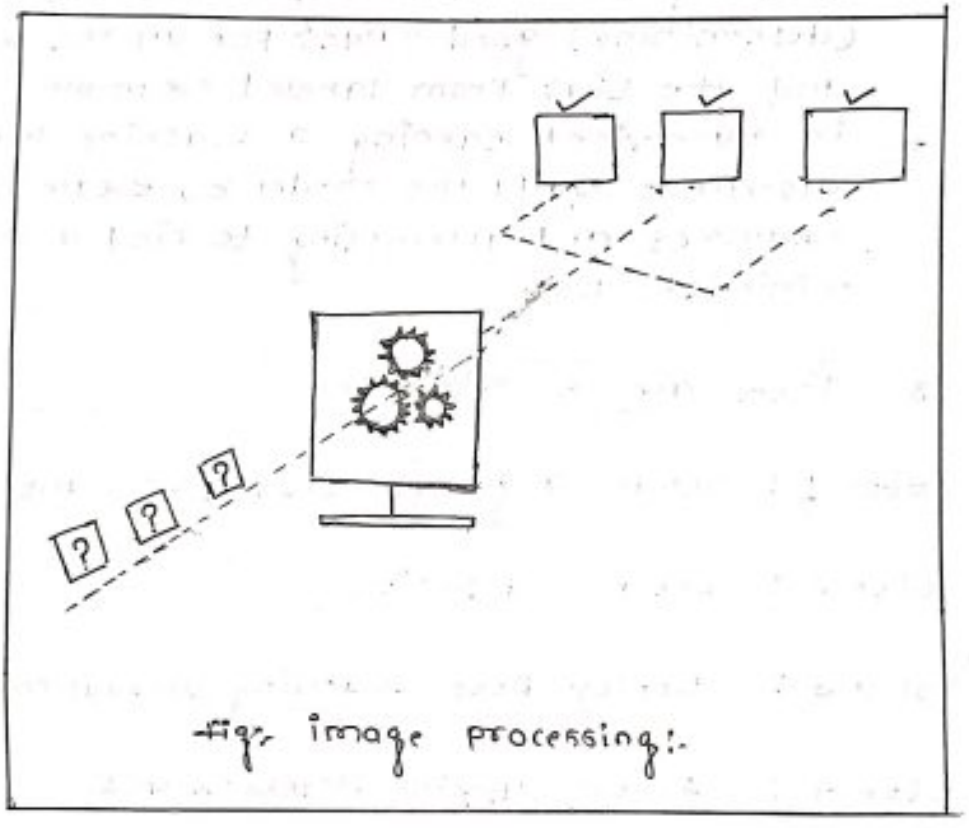


fig:- image processing:-

This blog post assumes that the user has knowledge on Deep learning especially on topics like convolutions, pooling, activation function and back propagation

following architectures.

1. AlexNet (2012)
2. ZFNet (2013)
3. Googlenet (2014)
4. ResNet (2015).
6. ResNeXt (2016)
7. DenseNet (2016)
8. SENet (2018).

Some architectures like google Net has involve over a period of time with the introduction of residual functions in ResNet paper

d. estimating the model performance.

- evaluating a model is a major part of building an effective machine learning model
- most frequent classification evaluation metric that we use should be 'Accuracy'
- The 4 aspects as as shown below :-
  1. the confusion matrix for a 2-class classification problem.
  2. the key classification matrics: Accuracy, Recall

Table of Confusion Matrix:-

		Actual value	
		Positive	Negative
Pridicted value.	positive	TP (True positive)	FP (false positive)
	Nagative	FN (False Nagative)	TN (True Nagative)

Possible classification:

Outcome :- TP, FP, FN, TN.

Page No.
Date

precision, and F1-score

3. The difference between Recall and precision in specific cases.
4. Decision Threshold and Receiver operating Characteristic (ROC) Curve

- To illustrate we can see how the 4 classificar matrics are calculate (TP, FP, FN, TN).
- Confusion matrix table.

- the equation of tpr and fpr:-

1. tpr :-

$$\text{true positive rate} = \frac{\text{true positives}}{\text{true positives} + \text{false negative}}$$

2. fpr :-

$$\text{false positive rate} = \frac{\text{false positive}}{\text{false positive} + \text{true nagatives.}}$$

Conclusion :-

We have studied by image classification model by dividing those 4 stages we are studied. loading and processing image data training, define estemating the model image, performance.

10/10 28/9/2022

4. Use Auto encoder to implement anomaly detection. Build the model by using:

a. Import required libraries

b. Upload / access the dataset

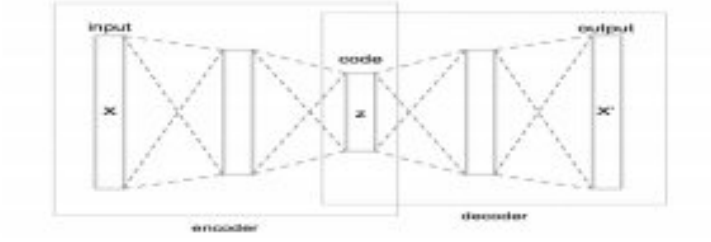
c. Encoder converts it into latent representation

d. Decoder networks convert it back to the original input

e. Compile the models with Optimizer, Loss, and Evaluation Metrics

Auto Encoder

Auto Encoder is an unsupervised Artificial Neural Network that attempts to encode the data by compressing it into the lower dimensions (bottleneck layer or code) and then decoding the data to reconstruct the original input. The bottleneck layer (or code) 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
- Image generation

Dimensionality reduction

Dimension reduction methods are based on the assumption that dimension of data is artificially inflated and its intrinsic dimension is much lower. As we increase the number of layers in an auto encoder, the size of the hidden layer will have to decrease. If the size of the hidden layer becomes smaller than the intrinsic dimension of the data then it will result in loss of information.

Anomaly Detection

Anomaly detection is the process of finding abnormalities in data. Abnormal data is defined as the ones that deviate significantly from the general behavior of the data. Some of the applications

of anomaly detection include fraud detection, fault detection, and intrusion detection. Anomaly Detection is also referred to as outlier detection.

Image Denoising

Today, Auto encoders are very good at denoising of images.

What happens when rain drops are on our window glass?

Of course, we can't get a clear image of **"What is behind the scene?"** Here rain drops can be seen as noise. So,

When our image gets corrupted or there is a bit of noise in it, we call this image as a *noisy image*.

To obtain proper information about the content of image, we want **Image Denoising**.

We define our auto encoder to remove (if not all) most of the noise of the image.

Image Compression

Usually, Auto encoders are really not good for data compression. **For Image Compression, it is pretty difficult for an auto encoder to do better than basic algorithms, like JPEG** and by being only specific for a particular type of images, we can prove this statement wrong. Thus, this data-specific property of auto encoders makes it impractical for compression of real-world data. One can only use them for data on which they were trained, and therefore, generalization requires a lot of data.

Feature Extraction

Encoding part of Auto encoders helps to learn important hidden features present in the input data, in the process to reduce the reconstruction error. During encoding, a new set of combination of original features is generated.

Image Generation

There is a type of Auto encoder, named Variational Auto encoder (VAE), this type of auto encoders are **Generative Model**, used to generate images.

The idea is that given input images like images of face or scenery, the system will generate similar images. The use is to:

Generate new characters of animation

Generate fake human images

Conclusion:

Thus we have studied Auto encoder for anomaly detection.

5. Implement the Continuous Bag of Words (CBOW) Model. Stages can be:

a. Data preparation

b. Generate training data

c. Train model

d. Output

Theory:

Word2vec is considered one of the biggest breakthroughs in the development of natural language processing. The reason behind this is because it is easy to understand and use. Word2vec is basically a word embedding technique that is used to convert the words in the dataset to vectors so that the machine understands. 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 word2vec model has two different architectures to create the word embeddings. They are:

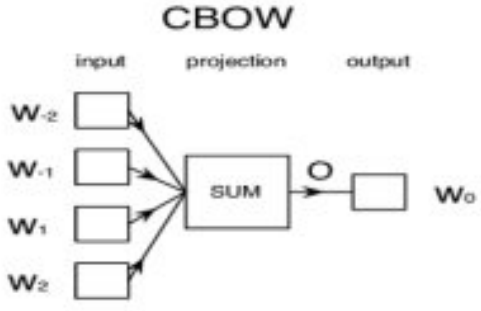
1. Continuous bag of words(CBOW)
2. Skip-gram model

In this article, we will learn about what CBOW is, the model architecture and the implementation of a CBOW model on a custom dataset.

What is the CBOW Model?

The CBOW model tries to understand the context of the words and takes this as input. It then tries to predict words that are contextually accurate. Let us consider an example for understanding this. Consider the sentence: 'It is a pleasant day' and the word 'pleasant' goes as input to the neural network. We are trying to predict the word 'day' here. We will use the one-hot encoding for the input words and measure the error rates with the one-hot encoded target word. Doing this will help us predict the output based on the word with least error.

The Model Architecture



The CBOW model architecture is as shown above. The model tries to predict the target word by trying to understand the context of the surrounding words. Consider the same sentence as above, 'It is a pleasant day'. The model converts this sentence into word pairs in the form (contextword, targetword). The user will have to set the window size. If the window for the context word is 2 then the word pairs would look like this: ([it, a], is), ([is, pleasant], a),([a, day], pleasant). With these word pairs, the model tries to predict the target word considered the context words.

If we have 4 context words used for predicting one target word the input layer will be in the form of four 1XW input vectors. These input vectors will be passed to the hidden layer where it is multiplied by a WXN matrix. Finally, the 1XN output from the hidden layer enters the sum layer where an element-wise summation is performed on the vectors before a final activation is performed and the output is obtained.

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

Conclusion

In the above article, we saw what a CBOW model is and how it works. We also implemented the model on a custom dataset and got good output. The purpose here was to give you a high-level idea of what word embeddings are and how CBOW is useful. These can be used for text recognition, speech to text conversion etc.

Page No.
Date

Experiment No - 06.

Aim :-

Object detection using transfer learning of CNN architectures.

Objective :-

a. Load in pre-trained CNN model trained on a large dataset.
b. Freeze parameters (weights) in model's lower convolutional layers.
c. Add custom classifier with several layer of trainable parameters of model.
d. Train classifier layer on training data available for task
e. fine-tune hyper parameters and unfreeze more layers as needed.

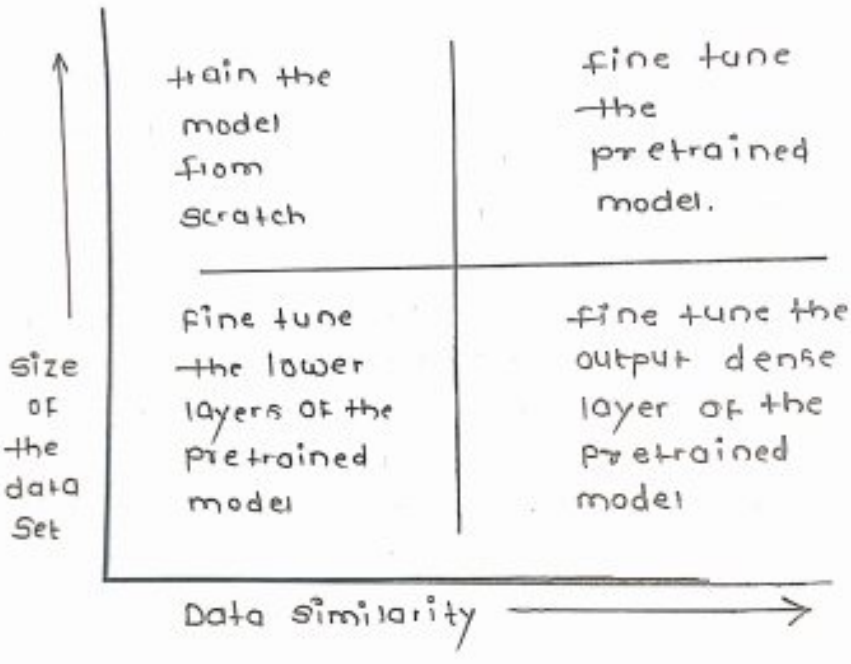
Theory :-

a. Load in pre-trained CNN model trained on a large dataset.

Artificial neural networks such as Recurrent neural networks Convolutional Neural Networks, GANs, and Autoencoders.

The learning rate used in training of the pre-trained model.

Learning rate gradually increases from 0.00002 to 0.002 at epoch 100, and linearly decreases until reaching 0.0002 at epoch 800



pre-trained model is a model created by some one else to solve a similar problem. Instead of building a model from scratch to solve a similar problem

b. freeze parameters (Weights) in model's lower convolutional layers.

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 layer
model.add (conv2D (64, (3,3), trainable = false))
```

layer freezing is process of freezing the weights of specific layer in deep learning network so that these wights don't change during training.

c. Add custom classifire with several layer of ~~traine~~ trainable parameter's of model.

We have compile information about the date of development and trainable parameter counts of if you're not familiar with the MINIST dataset it's a collection of 0-9 digits as images. These images are gray-scale and thus each image can be represented with an input shape of 28x 28x1. as shown in Line 6.

us. to take advantage of that the model has already learned without having to develop it from scratch.
used transfer leaning with efficientnet_B0, and to do is to gradually.

d. Train classifier layer on training data available for task.

- The weights in those layers to be re-initialized.
  base_model.trainable = False
- The next step is to add new trainable layers that will turn old features into predictions on the new dataset.
- This is important because the 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 layer as needed.

- The optimal hyper parameters, let use first understand these hyper parameters:
- learning rate, batch size, momentum, and high weight decay. these hyper parameters act as

- The learning rate is high, then training may not converge or even diverge.
- hyper-parameters for your model is hard. ESpecially if yo do it manually
- ~~select the~~
- Steps hyperparameter tuning:
  1. Select the ~~list of parameter~~ right type of model.
  2. Review the list of parameter's of the model build the HP space.
  3. Finding the method for searching the hyperparameter space.
  4. Applying the cross-validation scheme approach.
  5. Assess the model score to evaluate the model.

Conclusion :-

We have disscussed in detailed study of object detection using transferflow learning of CNN architecture.

Website for Experiment code

1. Deep Learning (sppudeeplearning.blogspot.com)

2. Implementing Feedforward neural networks with Keras and TensorFlow (sppudeeplearning.blogspot.com)

3. deep-learning-keras-tf-tutorial/cnn_cifar10_dataset.ipynb at master · codebasics/deep-learning-keras-tf-tutorial · GitHub

https://sppudeeplearning.blogspot.com/2022/09/3-build-image-classification-model-by.html

https://github.com/codebasics/deep-learning-keras-tf-tutorial/blob/master/16_cnn_cifar10_small_image_classification/cnn_cifar10_dataset.ipynb

4. Use Autoencoder to implement anomaly detection. (sppudeeplearning.blogspot.com)

5. Implement the Continuous Bag of Words (CBOW) Model. (sppudeeplearning.blogspot.com)