Expt. No. 2

Expt. Name Multi layer Perceptron

Page No. 9

Date : 02.02.23

Aim:

To develop a multilayer perceptron

Algorithm:

* Import the necessary libraries
* upload the dataset mnist using keras library
* Then convert the pixels into floating point values
* Then change the numbers into grayscale will makes the values becomes small so that the computation becomes easier. Divide all values by 255 that it will be converted from range 0 to 1
* We can get to understand the structure of dataset by printing the Feature matrix and target matrix of train and test data using .shape function
* Fix the image size with subplots(10,10) using matplotlib
* Using for loop visualize the data using imshow() and .reshape
* Display it using plt.show()
* Using Sequential to create a layer by layer models
* Flatten() is used to flatten the input without affecting its batch size and in each layer we activation to activate the required function
* Create first two Dense layers to make it fully connected
* use compile function with adam as it optimizer and with Sparse_categorical_crossentropy as its loss function
* Fit the data into the model
* To find the accuracy use evaluate() with verbose = 0
* Display the accuracy.
* The first two dense layers are used to make fully connected model & are hidden layer
* The last Dense layer is the output layer which contains 10 neurons that decide which category the image belongs to

Expt. No. 3. Page No. 13

Expt. Name Convolutional Neural Network Model Date 09.02.23

Aim:

To develop a convolutional Neural Network model

Algorithm:

Steps:

(i) Import required libraries

(ii) Load the cifar10 dataset using load_data and split them into training images and testing image plus also the labels of training and testing.

(iii) Normalize the pixel values to be between 0 and 1 by dividing it by 255

(iv) Define a list named class_label that consist the name of the images.

(v) Display the images using matplotlib

(vi) Use a for loop with range 25 and make subplots for each image

(vii) use xticks and yticks to get and set the current tick location and labels of x-axis and y-axis respectively.

(viii) make a positioned place using grid() and display it using imshow() along with their names by using xlabel()

(ix) Finally, display the entire grided images along with their class label using show()

(x) From tensorflow module import the model and use the sequential one

(xi) add a layer to the model using layers() with Conv2D, so that it creates a convolution layer that is convolved with the layer input to produce a tensor of outputs with relu as its activation function

(xii) Then calculate the largest or maximum value in every patch and the feature map using MaxPooling2D of Keras

(xiii) Repeat the (xi) and (xii) steps with different values.

(xiv) Flatten the generated layers input without affecting its batch size.

(xv) Create a dense layer with relu as its activation function which is the output layer that contains 10 neurons to make a conclusion

Expt. No. 4 Page No. 19

Expt. Name Develop Recurrent Neural Network model Date : 16.02.23

Aim:

To develop a recurrent neural network model

Algorithm:

(i) Import required libraries

(ii) Get the data using load() of tensorflow_datasets

(iii) Split them into Training data and Test data

(iv) use element_spec to look at the attributes of train data

(v) use a for loop to iterate every data and convert into array using numpy()

(vi) Declare the buffersize, batch_size, vocab_size.

(vii) Using Keras.layers.vectorization of Tensorflow give the max_tokens to the vocab_size so that it transforms a batch of strings into a dense representation.

(viii) initialize the RNN ie sequential to generate a model.

(ix) Define the bidirectional layer with LSTM(64)

(x) define the dense layer with relu as its activation function.

(xi) Create a variable sample_text and print it via model.predict(sample_text)

(xii) compile the RNN with Adam optimizer and BinaryCrossentropy as its loss function.

(xiii) Then fit the model with train data, 2 as it epochs, validation set as its test_dataset and with validation_step = 30.

(xiv) calculate the test loss and test accuracy using model.evaluate() test data and print it.

Expt. No. 5 Page No. 25

Expt. Name Visualize a deep learning model Date : 23.02.23

Aim:

To visualize a deep learning model.

Algorithm:

(i) Import required libraries

(ii) Create a model using sequential()

(iii) Add a conv2D layer from Keras with input shape (32,32,3), relu as its activation function and with padding equals the same

(iv) repeat (iii) to create another layer

(v) To calculate the largest or maximum value in every patch and feature map using MaxPooling from Keras for the added layer.

(vi) To get a textual and other information about the layers and their order in the model use summary().

(vii) using visualkeras module display a pictorial view of a deep learning model using layered_view().

Program:

```
import tensorflow as tf
import visualkeras
from tensorflow import keras
from keras.models import sequential
from tensorflow.keras.layers import Input, conv2D, Dense, Flatten, Dropout
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.models import Model
model = sequential()
model.add( conv2d 64,(4,4), input_shape = (32,32,3), activation = "relu",
                                                    padding = "same"))
model.add (MaxPooling2D (pool_size = (2,2)))
model.add (conv2D (128, (4,4), input_shape (32,32,3), activation = "relu",
                                                    padding = "same"))
```

Expt. No. 6

Page No. 29

Expt. Name Save and Load a model

Date 23.02.23

Aim:

To save and load a model

Algorithm:

(i) import required libraries

(ii) Extract the mnist dataset using load_data() of keras module

(iii) Split the extracted data into train data and test data

(iv) to cast the train and test records into float values use astype()

(v) normalize the image pixel value by dividing it by 255 so that the value range from 0 to 255

(vi) create a model of sequential, and in that model using Flatten() to reshape the data to 28 * 28 rows.

(vii) create the dense (first) layer with sigmoid as its activation function

(viii) Another dense layer with same sigmoid function

(ix) Then finally add output layer.

(x) Summarize the developed model

(xi) using save_weights('MLPWeights.h5') save the model to .h5

(xii) using load_weights('MLPWeights.h5') load the model

Program:

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.layers import Flatten
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layer import Activation
import matplotlib.pyplot as plt
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype('float32')
```

Expt. No. 7 Page No. 33

Expt. Name Plot a Model learning curve Date :

Aim:

To plot a model learning curve.

Algorithm:

(i) Import required libraries

(ii) Load Sklearn Breast cancer Dataset and split them on basis of target and data

(iii) Set up the network of sequential model and add layers with relu as its activation function.

(iv) Then add output layer with sigmoid as its activation function

(v) Configure the network with optimizer, loss function and accuracy

(vi) Split the data into train set and test set.

(vii) Fit the network with train data and validation data with appropriate epoch and batch size.

(viii) Get the loss, val_loss, accuracy, val_accuracy from the fitted model

(ix) Plot the data points of training & validation accuracy as its title, with proper label and display using legend()

(x) Plot the data points of training & validation loss as its title, with proper label and display using legend().

Program:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import dataset
from sklearn.model_selection import train_test_split
from keras import models
from keras import layers
from keras import optimizers
bc = dataset.load_breast_cancer()
```

Expt. No. 9

Page No. 43

Expt. Name Dropout - Early to reduce overfitting

Date : 16.03.23

Aim:

To write a program to dropout - early to reduce overfitting

Algorithm:

(i) Import required libraries

(ii) Split the dataset into train data and test data of mnist dataset which is loaded using .load_data()

(iii) reshape the x_train and x_test into (60000,784) & (10000,784). resp using reshape()

(iv) convert all the train and test data into float32 using astype()

(v) convert the x_train and x_test into gray scale.

(vi) declare num_classes = 10

(vii) convert the y_train and y_test of a class vectors (ie array like vectors) to binary class matrix using keras.utils.to_categorical with num_classes as its parameter.

(viii) To get number of columns use x_train.shape[1]

(ix) initialize sequential model.

(x) Build the model with 3 dense layer and 2 dropout layers with randomly dropping 20% of neurons.

(xi) Save the model in h5 format

(xii) compile the model with categorical_crossentropy as its loss function and adam as its optimizer.

(xiii) Calculate the test loss and test accuracy using evaluate() with test data and verbose = 0

(xiv) EarlyStopping (patience = 3) allows you to specify an arbitrary large number of training epochs and stop training once the models performance stops improving on a holdout validation set.

(xv) Declare the batch_size

(xvi) Fit the model and evaluate the model with test set.

Expt. No. 10

Page No. 43

Expt. Name Accelerate training with batch normalization Date: 28.02.22

Aim:

To write a program to accelerate training with batch normalization.

Algorithm:

(i) Import required libraries

(ii) Split the dataset into train and test images, labels using fashion_mnist dataset of keras.dataset.fashion_mnist.load_data()

(iii) Convert the train and test images into grayscale.

(iv) Keep the validation image as train images of index before 5000 and validation labels as train labels of index before 5000

(v) Create a model of sequential type, create a flatten layer with input shape [28,28], then create a dense layer with 300 neurons and use_bias = False, then place a BatchNormalization() layer.

(vi) Create activation layer with relu as its activation function, then create a dense layer with 200 neurons and use_bias = False.

(vii) Place a batch normalization layer.

(viii) Create an activation layers, then a dense layer with 100 neurons.

(ix) place a batch normalization layer, activation layer, then a output Dense layer with 10 neurons and softmax as its activation function.

(x) using a for loop print variable name in layers.TOLP layers)

(xi) Initalise the stochastic gradient descent of tensorflow, keras.optimizer.legacy.SGD with parameter lr, decay, momentum, nesterov

(xii) Compile the model

(xiii) fit the model with train_images, train_labels, epochs and with validation data

(xiv) evaluate the model using test_images and test_labels.

Expt. No. 11

Expt. Name Image classification using ResNet, VGG16, inception in imagenet dataset

Page No. 55

Date : 30.03.23

Aim:

To write a program for image classification using ResNet, VGG16, inception in imagenet dataset

Algorithm:

(i) Import required libraries

(ii) create a model of residual network with 50 layers and weights equals the imagenet

(iii) load an image using load_img with target size and convert into array using image.img_to_array().

(iv) using numpy expand the dimension of array along axis=0, then preprocess it using preprocess_input(x)

(v) predict the model using predict() with processed input

(vi) Then display the resnet predicted using decode_predictions(preds_resnet, top=3)[0]

(vii) Again create a model of VGG16 with weights equals the imagenet

(viii) Repeat the process from step (iii) to (vi)

(ix) create a model of InceptionV3 with weights = "imagenet"

(x) DO repeat the same process from step (iii)

Program:

```
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input
                                                    decode_predictions
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
                                                    decode_predictions
from tensorflow.keras.applications.inception_v3 import InceptionV3,
                                                    preprocess_input, decode_predictions
from tensorflow.keras.preprocessing import image
import numpy as np
```