Lab Specification
B.Sc. Engg. Part 4, Even Semester, Session: 2014-2015, Examination 2018
CSE-4232P (Cryptography and Network Security)

1. Suppose you are given a line of text as a plaintext, find out the corresponding Caesar Cipher (i.e. character three to the right modulo 26). Then perform the reverse operation to get the original plaintext.

2. Find out the Polygram Substitution Cipher of a given plaintext (the block size of 3). Then perform the reverse operation to get original plaintext. Consider

3. Consider the plaintext "DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY UNIVERSITY OF RAJSHAHI BANGLADESH", find out the corresponding Transposition Cipher (Take width as input). Then perform the reverse operation to get original plaintext.

4. Find out corresponding double Transposition Cipher of the above plaintext. Then perform the reverse operation to get original plaintext.

5. You are supplied a file of large non repeating set of truly random key letter. Your job is to encrypt the plaintext using ONE TIME PAD technique. Then perform the reverse operation to get original plaintext.

6. Use the Lehmann algorithm to check whether the given number P is prime or not?

7. Use the Robin-Miller algorithm to check whether the given number P is prime or not?

8. Write a program to implement MD5 one way hash function.

9. Write a program to implement Secure Hash Algorithm (SHA) one way hash function.

10. Encrypt the plaintext message using RSA algorithm. Then perform the reverse operation to get original plaintext.

11. Write a program to implement Diffie-Hellman Key Exchange.

pip install opencv-contrib-python matplotlib scipy scikit-learn

(Md. Tohidul Islam)
Assistant Professor
Dept. of CSE
University of Rajshahi

Lab Specification
4th Year 2007-2008 Examination 2011
CSE-416P (Cryptography and Network Security)

1. Suppose you are given a line of text as a plaintext, find out the corresponding Caesar Cipher (i.e. character three to the right modulo 26). Then perform the reverse operation to get original plaintext.

2. Find out the Polygram Substitution Cipher of a given plaintext (Consider the block size of 3). Then perform the reverse operation to get original plaintext.

3. Consider the plaintext "DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLY UNIVERSITY OF RAJSHAHI BANGLADESH", find out the corresponding Transposition Cipher (Take width as input). Then perform the reverse operation to get original plaintext.

4. Find out corresponding double Transposition Cipher of the above plaintext. Then perform the reverse operation to get original plaintext.

5. You are supplied a file of large non repeating set of truly random key letter. Your job is to encrypt the plaintext using ONE TIME PAD technique. Then perform the reverse operation to get original plaintext.

6. Use the Lehmann algorithm to check whether the given number P is prime or not?

7. Use the Robin-Miller algorithm to check whether the given number P is prime or not?

8. Write a program to implement MD5 one way hash function.

9. Write a program to implement Secured Hash Algorithm (SHA) one way hash function.

10. Encrypt the plaintext message using RSA algorithm. Then perform the reverse operation to get original plaintext.

(Md. Tohidul Islam)
Assistant Professor
Department of CSE
University of Rajshahi

https://github.com/Imran4424/Cryptography-LAB

Multimedia Lab:

1.Run length Encoding
2.LZW Encoding
3.Arithmetic Encoding
3.Huffman Encoding
4.DCT
Matlab:
Jpeg Compression Technique(JPEG200)

Web Engineering:

Anik's code: http://gg.gg/xfg0j
Book: http://gg.gg/xfgen
Note: Please click to 'i accept'

AI:

Tensorflow installation guide for Linux ::

Normal Installation steps:

1. sudo apt update
2. sudo apt install python3-dev python3-pip python3-venv
3. mkdir Tensorflow
4. cd Tensorflow
5. python3 -m venv --system-site-packages ./
6. source ./bin/activate
7. pip install --upgrade pip
8. pip install --upgrade tensorflow==2.6
9. pip install opencv-contrib-python matplotlib scipy scikit-learn
10. pip install keras==2.6.0
10. deactivate
11. cd

If you face any problem to install python3-dev python3-pip python3-venv, then follow:

a. sudo rm /var/lib/apt/lists/lock
b. sudo rm /var/cache/apt/archives/lock
c. sudo rm /var/lib/dpkg/lock*
d. sudo dpkg --configure -a

To test whether Tensorflow and other libraries are successfully installed or not

11. Tensorflow/bin/python

>> import tensorflow as tf

>> print(tf.__version__)

2.6.0

>> import cv2

>> import matplotlib

To activate Tensorflow environment:

1. source ./Tensorflow/bin/activate

AI Setup:

1.sudo apt update

2.sudo apt install python3-dev python3-pip python3-venv

3. mkdir Tensorflow

4. cd Tensorflow

5.python3 -m venv --system-site-packages ./

6. source ./bin/activate

7. pip install --upgrade pip

10. Pip install keras==2.6.0

```
import numpy as np
from tensorflow.keras.layers import Dense,Input
from tensorflow.keras.models import Model

def build_model():
    inputs = Input(1,)
    outputs = Dense(1)(inputs)
    model = Model(inputs,outputs)
    model.compile(loss = 'mse')
    model.summary()
    return model

def prepare_data():
    testX = np.arange(100)
    testY = hidden_function(testX)
```

```
    trainX = np.arange(100,65000)
    trainY = hidden_function(trainX)
    return testX,testY, trainX, trainY

def hidden_function(x):
    a=5; b=3;
    y =a*x +b;
    return y

def main():
    model = build_model()
    testX,testY,trainX,trainY = prepare_data()
    model.fit(trainX,trainY, epochs = 20)
    weight = model.layers[1].get_weights()[0][0][0]
    bias = model.layers[1].get_weights()[1][0]
    print('a:{}, b: {}'.format(weight,bias))

if __name__ == "__main__":
        main()
```

Running the code:

```
//Anik_Modak
#include<bits/stdc++.h>
using namespace std;

long long bigmod(long long b,long long p,long long m)
{
        int p1,p2,h;
        if(p==0) return 1;
        if(p%2==1)
        {
        p1=b%m;
        p2=bigmod(b,p-1,m)%m;
        return (p1*p2)%m;
        }
        else
        {
        h=bigmod(b,p/2,m)%m;
        return (h*h)%m;
        }
}

int main()
```

```
{
	int n, t;
	cout<<"Given Number: ";
	cin>>n;

	cout<<"Enter number of tests: ";
	cin>>t;

	if(n==2)
	cout<<"2 is Prime.";
	else if(n%2==0)
	cout << n << " is Composite";
	return 0;
}
```

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2
import cv2
import matplotlib.pyplot as plt
import numpy as np


def main():
	model = VGG16()
	model.summary()

	#load image
	imgPath = '/home/cse/Tensorflow/Lab_Works/royal_enfield.jpg'
	bgrImg = cv2.imread(imgPath)
	print(bgrImg.shape)


	#convert image from BGR into RGB format
	rgbImg = cv2.cvtColor(bgrImg, cv2.COLOR_BGR2RGB)

	#Reshape image so that it can fit into the model
	#display_img(rgbImg)
	rgbImg = cv2.resize(rgbImg, (224, 224))
	#rgbImg = np.expand_dims(rgbImg, axis=0)

	#Expand dimension since model accepts 4D data
	print(rgbImg)
	prediction = model.predict(rgbImg)
	print(prediction)
```

```
def display_img(img):
        plt.imshow(img)
        plt.show()
        plt.close()

if '__name__' == '__main__':
        main()
```

```
 Train a fully connected deep neural network for recognizing English
digits, i.e., 0, 1, 2, ..., 9.
# Sangeeta Biswas
# 31.12.2021

from tensorflow.keras.datasets import mnist
from tensorflow.keras.layers import Input, Dense, Flatten
from tensorflow.keras import Model
from tensorflow.keras.optimizers import RMSprop
import numpy# as np
from tensorflow.keras.callbacks import EarlyStopping, History
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical

DIR = '/home/bibrity/DeepLearning/'

def main():
    # Prepare data sets
    trainX, trainY, testX, testY = prepare_data()

    # Build a model
    model = build_model()

    # Train the model
    callbackList = [EarlyStopping(monitor = 'val_loss', patience = 10),
History()]
    history = model.fit(trainX, trainY, epochs = 300, batch_size = 16,
callbacks = callbackList, validation_split = 0.2)
    plot_loss(history)

    # Check what the model predicts.
```

```
    predictY = model.predict(testX)
    for i in range(10):
       y = np.argmax(testY[i])
       pY = np.argmax(predictY[i])
       print('Original Y: {}, Predicted Y: {}'.format(y, pY))

    # Estimate the performance of the NN.
    model.compile(metrics = 'accuracy')
    model.evaluate(testX, testY)

def build_model():
    inputs = Input((28, 28))
    x = Flatten()(inputs)
    x = Dense(32, activation = 'sigmoid')(x)
    x = Dense(16, activation = 'sigmoid')(x)
    outputs = Dense(10)(x)

    model = Model(inputs, outputs)
    model.summary()

    model.compile(loss = 'mse', optimizer = RMSprop(learning_rate =
0.001))

    return model

def prepare_data():
    # Load data
    (trainX, trainY), (testX, testY) = mnist.load_data()
    #plot_digits(trainX[:9], trainY[:9])
    print(trainX.shape, trainY.shape, testX.shape, testY.shape)

    # Convert numeric digit labels into one-hot vectors.
    # 0: 1 0 0 0 0 0 0 0 0 0
    # 1: 0 1 0 0 0 0 0 0 0 0
    # 2: 0 0 1 0 0 0 0 0 0 0
    print('Labels: {}, DataType: {}'.format(trainY[:10],
trainY[:10].dtype))
    classN = 10
    trainY = to_categorical(trainY, classN)
    testY = to_categorical(testY, classN)
```

```
    print('Labels: {}, DataType: {}'.format(trainY[:10],
trainY[:10].dtype))

    # To convert pixel values from 0-255 into 0-1.
    print('DataType: {}, Max: {}, Min: {}'.format(trainX.dtype,
trainX.max(), trainX.min()))
    trainX = trainX.astype(np.float32)
    testX = testX.astype(np.float32)
    trainX /= 255
    testX /= 255
    print('DataType: {}, Max: {}, Min: {}'.format(trainX.dtype,
trainX.max(), trainX.min()))

    return trainX, trainY, testX, testY

def plot_digits(x, y):
    n = len(y)
    plt.figure(figsize = (20,20))
    for i in range(n):
       plt.subplot(3, 3, i+1)
       plt.imshow(x[i], cmap = 'gray')
       plt.title(y[i])
    plt.show()
    plt.close()

def plot_loss(history):
    loss = history.history['loss']
    valLoss = history.history['val_loss']
    epochs = range(1, len(loss) + 1)

    plt.figure(figsize = (20, 20))
    plt.rcParams['font.size'] = '20'
    plt.plot(epochs, loss, 'bo-', label = 'Training loss')
    plt.plot(epochs, valLoss, 'k*-', label = 'Validation loss')
    plt.title('Training and validation loss')
    plt.legend()

    figPath = DIR + 'Digit_Recognizer_TrainvsVal_Loss.png'
    plt.savefig(figPath)
    plt.close()
```

```
if __name__ == '__main__':
    main()



13-01-21
AI LAB


from tensorflow.keras.applications.vgg16 import VGG16,decode_predictions
from tensorflow.keras.applications.inception_resnet_v2 import
InceptionResNetV2
from  tensorflow.keras.preprocessing.image import load_img
import cv2
import matplotlib.pyplot as plt
import numpy as np
def main():
   model = VGG16()
   model.summary()
   imgPath = './elephant.jpeg'
   # img = load_img(imgPath)
   img = cv2.imread(imgPath)
   print(img.shape)
   rgbImg = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
   # reshape  image
   display(rgbImg)
   rgbImg =cv2.resize(rgbImg,(224,224))
   display(rgbImg)
   rgbImg = np.expand_dims(rgbImg,axis =0)
   print(rgbImg.shape)
   prediction = decode_predictions(model.predict(rgbImg))
   print(prediction)

def display(img):
   plt.imshow(img)
   plt.show()
   plt.close()

if __name__=='__main__':
```

```
main()
```