BHASKARACHARYA NATIONAL INSTITUTE FOR SPACE APPLICATIONS AND GEO-INFORMATICS

WEEKLY PROGRESS REPORT (20/03/2023 – 26/03/2023)

WEEK 9

PROJECT NAME	MALWARE DETECTION USING ML

PROJECT DESCRIPTION :	**DESIGN AND IMPLEMENT ML MODEL TO DETECT MALWARE IN SYSTEM**
GROUP MEMBER :	**PRATHAM PATEL, Shubham Patel, Shashank Sharma, Yash Soni**
GROUP ID :	**I2**
GROUP GUIDE :	**HARSH KIRATSATA**
GROUP COORDINATOR :	**SIDHDHARTH PATEL**
COLLEGE GUIDE NAME :	**MAYANK PATEL, RITIKA LADHA, KINJAL CHAUDHARI**
COLLEGE GUIDE No. :	**+91 9033280445,8866438698, 9723173567**
COLLEGE GUIDE EMAIL. :	**mayank.patel@adaniuni.ac.in, ritika.ladha@adaniuni.ac.in, kinjal.chaudhari@adaniuni.ac.in**
COLLEGE NAME :	**ADANI INSTITUTE OF INFRASTRUCTURE AND ENGINEERING**

20/03/2023 TILL 26/03/2023 (7 DAYS)

Completing the program for Tensor flow.

20/03/2023	Programing the layer function for neural networks.
21/03/2023	Understanding Architecture of NNs.
22/03/2023	Programing loss and evaluate functions.
23/03/2023	Programming placeholders, core functions, debugging and running sessions
24/03/2023	Interpreting the result and changing weights for NN, adjusting epochs for more accuracy.
25/03/2023	Holiday (4th Saturday).
26/03/2023	Holiday (Sunday).

WEEK 10 (PLAN)	We are going to implement KDD in TensorFlow.

REFERENCE :

- https://www.youtube.com/watch?v=ySGeWc7TWBc&t=2158s
- https://github.com/shashankgsharma/malwaredetectionsystem
- https://www.youtube.com/playlist?list=PL74sw1ohGx7GHqDHCkXZeqMQBVUTMrVLE
- https://stackoverflow.com/questions/37383812/tensorflow-module-object-has-no-attribute-placeholder
- https://www.google.com/search?client=firefox-b-d&q=+module+%27tensorflow%27+has+no+attribute+%27placeholder%27
- TensorFlow Documentation

Screenshots :

In [80]:
```
A = x_train.shape[1]
B = len(y_train_onehot[0])
print(A)
print(B)
print("begin:_____")
```

```
1000
2
begin:_____
```

In [81]:
```
precision_scores_list = []
accuracy_scores_list = []
def print_stats_metrics (y_test, y_pred):
    print('Accuracy: %.2f' % accuracy_score (y_test,y_pred) )
    #Accuracy: 0.84
    accuracy_scores_list.append(accuracy_score (y_test, y_pred) )
    confmat = confusion_matrix(y_true=y_test, y_pred=y_pred)
    print ("confusion matrix")
    print(confmat)
    print (pd.crosstab(y_test, y_pred, rownames=['True'], colnames=['Predicted']))
    print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
    print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
    print('Fl-measure: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```

In [82]:
```
def plot_metric_per_epoch():
    x_epochs = []
    y_epochs = []
    for i, val in enumerate (accuracy_scores_list):
        x_epochs.append(i)
        y_epochs.append(val)
    plt.scatter(x_epochs, y_epochs,s=50,c='lightgreen', marker='s', label='score')
    plt.xlabel('epoch')
    plt.ylabel('score')
    plt.title('Score per epoch')
    plt.legend()
    plt.grid()
    plt.show()
```

In [83]:
```
def layer (input, weight_shape, bias_shape):
    weight_stddev = (2.0/weight_shape[0])**0.5
    w_init = tf.random_normal_initializer(stddev = weight_stddev)
    bias_init = tf.constant_initializer (value=0)
    W = tf.get_variable("W", weight_shape, initializer=w_init)
    b = tf.get_variable("b", bias_shape, initializer= bias_init)
    return tf.nn.relu(tf.matmul(input, W) + b)
```

In [84]:
```
def inference_deep_layers (x_tf, n_features, n_columns):
    with tf.variable_scope("hidden_1"):
        hidden_1 = layer (x_tf, [n_features, 30], [30])
    with tf.variable_scope("hidden_2"):
        hidden_2 = layer (hidden_1, [30, 25], [25])
    with tf.variable_scope("hidden_3"):
        hidden_3 = layer (hidden_2, [25, 10],[10])
    with tf.variable_scope("hidden_4"):
        hidden_4 = layer (hidden_3, [10, 5],[5])
    with tf.variable_scope("output"):
        output = layer(hidden_4, [5, n_columns], [n_columns])
    return output
```

In [85]:
```
def loss_deep (output, y_tf):
    xentropy = tf.nn.softmax_cross_entropy_with_logits (logits=output, \
                                                        labels=y_tf)
    loss = tf.reduce_mean(xentropy)
    return loss

def training (cost):
    optimizer = tf.train.GradientDescentOptimizer (learning_rate)
    train_op = optimizer.minimize(cost)
    return train_op
```

In [86]:
```
def evaluate(output, y_tf):
    correct_prediction = tf.equal(tf.argmax(output, 1), tf.argmax(y_tf,1))
    accuracy = tf.reduce_mean (tf.cast (correct_prediction, "float"))
    return accuracy
```

In [94]:
```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
x_tf = tf.placeholder("float", [None,A])
y_tf = tf.placeholder("float", [None,B])
```

In [95]:
```
output = inference_deep_layers(x_tf,A,B)
cost = loss_deep (output,y_tf)
train_op=training(cost)
eval_op = evaluate(output,y_tf)
```

In [89]:
```
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)
#argmax returns the index of the max value for each row
```

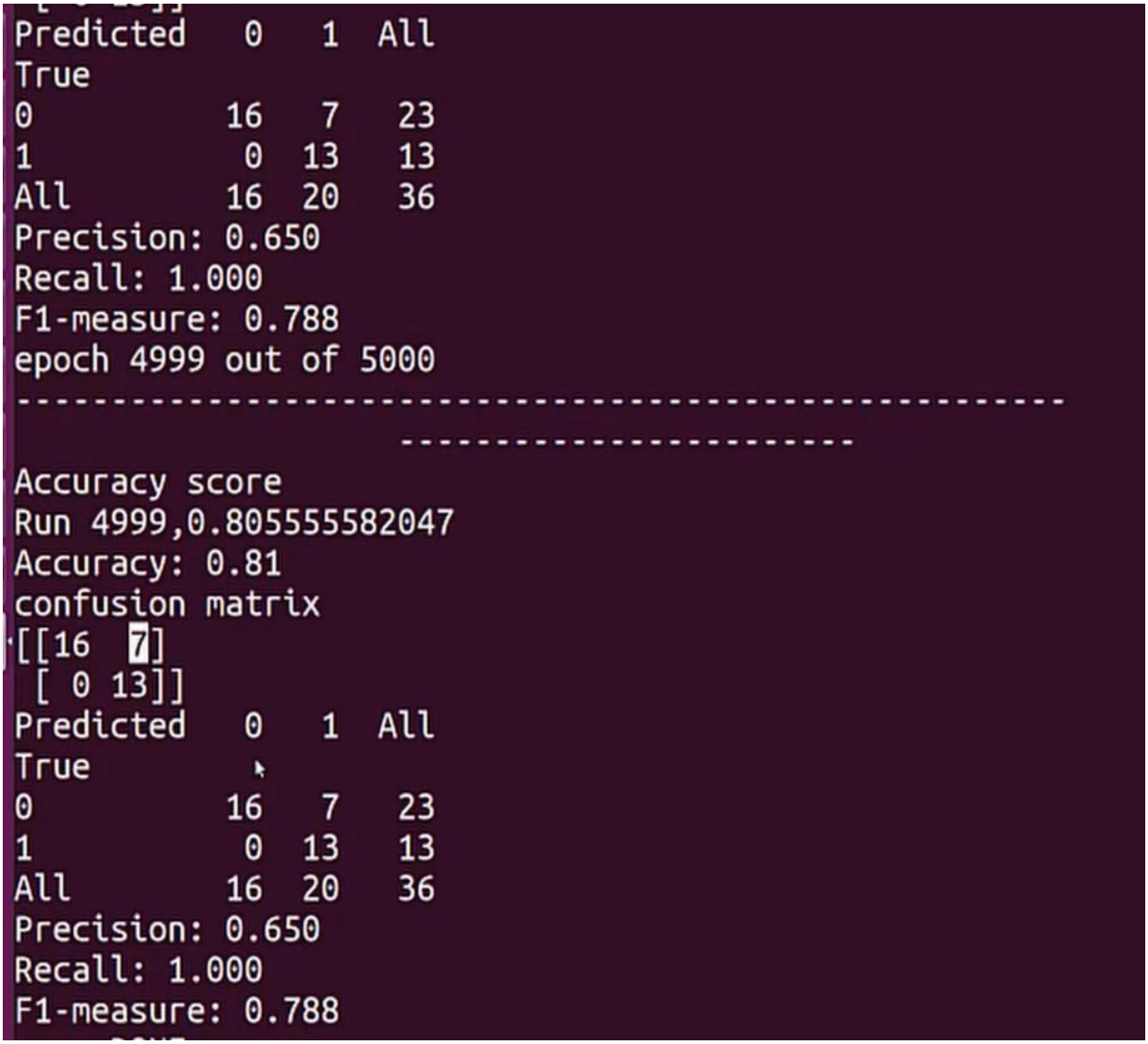
In [90]:
```
y_p_metrics = tf.argmax(output,1)
```

In [91]:
```
for i in range(n_epochs):
    print("epoch %s out of %s" % (i,n_epochs))
    sess.run(train_op,  feed_dict={x_tf:x_train,y_tf:y_train_onehot})
    print ("-------------------------------------------------")

    print ("Accuracy score")
    result, y_result_metrics = sess.run([eval_op, y_p_metrics], \
                                        feed_dict={x_tf: x_test, y_tf: y_test_onehot})

    print("Run {}, {}".format(i,result))
    y_true = np.argmax(y_test_onehot,1)
    y_pred = y_result_metrics
    print_stats_metrics(y_true, y_pred)

    if i ==n_epochs - 1:
        plot_metric_per_epoch()
```


Predicted 0 1 All
True
0 16 7 23
1 0 13 13
All 16 20 36
Precision: 0.650
Recall: 1.000
F1-measure: 0.788
epoch 4999 out of 5000
Accuracy score
Run 4999,0.805555582047
Accuracy: 0.81
confusion matrix
[[16 7]
[0 13]]
Predicted 0 1 All
True
0 16 7 23
1 0 13 13
All 16 20 36
Precision: 0.650
Recall: 1.000
F1-measure: 0.788