assignment2

September 13, 2017

```
In [ ]: # OneVsOne program

In [26]: #   OneVsOne program
         from svmutil import *
         from svm import *
         import numpy as np
         import csv
         from collections import *
         import matplotlib.pyplot as plt
         from sklearn.metrics import *


         def make_file_compatible_libsvm(feat_filename , label_filename,output_file ) :

             x = np.genfromtxt(feat_filename ,delimiter=',', autostrip=True)
             y_label = np.genfromtxt(label_filename ,delimiter='\n')
             x_out = open(output_file,"w")

             counter = 0

             for index,i in enumerate(x , 0):

                 str_=str(int(y_label[index])) + ' '

                 for counter ,value in enumerate(i,1) :
                     if(value==0 or value==0.0) :
                         continue
                     str_ = str_ + str(counter) +':'+str(value)+' '

                 x_out.write(str_+'\n')
             x_out.close()




         #this function takes the label file and according to label it labels it +1 and other la
         #and outputs the file compatibe with libsvm
```

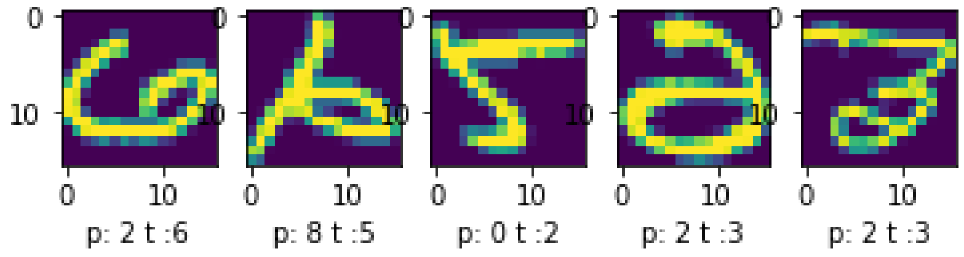
```python
def one_vs_one_compatible_libsvm(feat_filename , label_filename,output_file ,label1 ,la

    x = np.genfromtxt(feat_filename ,delimiter=',', autostrip=True)
    y_label = np.genfromtxt(label_filename ,delimiter='\n')
    x_out = open(output_file,"w")

    counter = 0

    for index,i in enumerate(x , 0):

        label_ = y_label[index]

        if(label_==label2):
            str_ = '-1 '
        elif(label_ == label1) :
            str_ = '+1 '
        else:
            continue

        for counter ,value in enumerate(i,1) :
            if(value==0.0 ) :
                continue
            str_ = str_ + str(counter) +':'+str(value)+' '

        x_out.write(str_+'\n')
    x_out.close()


#creating models for all the given  classes in training set
def predict_one_one(xts , m , labels ):


    score_  =[]
    for i in range(10):
        score_.append(0)

    for i in range(len(labels)):
        for j in range(i+1,len(labels)):
            p_label = libsvm.svm_predict( m[i][j - i -1], xts)
            if(p_label== 1):
                score_[i]+=1
            elif(p_label == -1):
                score_[j]+=1

    return score_
```

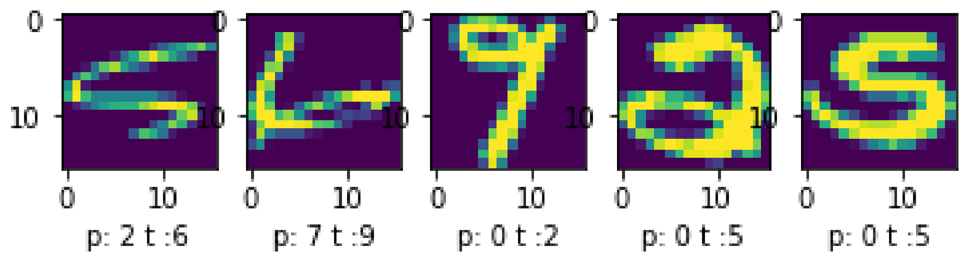
```
def one_vs_one_models(xtr ,ytr,c,lamda ) :

    #getting the number of labels in the training data
    l = np.genfromtxt(ytr,delimiter='\n')
    labels = set(l)
    #print(labels)

    m =defaultdict(list)
    for i in range(len(labels)) :
        for j in range(i+1,len(labels)):
            one_vs_one_compatible_libsvm(xtr,ytr,"output.csv",i,j)
            y, x = svm_read_problem('output.csv')
            model_= svm_train(y , x )
            m[i].append(model_)


    return m ,labels



#************************************************************************************
#as I have got this lamda
m ,labels = one_vs_one_models("USPSTrain.csv" ,"USPSTrainLabel.csv" ,100,0.000012229897
make_file_compatible_libsvm("USPSTest.csv","USPSTestLabel.csv","output.csv")


predicted_label=[]
yts ,xts = svm_read_problem("output.csv")

p_label_file = open("p_label_file.csv" ,'w')
for i in range(len(xts)):
    xts_, idx = gen_svm_nodearray(xts[i])
    score = predict_one_one(xts_ ,m ,labels)
    p_label_file.write(str(score.index(max(score)))+'\n')
    predicted_label.append(score.index(max(score)))

#************************************************************************************
#---------------------calculating f1 ---------------------------
true_predicted_label = 0
for i in range (len(yts)):
    if(predicted_label[i] == yts[i]):
        true_predicted_label +=1

#print("accuracy => ",true_predicted_label/len(yts))
#************************************************
true_predicted_label = []
for i in range (len(labels)):
    true_predicted_label.append(0)
```

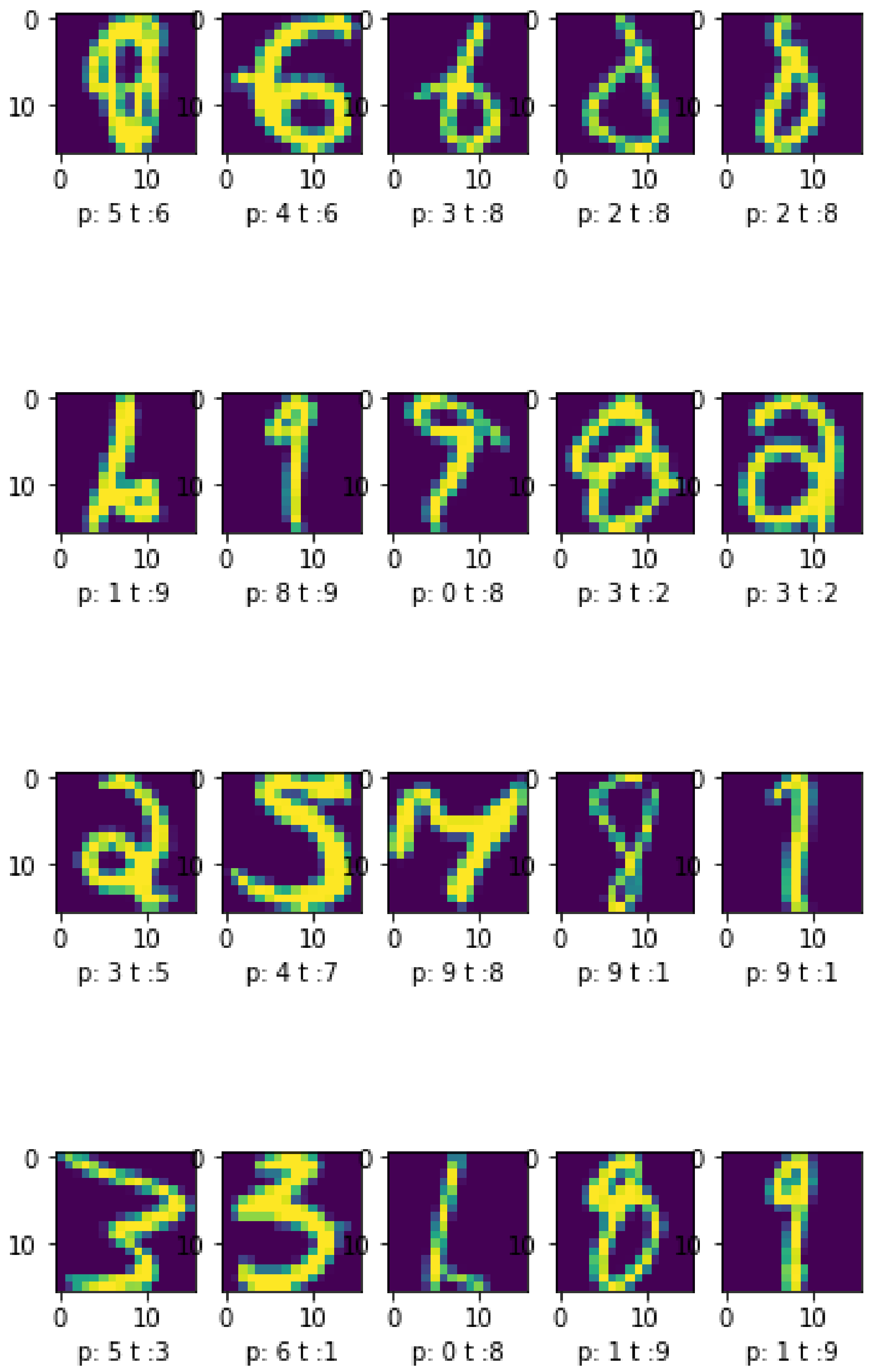
```
true_label = []
for i in range (len(labels)):
    true_label.append(0)

predict_by_classifier = []
for i in range (len(labels)):
    predict_by_classifier.append(0)

for i in range(len(labels)):
    for j in range(len(yts)):
        if(yts[j] == i and predicted_label[j] == i ):
            true_predicted_label[i] +=1
        if(yts[j] == i):
            true_label[i] +=1
        if(predicted_label[j] == i):
            predict_by_classifier[i] +=1

recall_ = sum(true_predicted_label)/sum(true_label)
precision_ = sum(true_predicted_label)/sum(predict_by_classifier)
#calculating f1_score
f1_score = 2*recall_*precision_/(recall_+ precision_)

print("f1_score => ",f1_score)
#----------------------------end of calucating f1 score-------------------



#caculating confusion matrix
y_true = ['0','1','2','3','4','5','6','7','8','9']
y_pred = ['0','1','2','3','4','5','6','7','8','9']
mat = confusion_matrix(yts, predicted_label)
print("confusion_matrix")
print(mat)



#plotting the missclassified images

gs = plt.GridSpec(1, 5)
counter = 0
xts = np.genfromtxt("USPSTest.csv",delimiter=',' ,autostrip=True)
for j in range(len(yts)):
    if predicted_label[j] != yts[j] :
        labe = 'p: '+str(predicted_label[j])+' t :'+str(int(yts[j]))
        plt.xlabel(labe)

        draw = np.array(xts[j])
```

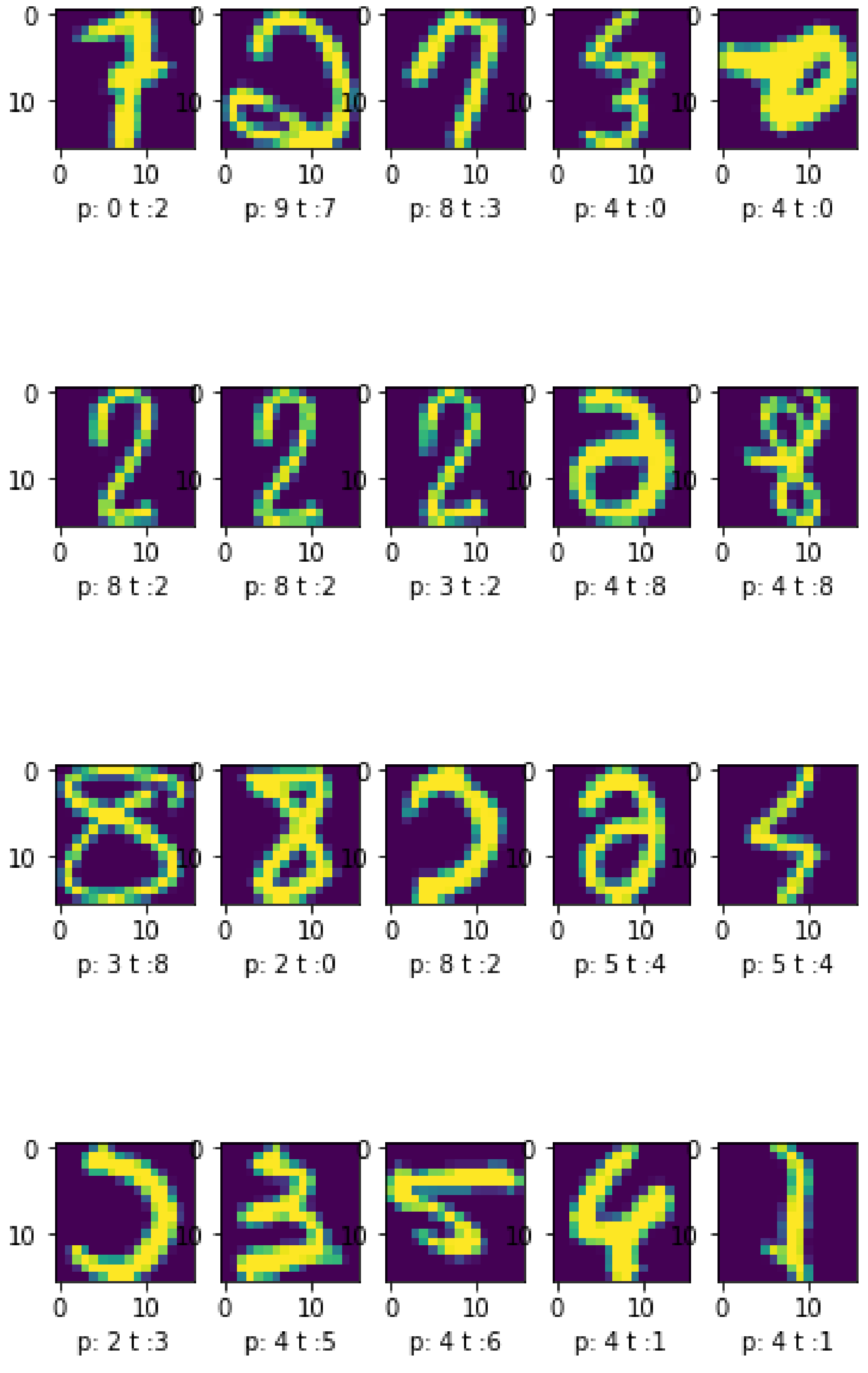
```
                draw = draw.reshape(16,16)

                img =plt.subplot(gs[counter])
                img.imshow(draw)
                counter +=1
                if(counter==5):
                    labe = 'p: '+str(predicted_label[j])+' t :'+str(int(yts[j]))
                    plt.xlabel(labe)
                    plt.show()
                    gs = plt.GridSpec(1, 5)
                    counter=0
```

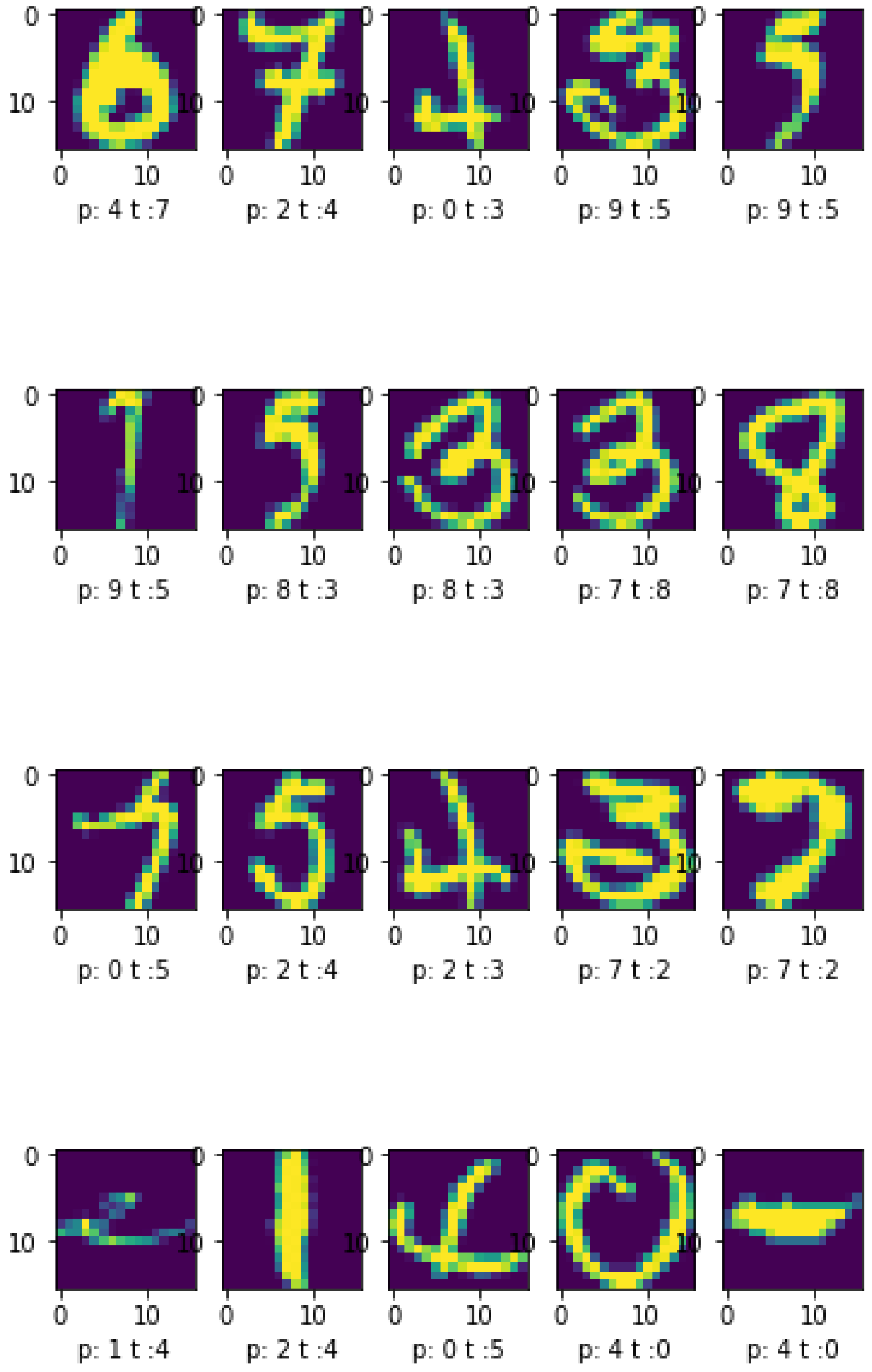
```
f1_score =>  0.9282511210762332
confusion_matrix
[[354   0   2   0   2   0   0   0   1   0]
 [  0 253   1   0   6   0   3   0   0   1]
 [  3   0 180   2   5   1   1   1   5   0]
 [  1   0   7 146   1   7   0   0   4   0]
 [  0   1   6   0 184   1   2   1   1   4]
 [  6   0   0   9   2 139   0   0   1   3]
 [  1   0   4   0   3   2 158   0   2   0]
 [  0   0   1   0   7   0   0 135   1   3]
 [  4   0   3   5   1   4   0   1 146   2]
 [  0   2   0   0   3   1   0   1   2 168]]
```

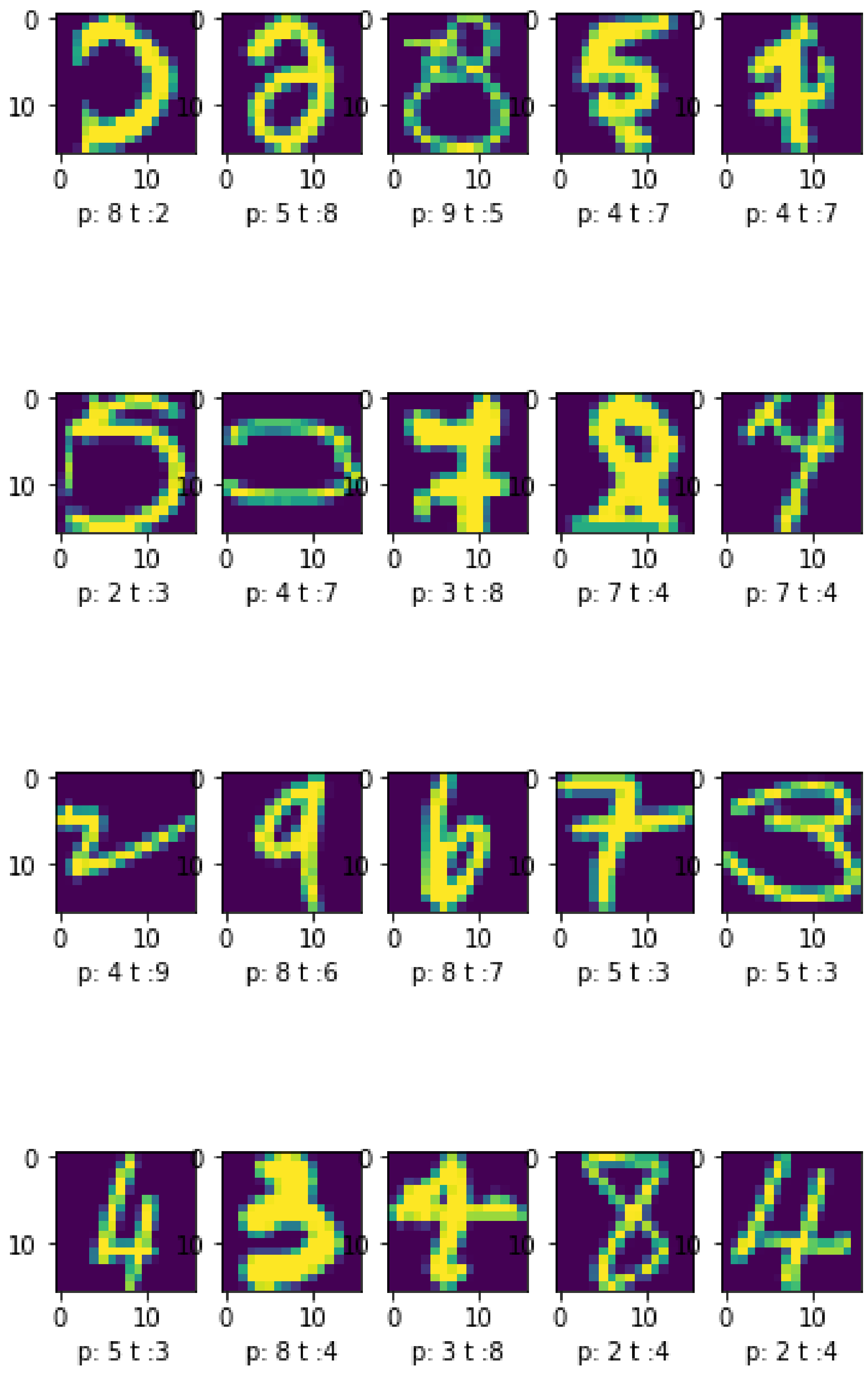


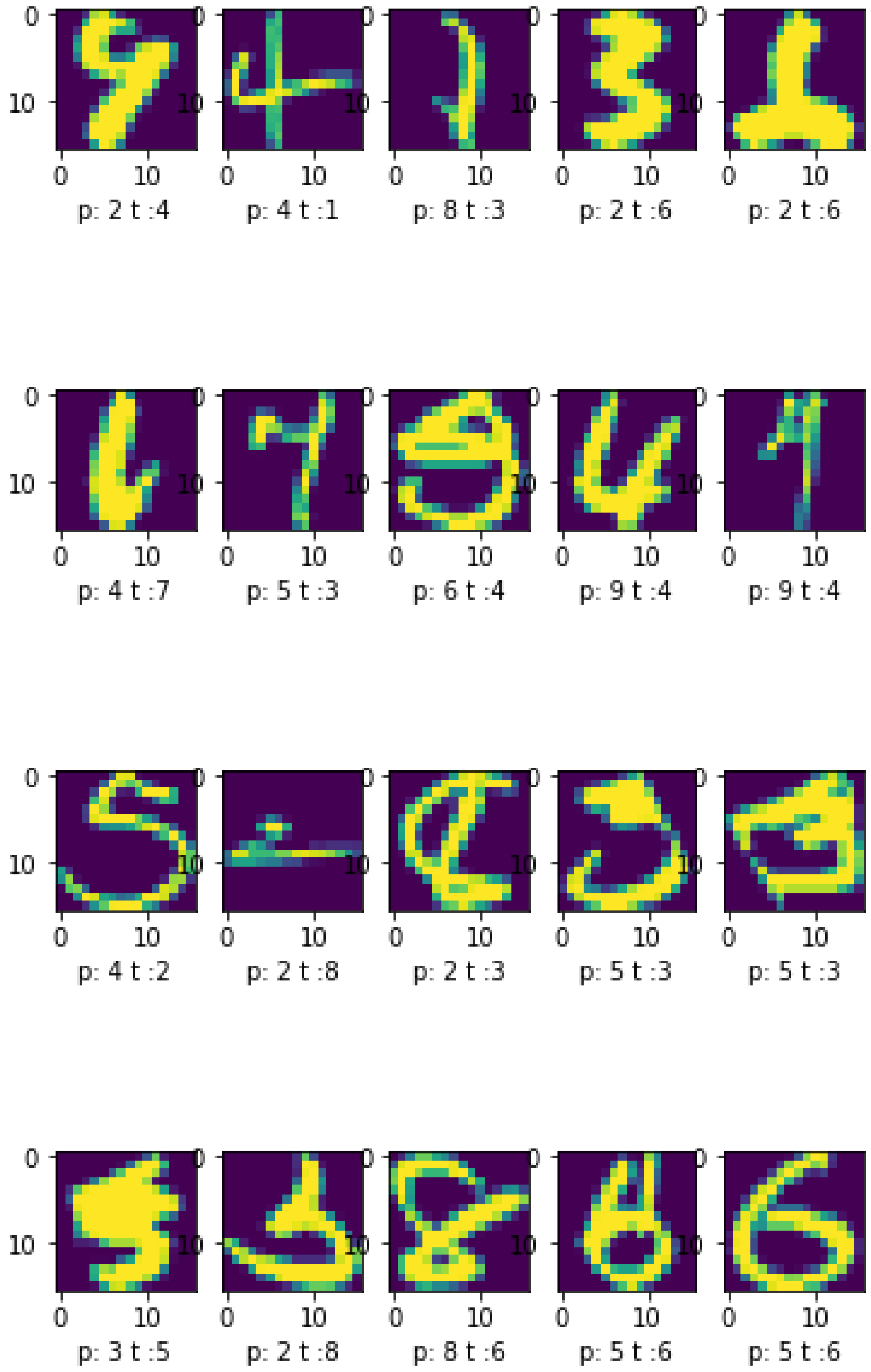


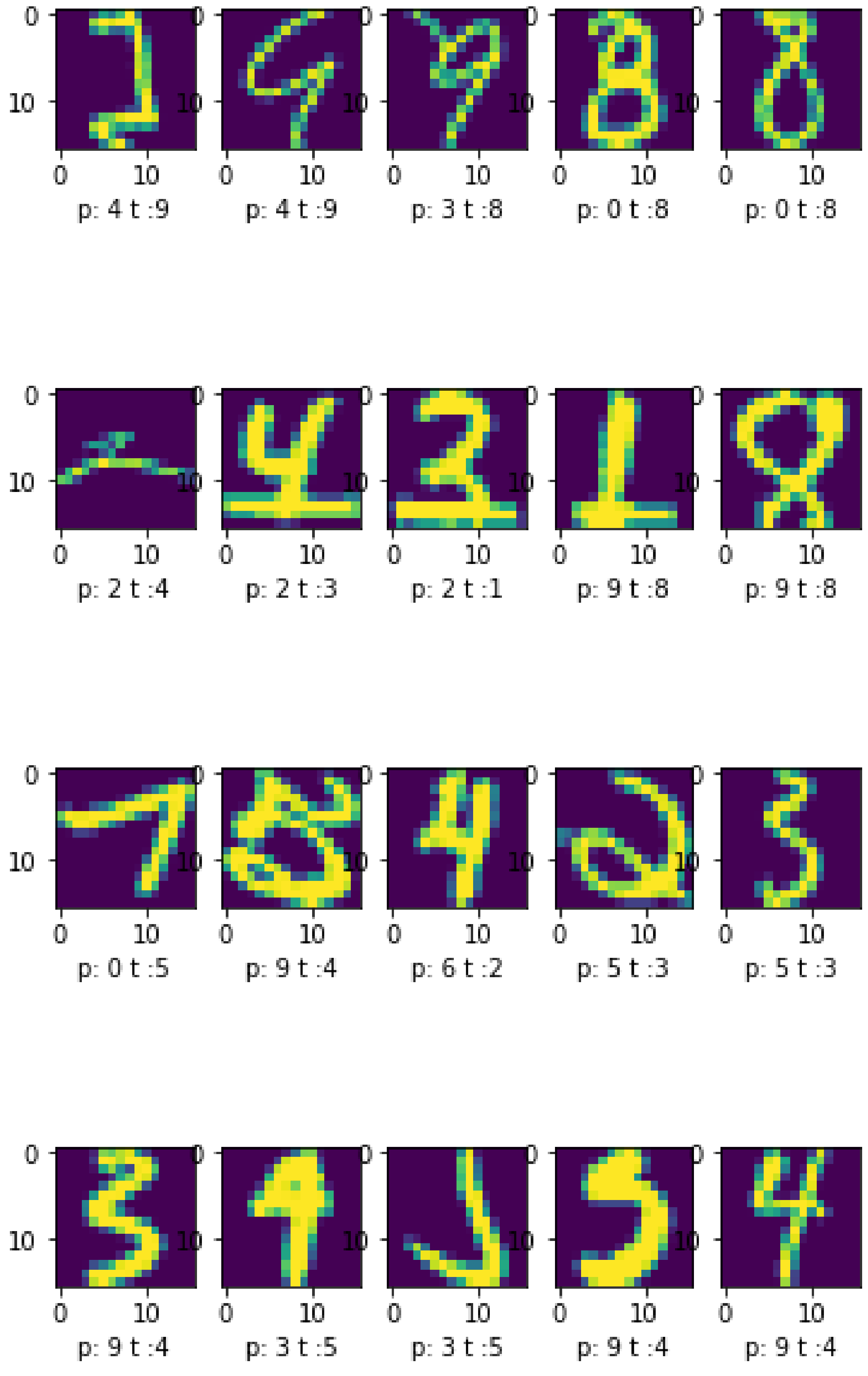

p: 5 t :6
p: 4 t :6
p: 3 t :8
p: 2 t :8
p: 2 t :8
p: 1 t :9
p: 8 t :9
p: 0 t :8
p: 3 t :2
p: 3 t :2
p: 3 t :5
p: 4 t :7
p: 9 t :8
p: 9 t :1
p: 9 t :1
p: 5 t :3
p: 6 t :1
p: 0 t :8
p: 1 t :9
p: 1 t :9

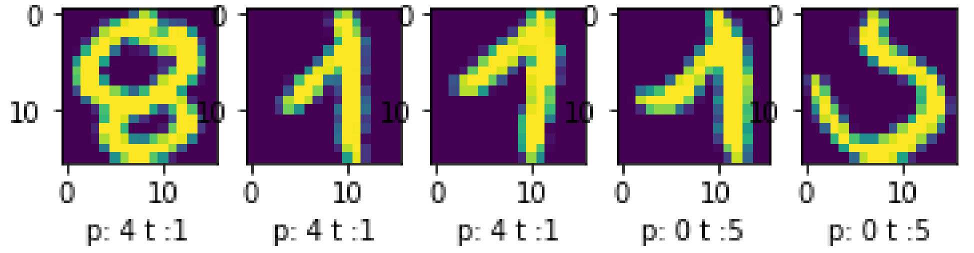

p: 0 t :2
p: 9 t :7
p: 8 t :3
p: 4 t :0
p: 4 t :0
p: 8 t :2
p: 8 t :2
p: 3 t :2
p: 4 t :8
p: 4 t :8
p: 3 t :8
p: 2 t :0
p: 8 t :2
p: 5 t :4
p: 5 t :4
p: 2 t :3
p: 4 t :5
p: 4 t :6
p: 4 t :1
p: 4 t :1

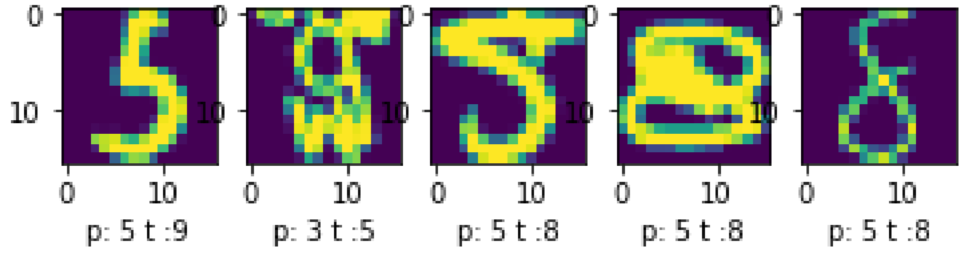

p: 4 t :7
p: 2 t :4
p: 0 t :3
p: 9 t :5
p: 9 t :5
p: 9 t :5
p: 8 t :3
p: 8 t :3
p: 7 t :8
p: 7 t :8
p: 0 t :5
p: 2 t :4
p: 2 t :3
p: 7 t :2
p: 7 t :2
p: 1 t :4
p: 2 t :4
p: 0 t :5
p: 4 t :0
p: 4 t :0


p: 8 t :2
p: 5 t :8
p: 9 t :5
p: 4 t :7
p: 4 t :7
p: 2 t :3
p: 4 t :7
p: 3 t :8
p: 7 t :4
p: 7 t :4
p: 4 t :9
p: 8 t :6
p: 8 t :7
p: 5 t :3
p: 5 t :3
p: 5 t :3
p: 8 t :4
p: 3 t :8
p: 2 t :4
p: 2 t :4


p: 2 t :4
p: 4 t :1
p: 8 t :3
p: 2 t :6
p: 2 t :6
p: 4 t :7
p: 5 t :3
p: 6 t :4
p: 9 t :4
p: 9 t :4
p: 4 t :2
p: 2 t :8
p: 2 t :3
p: 5 t :3
p: 5 t :3
p: 3 t :5
p: 2 t :8
p: 8 t :6
p: 5 t :6
p: 5 t :6


p: 4 t :9
p: 4 t :9
p: 3 t :8
p: 0 t :8
p: 0 t :8
p: 2 t :4
p: 2 t :3
p: 2 t :1
p: 9 t :8
p: 9 t :8
p: 0 t :5
p: 9 t :4
p: 6 t :2
p: 5 t :3
p: 5 t :3
p: 9 t :4
p: 3 t :5
p: 3 t :5
p: 9 t :4
p: 9 t :4





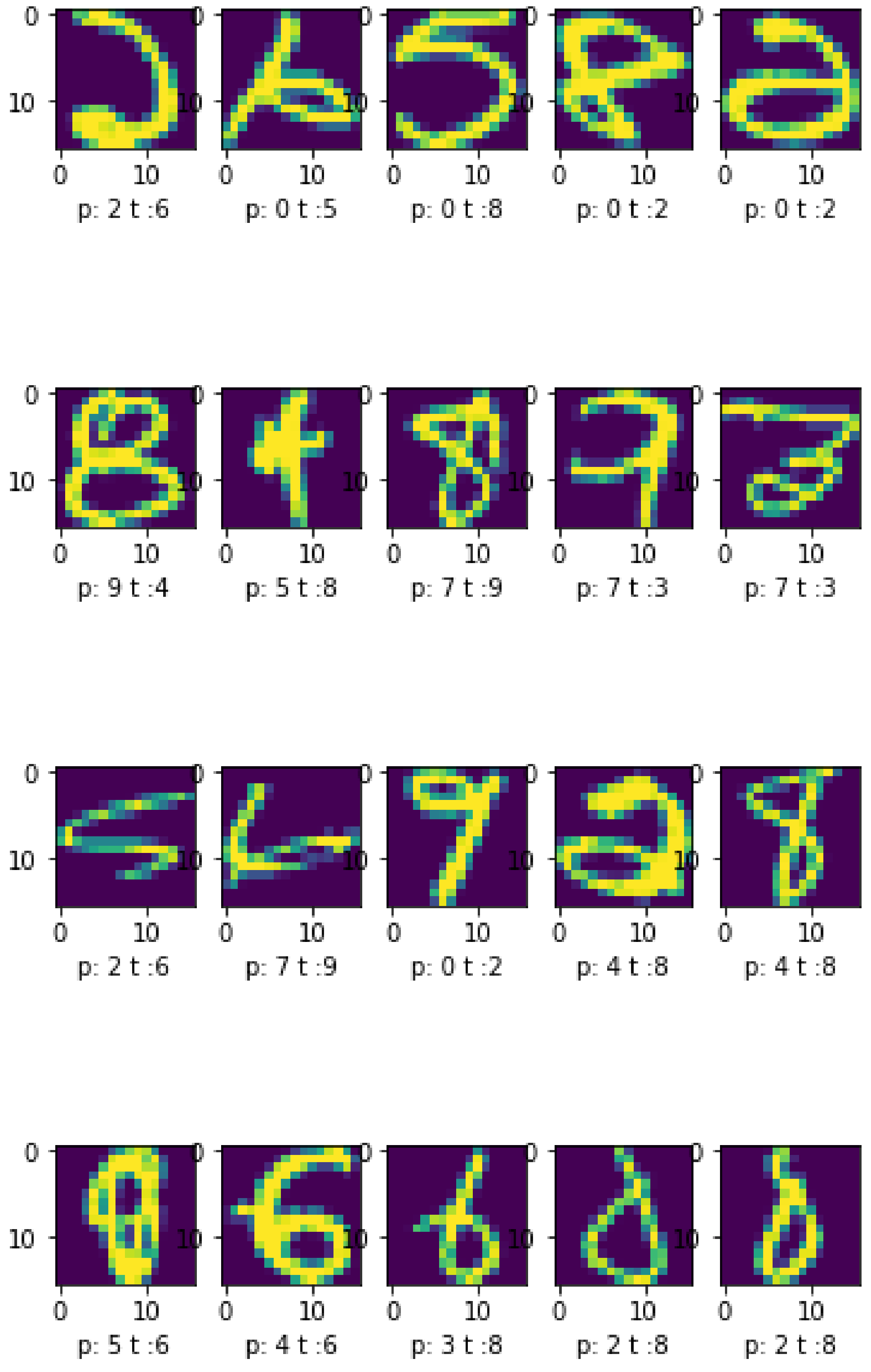
```
In [ ]: #OneVsRest program

In [24]: ## OneVsRest program

         from svmutil import *
         from svm import *
         import numpy as np
         import csv
         from collections import *
         from sklearn.metrics import *


         def make_file_compatible_libsvm(feat_filename , label_filename,output_file ) :

             x = np.genfromtxt(feat_filename ,delimiter=',', autostrip=True)
             y_label = np.genfromtxt(label_filename ,delimiter='\n')
             x_out = open(output_file,"w")

             counter = 0
             for index,i in enumerate(x , 0):

                 str_=str(int(y_label[index])) + ' '

                 for counter ,value in enumerate(i,1) :
                     if(value==0 or value==0.0) :
```

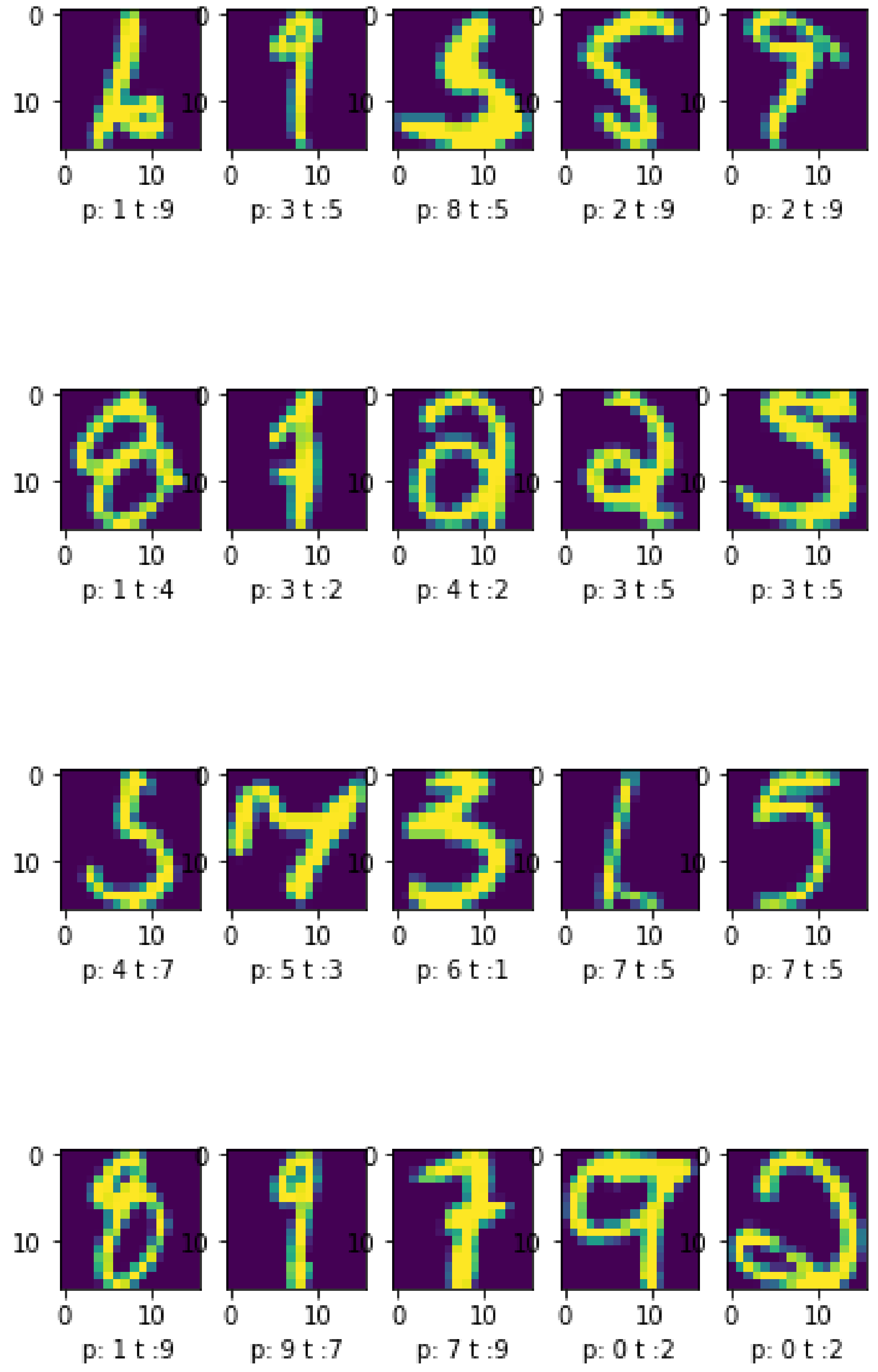
```
                continue
            str_ = str_ + str(counter) +':'+str(value)+' '

        x_out.write(str_+'\n')



#this function takes the label file and according to label it labels it +1 and other la
#and outputs the file compatibe with libsvm
def one_vs_rest_compatible_libsvm(feat_filename , label_filename,output_file ,label1) :

    x = np.genfromtxt(feat_filename ,delimiter=',', autostrip=True)
    y_label = np.genfromtxt(label_filename ,delimiter='\n')
    x_out = open(output_file,"w")

    counter = 0

    for index,i in enumerate(x , 0):

        label_ = y_label[index]

        if(label_==label1):
            str_ = '+1 '
        else:
            str_ = '-1 '

        for counter ,value in enumerate(i,1) :
            if(value==0.0 ) :
                continue
            str_ = str_ + str(counter) +':'+str(value)+' '

        x_out.write(str_+'\n')

    x_out.close()

#models training
def one_vs_rest_models(xtr ,ytr,c,lamda ) :

    #getting the number of labels in the training data
    l = np.genfromtxt(ytr,delimiter='\n')
    labels = set(l)


    m =[]
    for i in range(len(labels)) :
        one_vs_rest_compatible_libsvm(xtr,ytr,"output.csv",i)
        y, x = svm_read_problem('output.csv')
        model_= svm_train(y , x )
        m.append(model_)


    return m ,labels



##calling the for training model
m ,labels = one_vs_rest_models("USPSTrain.csv" ,"USPSTrainLabel.csv" ,100,.0000128)

#making the testing data compatible with libsvm
make_file_compatible_libsvm("USPSTest.csv" ,"USPSTestLabel.csv","output.csv")

#for predicting the labels with binary one_vs_rest classifier
yts ,xts = svm_read_problem("output.csv")

#for each model getting the confidence and on the basis of confidence we predict the la
p_val = []
for i in range(10) :
    p_val.append([])
    p_label, p_acc, p_va = svm_predict( yts ,xts, m[i] )


    for j in p_va :
        p_val[i].append(j[0])
p_val = np.array(p_val)
predicted_label = np.argmax(p_val, axis = 0)

#saving the predicted value to a file
#-----------------------------------------------------------
p_label_file = open("p_label_file.csv" ,'w')
str_ =''
for i in predicted_label :
    str_ = str_+ str(i)+'\n'

p_label_file.write(str_)
#-----------------------------------------------------------

#printing the labels for the test data
print(np.argmax(p_val, axis = 0))


#---------------------calculating f1 ---------------------------
true_predicted_label = 0
for i in range (len(yts)):
    if(predicted_label[i] == yts[i]):
        true_predicted_label +=1
```

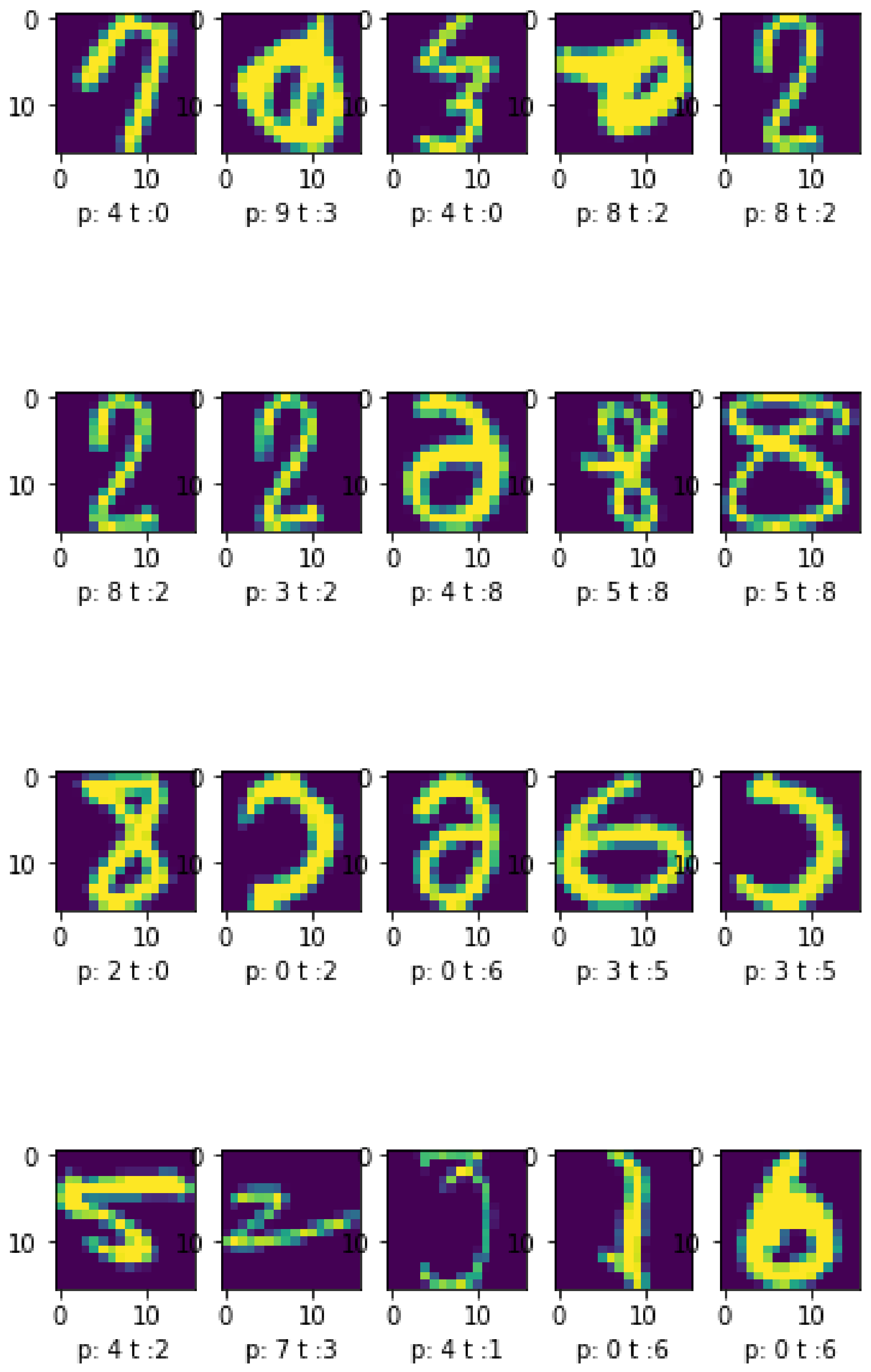
```
#print("accuracy => ",true_predicted_label/len(yts))
true_predicted_label = []
for i in range (len(labels)):
    true_predicted_label.append(0)

true_label = []
for i in range (len(labels)):
    true_label.append(0)

predict_by_classifier = []
for i in range (len(labels)):
    predict_by_classifier.append(0)

for i in range(len(labels)):
    for j in range(len(yts)):
        if(yts[j] == i and predicted_label[j] == i ):
            true_predicted_label[i] +=1
        if(yts[j] == i):
            true_label[i] +=1
        if(predicted_label[j] == i):
            predict_by_classifier[i] +=1

recall_ = sum(true_predicted_label)/sum(true_label)
precision_ = sum(true_predicted_label)/sum(predict_by_classifier)
#calculating f1_score
yts1 =[]
for i in yts:
    yts1.append(int(i))

yts2 =[]
for i in predicted_label:
    yts2.append(int(i))
f1_score = 2*recall_*precision_/(recall_+ precision_)

#----------------------------end of calucating f1 score-------------------


#caculating confusion matrix
y_true = ['0','1','2','3','4','5','6','7','8','9']
y_pred = ['0','1','2','3','4','5','6','7','8','9']
mat = confusion_matrix(yts, predicted_label)
print("confusion_matrix")

print(mat)
#plotting the missclassified images
import matplotlib.pyplot as plt

gs = plt.GridSpec(1, 5)
```

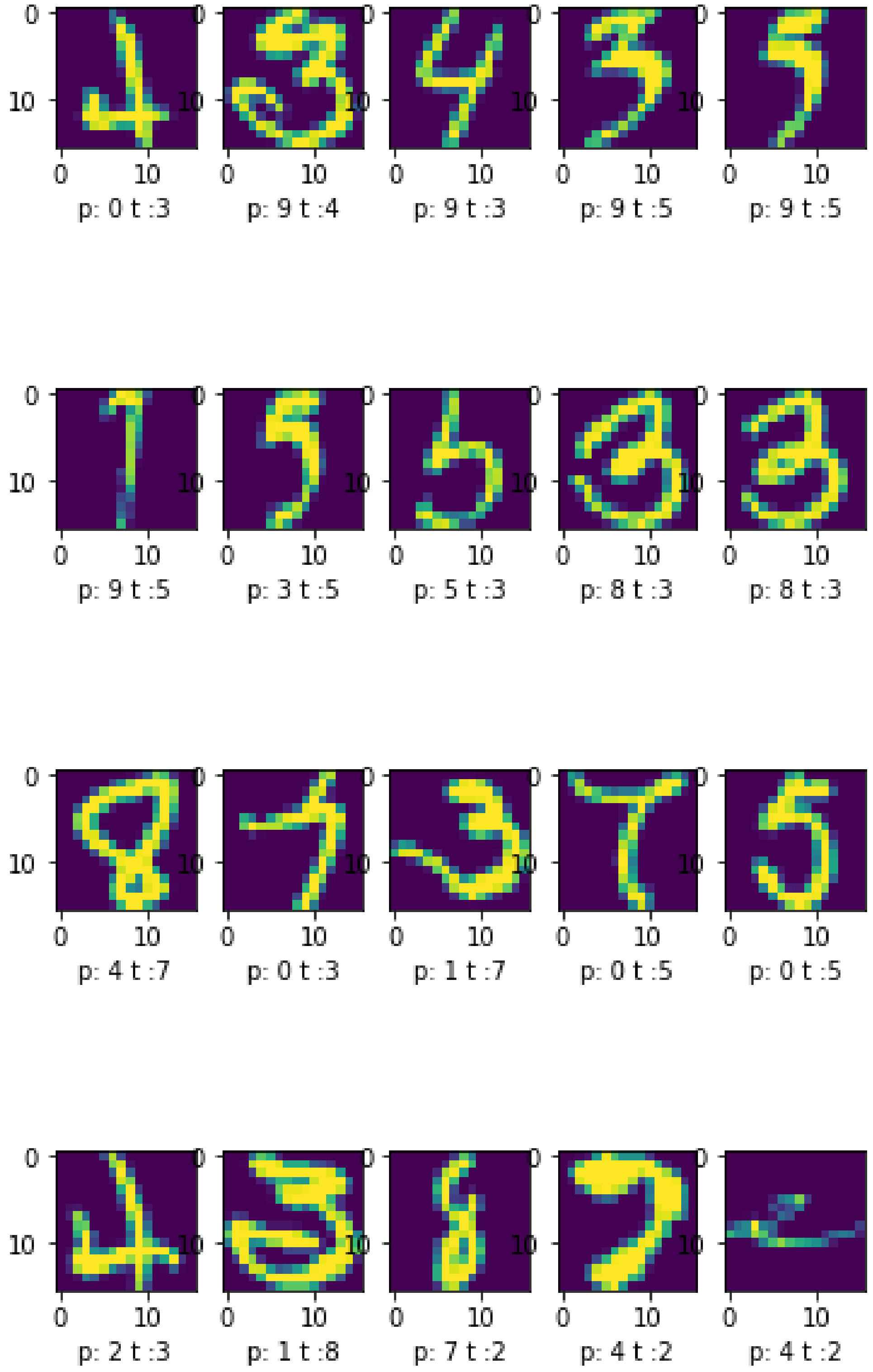
```
counter = 0
xts = np.genfromtxt("USPSTest.csv",delimiter=',' ,autostrip=True)

for j in range(len(yts)):
    if predicted_label[j] != yts[j] :
        labe = 'p: '+str(predicted_label[j])+' t :'+str(int(yts[j]))
        plt.xlabel(labe)

        draw = np.array(xts[j])
        draw = draw.reshape(16,16)

        img =plt.subplot(gs[counter])
        img.imshow(draw)
        counter +=1
        if(counter==5):
            labe = 'p: '+str(predicted_label[j])+' t :'+str(int(yts[j]))
            plt.xlabel(labe)
            plt.show()
            gs = plt.GridSpec(1, 5)
            counter=0
```

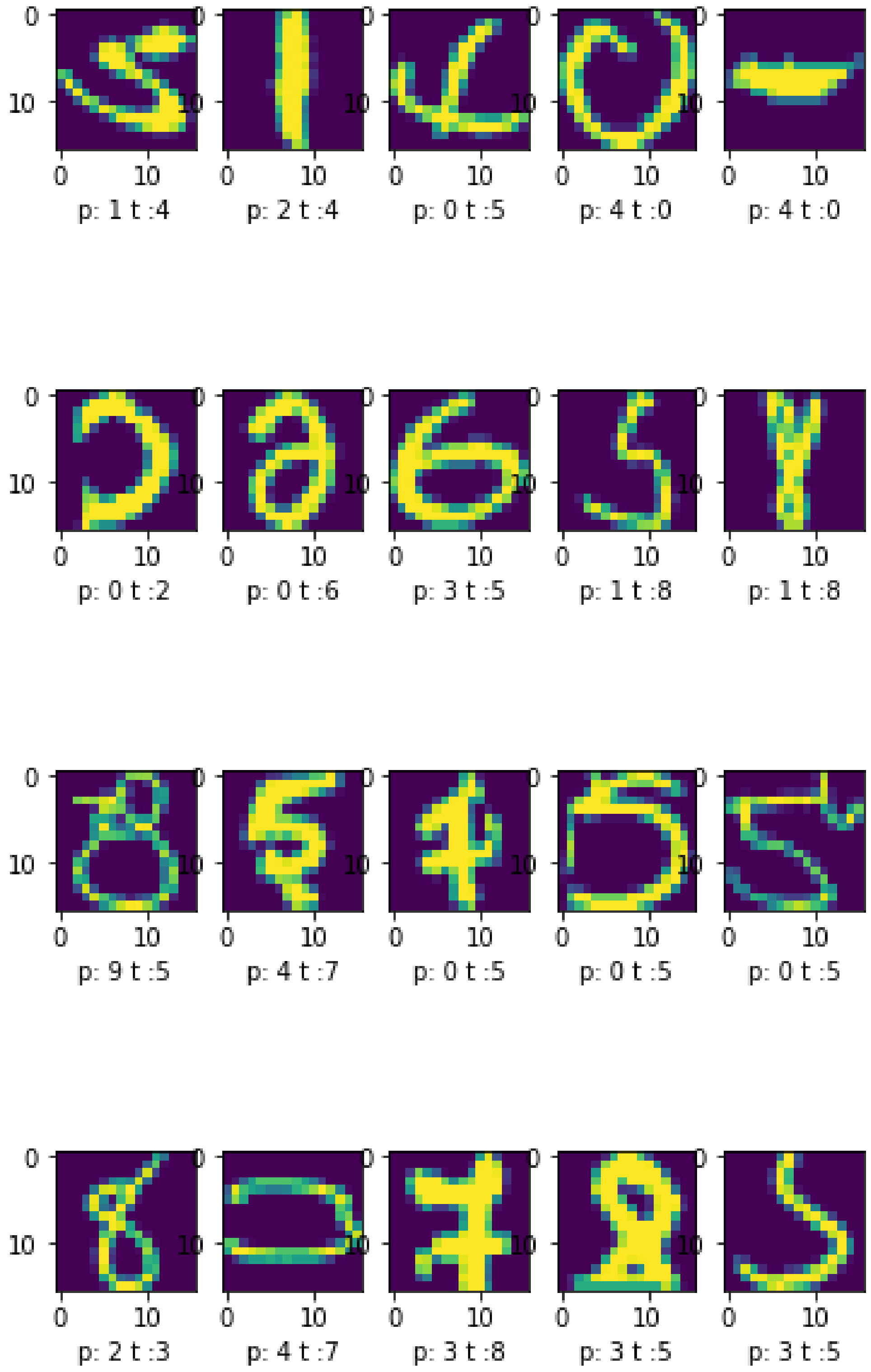
```
Accuracy = 0% (0/2007) (classification)
Accuracy = 12.4564% (250/2007) (classification)
Accuracy = 0% (0/2007) (classification)
Accuracy = 0.0996512% (2/2007) (classification)
Accuracy = 0.199302% (4/2007) (classification)
Accuracy = 0% (0/2007) (classification)
Accuracy = 0% (0/2007) (classification)
Accuracy = 0% (0/2007) (classification)
Accuracy = 0% (0/2007) (classification)
Accuracy = 0% (0/2007) (classification)
[9 6 3 ..., 4 0 1]
confusion_matrix
[[353   0   2   0   3   0   0   0   0   1]
 [  0 255   0   1   5   0   3   0   0   0]
 [  5   0 174   4   6   1   3   2   3   0]
 [  3   0   4 147   1   6   0   2   1   2]
 [  1   2   5   0 183   0   3   1   0   5]
 [  8   0   0  17   2 128   0   1   1   3]
 [  3   0   3   0   2   2 158   0   2   0]
 [  1   1   2   0   6   0   0 134   0   3]
 [  5   2   2   5   3   8   0   2 137   2]
 [  0   3   1   0   3   1   0   3   1 165]]
```

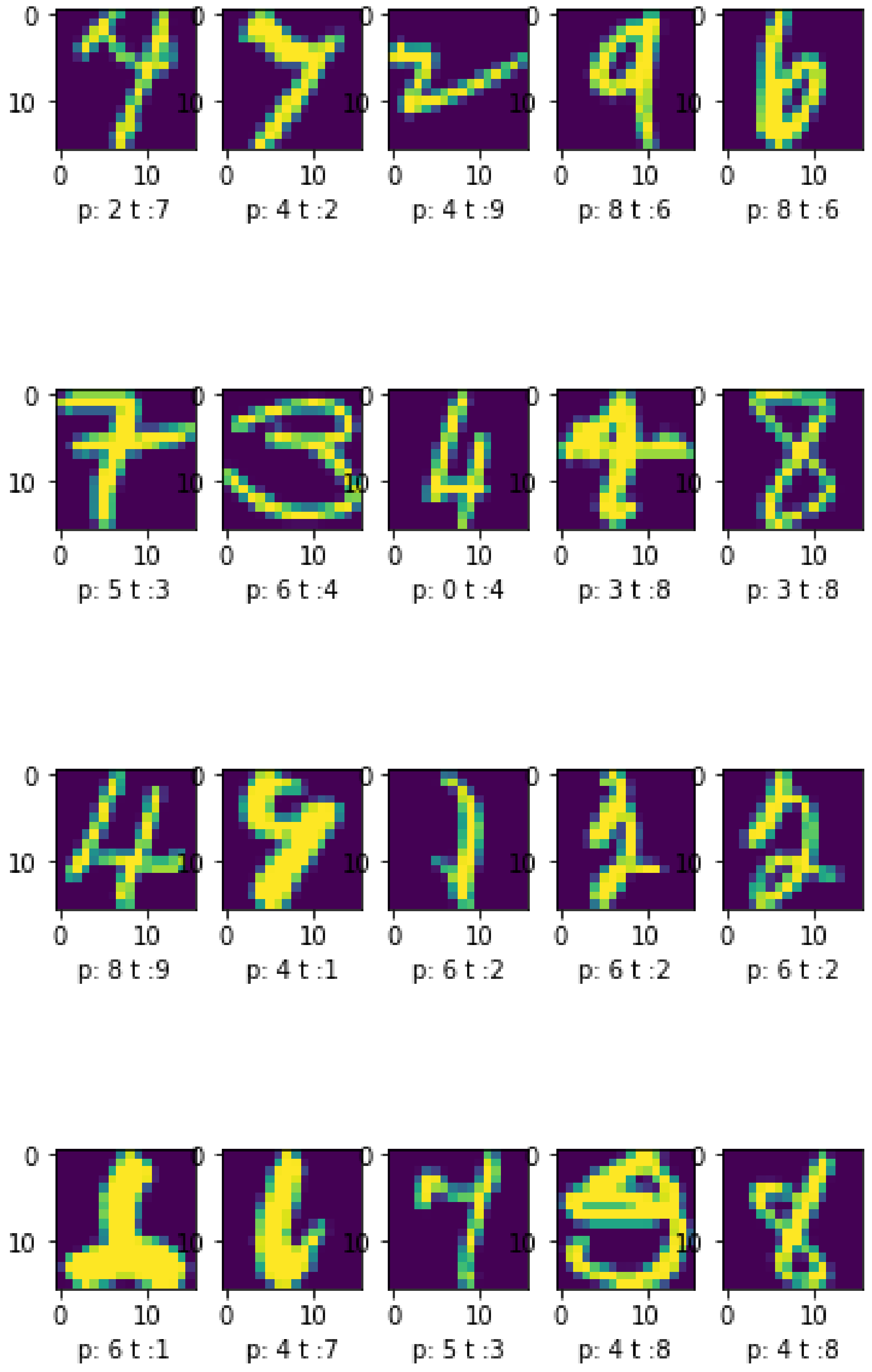

p: 2 t :6
p: 0 t :5
p: 0 t :8
p: 0 t :2
p: 0 t :2
p: 9 t :4
p: 5 t :8
p: 7 t :9
p: 7 t :3
p: 7 t :3
p: 2 t :6
p: 7 t :9
p: 0 t :2
p: 4 t :8
p: 4 t :8
p: 5 t :6
p: 4 t :6
p: 3 t :8
p: 2 t :8
p: 2 t :8

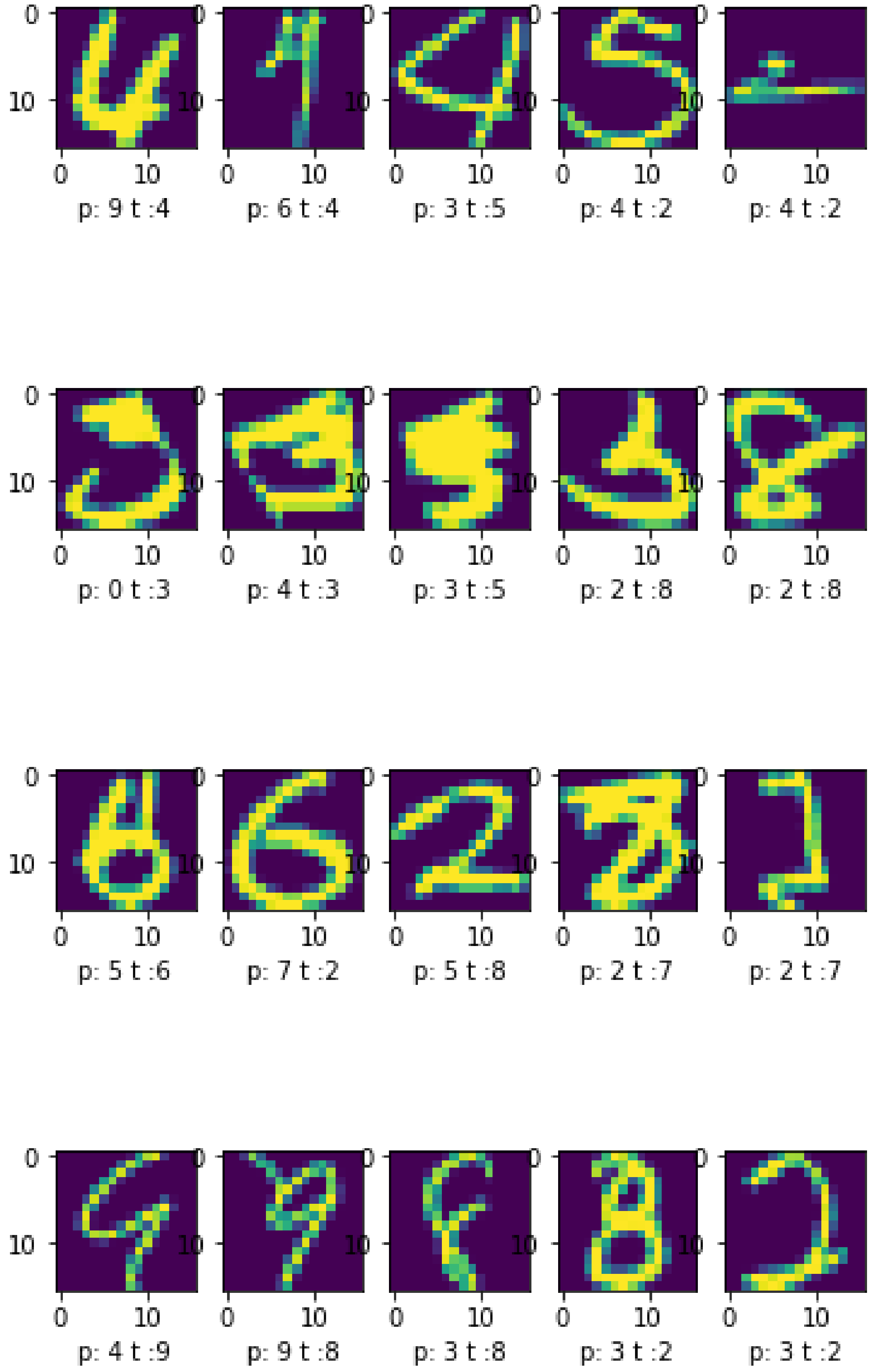

p: 1 t :9
p: 3 t :5
p: 8 t :5
p: 2 t :9
p: 2 t :9
p: 1 t :4
p: 3 t :2
p: 4 t :2
p: 3 t :5
p: 3 t :5
p: 4 t :7
p: 5 t :3
p: 6 t :1
p: 7 t :5
p: 7 t :5
p: 1 t :9
p: 9 t :7
p: 7 t :9
p: 0 t :2
p: 0 t :2

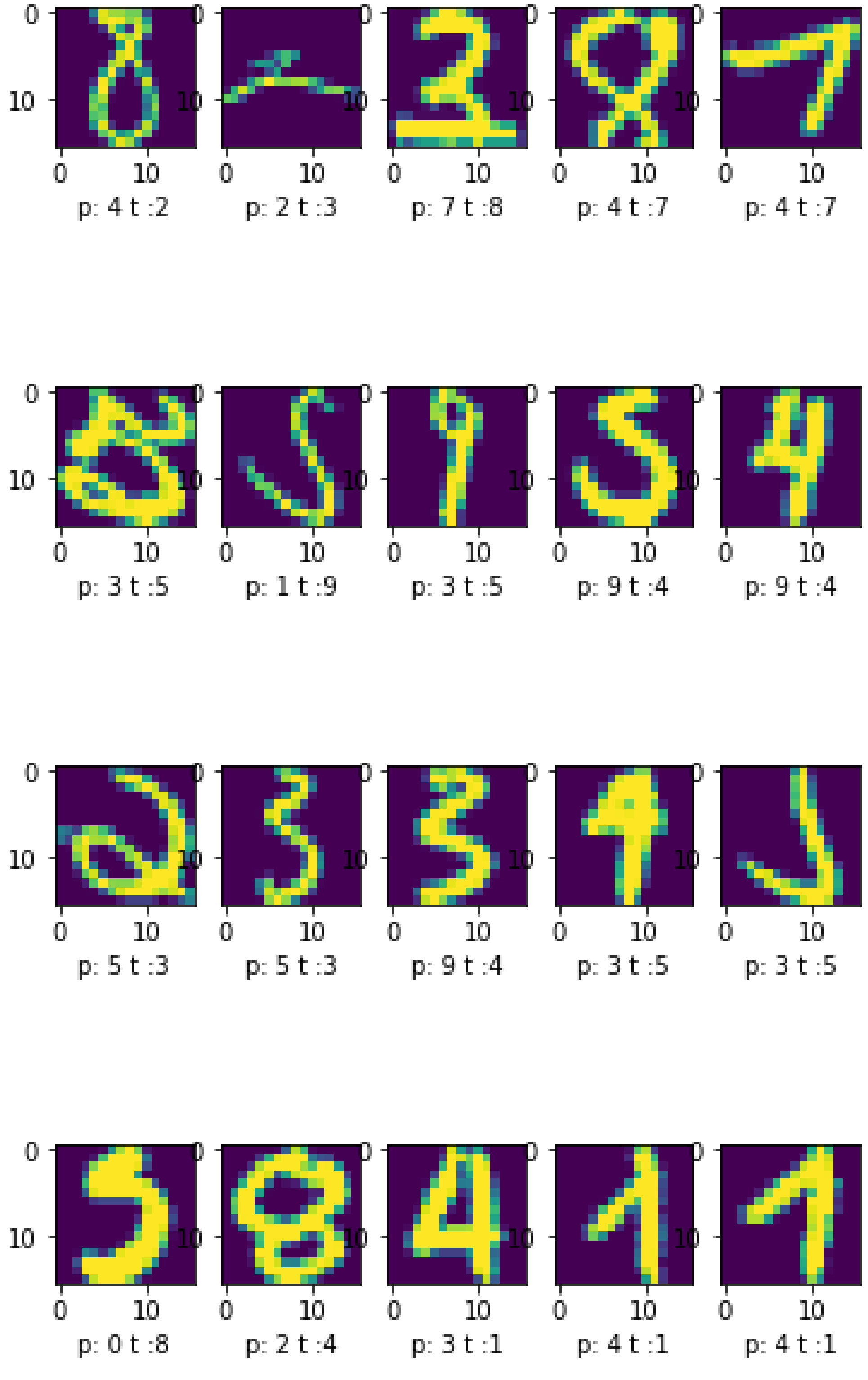

p: 4 t :0
p: 9 t :3
p: 4 t :0
p: 8 t :2
p: 8 t :2
p: 8 t :2
p: 3 t :2
p: 4 t :8
p: 5 t :8
p: 5 t :8
p: 2 t :0
p: 0 t :2
p: 0 t :6
p: 3 t :5
p: 3 t :5
p: 4 t :2
p: 7 t :3
p: 4 t :1
p: 0 t :6
p: 0 t :6

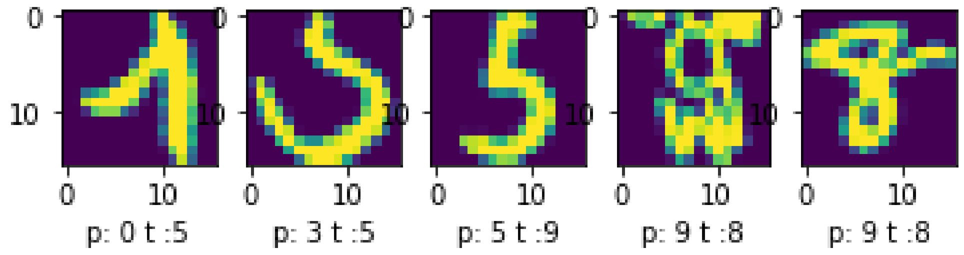

p: 0 t :3
p: 9 t :4
p: 9 t :3
p: 9 t :5
p: 9 t :5
p: 9 t :5
p: 3 t :5
p: 5 t :3
p: 8 t :3
p: 8 t :3
p: 4 t :7
p: 0 t :3
p: 1 t :7
p: 0 t :5
p: 0 t :5
p: 2 t :3
p: 1 t :8
p: 7 t :2
p: 4 t :2
p: 4 t :2

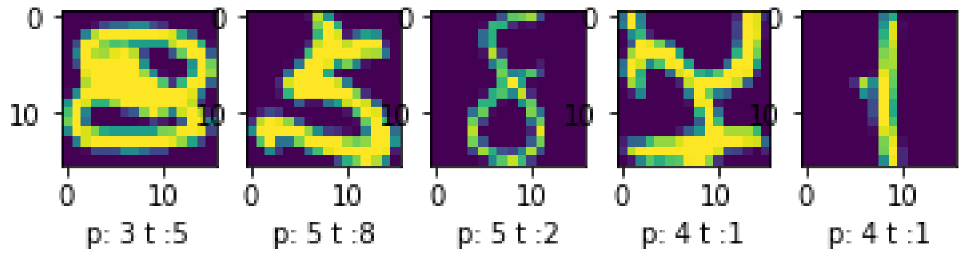

p: 1 t :4
p: 2 t :4
p: 0 t :5
p: 4 t :0
p: 4 t :0
p: 0 t :2
p: 0 t :6
p: 3 t :5
p: 1 t :8
p: 1 t :8
p: 9 t :5
p: 4 t :7
p: 0 t :5
p: 0 t :5
p: 0 t :5
p: 2 t :3
p: 4 t :7
p: 3 t :8
p: 3 t :5
p: 3 t :5


p: 2 t :7
p: 4 t :2
p: 4 t :9
p: 8 t :6
p: 8 t :6
p: 5 t :3
p: 6 t :4
p: 0 t :4
p: 3 t :8
p: 3 t :8
p: 8 t :9
p: 4 t :1
p: 6 t :2
p: 6 t :2
p: 6 t :2
p: 6 t :1
p: 4 t :7
p: 5 t :3
p: 4 t :8
p: 4 t :8


p: 9 t :4
p: 6 t :4
p: 3 t :5
p: 4 t :2
p: 4 t :2
p: 0 t :3
p: 4 t :3
p: 3 t :5
p: 2 t :8
p: 2 t :8
p: 5 t :6
p: 7 t :2
p: 5 t :8
p: 2 t :7
p: 2 t :7
p: 4 t :9
p: 9 t :8
p: 3 t :8
p: 3 t :2
p: 3 t :2


p: 4 t :2
p: 2 t :3
p: 7 t :8
p: 4 t :7
p: 4 t :7
p: 3 t :5
p: 1 t :9
p: 3 t :5
p: 9 t :4
p: 9 t :4
p: 5 t :3
p: 5 t :3
p: 9 t :4
p: 3 t :5
p: 3 t :5
p: 0 t :8
p: 2 t :4
p: 3 t :1
p: 4 t :1
p: 4 t :1





```
In [ ]:
```