Feature Extraction-Copy2

September 17, 2018

1 Helper Functions and Training Procedure:

```
In [2]: import cv2
        import numpy as np
        import matplotlib.pyplot as plt
        from mnist import MNIST
        from skimage.feature import hog
        import itertools
```

```
In [13]: from sklearn import svm
```

```
In [17]: from sklearn import neighbors
```

```
In [3]: def generate_hog(X, cell_size=(8,8)):
            hog_img=[]
            for img in X:
                vec=hog(img, pixels_per_cell=cell_size)
                hog_img.append(vec)
            return np.asarray(hog_img)
```

```
In [4]: def Sigmoid(x):
            return np.asarray(1/(1+np.exp(-x)))

        def ReLU(x):
            return np.asarray(np.maximum(0,x))

        def Softmax(x):
            b=np.exp(x)
            c=np.sum(b, axis=0)
            d=np.divide(b, c)
            return d

        def ReLU_grad(x):
            return np.heaviside(x,0)

        def Sigmoid_grad(x):
            return np.asarray(np.multiply(Sigmoid(x),1-Sigmoid(x)))
```

```
def CrossEntropy(target_, output_):
    return np.sum(-np.sum(np.multiply(target_, np.log((output_))), axis=0))

def CrossEntropy_grad(target_, output_):
    return -np.divide(target_, output_)

def Softmax_CE_grad(target_, pred_):
    grad=-pred_+target_
    return grad
```

In [5]:

```
def add_noise(images):
    size=images.shape
    x=np.random.normal(loc=0.0, scale=10, size=size)
    noisy=np.add(images,x)
    return noisy

def labels_to_class(labels):
    return np.argmax(labels, axis=1)

def confusion_matrix(target_, pred):
    size=len(target_[0])
    target_class=labels_to_class(target_)
    pred_class=labels_to_class(pred)
    cm=np.zeros([size, size])
    for a,p in zip(target_class, pred_class):
        cm[a][p]+=1
    return cm

def cm_metrics(cm):
    diag=(np.diagonal(cm))
    psum=np.sum(cm, axis=0, dtype=np.float32)
    rsum=np.sum(cm, axis=1, dtype=np.float32)
    p=np.divide(diag, psum)
    r=np.divide(diag, rsum)
    prod=np.multiply(p,r)
    sum_=p+r
    f=2*np.divide(prod,sum_)
    a=1.*np.sum(diag)/np.sum(cm)
    return (a,p,r,f)

def plot_confusion_matrix(cm, target_names, title='Confusion matrix',):

    cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    tick_marks = np.arange(len(target_names))
    plt.xticks(tick_marks, target_names)
    plt.yticks(tick_marks, target_names)

    thresh = cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, "{:,}".format(cm[i, j]),
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show()

def plot_loss(train_loss, test_loss, title):
    x=200*np.arange(0, len(train_loss))
    plt.plot(x,train_loss, label='Train')
    plt.plot(x,test_loss, label='Test')
    plt.title(title)
    plt.xlabel("Iterations")
    plt.ylabel("Average Loss")
    plt.legend(loc='upper left')
    plt.show()

def top_k_pred(pred, k):
    sort=np.argsort(pred, axis=0)
    return sort[::-1][:k]
```

In [6]:

```
def shuffle_data(images, labels):
    index=np.random.permutation(len(images))
    shuff_images, shuff_labels=images[index], labels[index]
    return shuff_images, shuff_labels

def get_split_mask(images):
    size=len(images)/5
    a=np.ones(5*size, dtype=bool)
    b=np.arange(10)
    mask=[]
    for i in range(5):
        mask.append([False if (j<(i+1)*size)&(j>=(i)*size) else x for j,x in enumerate(a
    return mask

def get_split_data(mask, images, labels, index):
    mask_=mask[index]
    train_images=images[mask_]
```

```
            train_labels=labels[mask_]
            inv_mask_=np.invert(mask_)
            test_images=images[inv_mask_]
            test_labels=labels[inv_mask_]
            return train_images, train_labels, test_images, test_labels

        def get_accuracy(model, test_images, test_labels):
            batch_size=len(test_images[0])
            count=np.zeros(batch_size)
            pred=model.forward(test_images)
            count=[1 if np.argmax(pred[:,i], axis=0)==np.argmax(test_labels[:,i], axis=0) else 0
            correct=np.sum(count)
            accuracy=100*correct/batch_size
            return(pred, accuracy)

        def SGD_mom(model, batch_images, batch_labels, l2):
            batch_size=len(batch_images[0])
            loss=model.update(batch_images, batch_labels, l2)
            return loss

        def one_hot(labels):
            a=np.zeros((len(labels), 10))
            a[np.arange(len(labels)),labels]=1
            return a

In [7]: class MLP(object):
            def __init__(self, input_size, h1_size, h2_size, output_size):
                self.W1=np.random.normal(loc=0.0, scale=0.08, size=(h1_size, input_size) )
                self.W2=np.random.normal(loc=0.0, scale=0.08, size=(h2_size, h1_size) )
                self.W3=np.random.normal(loc=0.0, scale=0.08, size=(output_size, h2_size) )
                self.B1=np.zeros(h1_size).reshape(-1,1)
                self.B2=np.zeros(h2_size).reshape(-1,1)
                self.B3=np.zeros(output_size).reshape(-1,1)

                self.W1_grad=np.zeros_like(self.W1)
                self.W2_grad=np.zeros_like(self.W2)
                self.W3_grad=np.zeros_like(self.W3)
                self.B1_grad=np.zeros_like(self.B1)
                self.B2_grad=np.zeros_like(self.B2)
                self.B3_grad=np.zeros_like(self.B3)

                self.W1_mom=np.zeros_like(self.W1)
                self.W2_mom=np.zeros_like(self.W2)
                self.W3_mom=np.zeros_like(self.W3)
                self.B1_mom=np.zeros_like(self.B1)
                self.B2_mom=np.zeros_like(self.B2)
                self.B3_mom=np.zeros_like(self.B3)
```

```
def update(self, input_, target_, l2=0):

    batch_size=len(input_[0])
    x=input_
    h1=np.add(np.matmul(self.W1, input_), self.B1)
    a1=act[act_ind](h1)
    h2=np.add(np.matmul(self.W2, a1), self.B2)
    a2=act[act_ind](h2)
    h3=np.add(np.matmul(self.W3, a2), self.B3)
    a3=Softmax(h3)

    loss=CrossEntropy(target_, a3)
    _E_h3=Softmax_CE_grad(a3, target_)

    _a2_h2=act_grad[act_ind](h2)
    _a1_h1=act_grad[act_ind](h1)

    _E_W3=np.matmul(_E_h3,np.transpose(a2))
    _E_B3=np.sum(_E_h3, axis=1).reshape(-1,1)
    _E_a2=np.matmul(np.transpose(self.W3), _E_h3)

    _E_h2=np.multiply(_E_a2, _a2_h2)
    _E_W2=np.matmul(_E_h2, np.transpose(a1))
    _E_B2=np.sum(_E_h2, axis=1).reshape(-1,1)
    _E_a1=np.matmul(np.transpose(self.W2), _E_h2)

    _E_h1=np.multiply(_E_a1, _a1_h1)
    _E_W1=np.matmul(_E_h1, np.transpose(x))
    _E_B1=np.sum(_E_h1, axis=1).reshape(-1,1)
    _E_x=np.matmul(np.transpose(self.W1), _E_h1)

    self.W1_grad=_E_W1/batch_size+self.W1*2*l2
    self.W2_grad=_E_W2/batch_size+self.W2*2*l2
    self.W3_grad=_E_W3/batch_size+self.W3*2*l2
    self.B1_grad=_E_B1/batch_size
    self.B2_grad=_E_B2/batch_size
    self.B3_grad=_E_B3/batch_size

    self.W1_mom=gamma*self.W1_mom+lr*self.W1_grad
    self.W2_mom=gamma*self.W2_mom+lr*self.W2_grad
    self.W3_mom=gamma*self.W3_mom+lr*self.W3_grad
    self.B1_mom=gamma*self.B1_mom+lr*self.B1_grad
    self.B2_mom=gamma*self.B2_mom+lr*self.B2_grad
    self.B3_mom=gamma*self.B3_mom+lr*self.B3_grad

    self.W1-=self.W1_mom
    self.W2-=self.W2_mom
```

```
                self.W3-=self.W3_mom
                self.B1=self.B1-self.B1_mom
                self.B2=self.B2-self.B2_mom
                self.B3=self.B3-self.B3_mom

                return loss

            def forward(self, input_):
                x=input_
                h1=np.add(np.matmul(self.W1, input_), self.B1)
                a1=act[act_ind](h1)
                h2=np.add(np.matmul(self.W2, a1), self.B2)
                a2=act[act_ind](h2)
                h3=np.add(np.matmul(self.W3, a2), self.B3)
                a3=Softmax(h3)
                return a3
```

```
In [28]: input_size=81
         h1_size=500
         h2_size=250
         output_size=10
         gamma=0.99
         lr=1e-3
         act_ind=0
         batch_size=64
         epochs=10
         act=[Sigmoid, ReLU]
         act_grad=[Sigmoid_grad, ReLU_grad]
         actstr={0:'Sigmoid',1:'ReLU'}
```

```
In [9]: from mnist import MNIST
        data=MNIST('/home/pradeep/data/')
        images, labels_ = data.load_training()

        images=np.asarray(images)
        labels_=np.asarray(labels_)
        images=images.reshape(-1,28,28)

        act=[Sigmoid, ReLU]
        act_grad=[Sigmoid_grad, ReLU_grad]

        images, labels_=shuffle_data(images, labels_)
        labels=one_hot(labels_)
        mask=get_split_mask(images)
        hog_img=generate_hog(X=images)
        hog_img*=255
```

```
/home/pradeep/.local/lib/python2.7/site-packages/skimage/feature/_hog.py:150: skimage_deprecatio
```

```
  skimage_deprecation)
```

```
In [27]: def train(model, epochs, images, labels, fold_index, l2=0):
             train_loss=[]
             test_loss=[]
             train_images, train_labels, test_images, test_labels=get_split_data(mask, images, l
             num_batches=len(train_images)/batch_size
             batch_images=np.array_split(train_images, num_batches)
             batch_labels=np.array_split(train_labels, num_batches)
             for epoch in range(epochs):
                 for i in range(num_batches):
                     size=len(batch_images[i])
                     loss=SGD_mom(model, np.transpose(batch_images[i]), np.transpose(batch_label
                     train_avg_loss=loss/size
                     if((i)%200==0):
                         print("Epoch "+str(epoch+1)+" Iteration "+str(i+1)+" : Avg Loss = "+str
                         pred, accuracy=get_accuracy(model, np.transpose(test_images), np.transp
                         test_avg_loss= CrossEntropy(np.transpose(test_labels), pred)/len(test_i
                         train_loss.append(train_avg_loss)
                         test_loss.append(test_avg_loss)
             plot_loss(train_loss, test_loss, "Plot of loss for "+actstr[act_ind]+" for Fold "+s
             output_cm_scores(model, test_images, test_labels)
             #plot_images(model, test_images[:20], 3)
```

```
In [10]: def output_cm_scores_(pred, labels):
             cm=confusion_matrix_(pred=pred, target_=labels)
             a,p,r,f=cm_metrics(cm)
             print "Accuracy = "+ str(a)
             print "Precision = ", p
             print "Recall = ", r
             print "F1 Score = ", f
             target_names=['0','1','2','3','4','5','6','7','8','9']
             plot_confusion_matrix(cm,target_names)

         def confusion_matrix_(target_, pred):
             size=10
             cm=np.zeros([size, size])
             for a,p in zip(target_, pred):
                 cm[a][p]+=1
             return cm
```

```
In [18]: def train_KNN(hog_img, labels, fold_index):
             KNN=neighbors.KNeighborsClassifier(n_neighbors=5, n_jobs=-1)
             mask=get_split_mask(hog_img)
             train_hog, train_labels, test_hog, test_labels=get_split_data(images=hog_img, label
             KNN.fit(train_hog, train_labels)
             pred=KNN.predict(test_hog)
             output_cm_scores_(pred, np.transpose(test_labels))
```

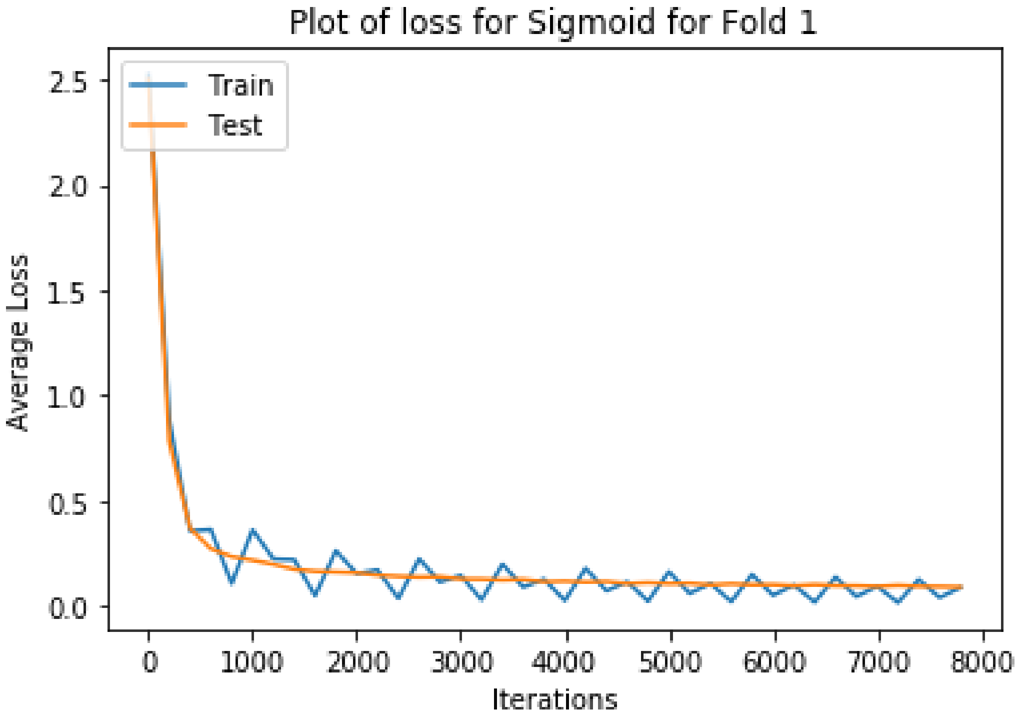
```
In [1]: def train_SVM(hog_img, labels, fold_index):
            SVM=svm.LinearSVC(random_state=0)
            mask=get_split_mask(hog_img)
            train_hog, train_labels, test_hog, test_labels=get_split_data(images=hog_img, labels
            SVM.fit(train_hog, train_labels)
            pred=SVM.predict(test_hog)
            output_cm_scores_(pred,np.transpose(test_labels))
```

2 Training Results:

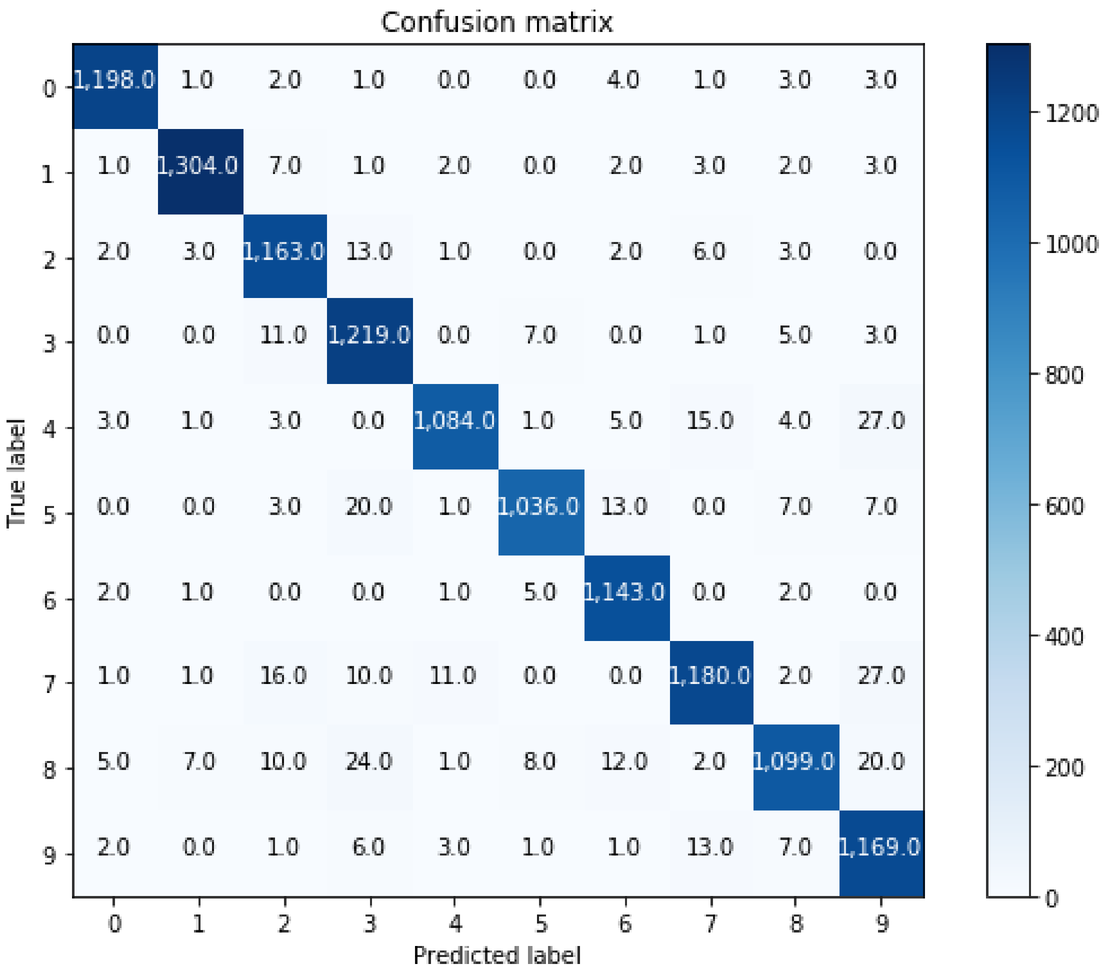
2.0.1 Sigmoid Activation: Fold - 1 , Learning Rate=1e-3

```
In [71]: model0=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=0, model=model0)
```

```
Epoch 1 Iteration 1 : Avg Loss = 2.5237310604414125
Epoch 1 Iteration 201 : Avg Loss = 0.884445213530523
Epoch 1 Iteration 401 : Avg Loss = 0.35903156786276985
Epoch 1 Iteration 601 : Avg Loss = 0.3655924703778429
Epoch 2 Iteration 1 : Avg Loss = 0.10902911774288424
Epoch 2 Iteration 201 : Avg Loss = 0.36398440514331526
Epoch 2 Iteration 401 : Avg Loss = 0.2258086851886312
Epoch 2 Iteration 601 : Avg Loss = 0.22260797401805532
Epoch 3 Iteration 1 : Avg Loss = 0.05129968587343006
Epoch 3 Iteration 201 : Avg Loss = 0.26492773978105
Epoch 3 Iteration 401 : Avg Loss = 0.1559271781710433
Epoch 3 Iteration 601 : Avg Loss = 0.1737974770920139
Epoch 4 Iteration 1 : Avg Loss = 0.03843904655200772
Epoch 4 Iteration 201 : Avg Loss = 0.22684122539521784
Epoch 4 Iteration 401 : Avg Loss = 0.11652301922279068
Epoch 4 Iteration 601 : Avg Loss = 0.14757747366434396
Epoch 5 Iteration 1 : Avg Loss = 0.031926515058978984
Epoch 5 Iteration 201 : Avg Loss = 0.20134238224677115
Epoch 5 Iteration 401 : Avg Loss = 0.0924885450678404
Epoch 5 Iteration 601 : Avg Loss = 0.12984148966792264
Epoch 6 Iteration 1 : Avg Loss = 0.028390987012759175
Epoch 6 Iteration 201 : Avg Loss = 0.18114311356343926
Epoch 6 Iteration 401 : Avg Loss = 0.0749974610359073
Epoch 6 Iteration 601 : Avg Loss = 0.11802888229472246
Epoch 7 Iteration 1 : Avg Loss = 0.02568164330554715
Epoch 7 Iteration 201 : Avg Loss = 0.16530945625132126
Epoch 7 Iteration 401 : Avg Loss = 0.062439104136879364
Epoch 7 Iteration 601 : Avg Loss = 0.10933906205265685
Epoch 8 Iteration 1 : Avg Loss = 0.02327578612019556
Epoch 8 Iteration 201 : Avg Loss = 0.15203635959131023
Epoch 8 Iteration 401 : Avg Loss = 0.053729359559740106
Epoch 8 Iteration 601 : Avg Loss = 0.10221266638664009
Epoch 9 Iteration 1 : Avg Loss = 0.02109309421811379
Epoch 9 Iteration 201 : Avg Loss = 0.14007147946780688
Epoch 9 Iteration 401 : Avg Loss = 0.04746798105823839
Epoch 9 Iteration 601 : Avg Loss = 0.0961481985689468
Epoch 10 Iteration 1 : Avg Loss = 0.018899770552620404
Epoch 10 Iteration 201 : Avg Loss = 0.12860275403795643
Epoch 10 Iteration 401 : Avg Loss = 0.04269410014086432
Epoch 10 Iteration 601 : Avg Loss = 0.09044278243873298
40
```



```
Accuracy = 0.96625
Precision =  [0.98682043 0.98937785 0.95641447 0.94204019 0.98188406 0.97920605
 0.96700508 0.96642097 0.9691358  0.92851469]
Recall =  [0.98763397 0.98415094 0.97485331 0.97833066 0.94838145 0.95308188
 0.99046794 0.94551282 0.92508418 0.97173732]
F1 Score =  [0.98722703 0.98675747 0.96554587 0.95984252 0.96484201 0.96596737
 0.97859589 0.95585257 0.94659776 0.94963444]
```

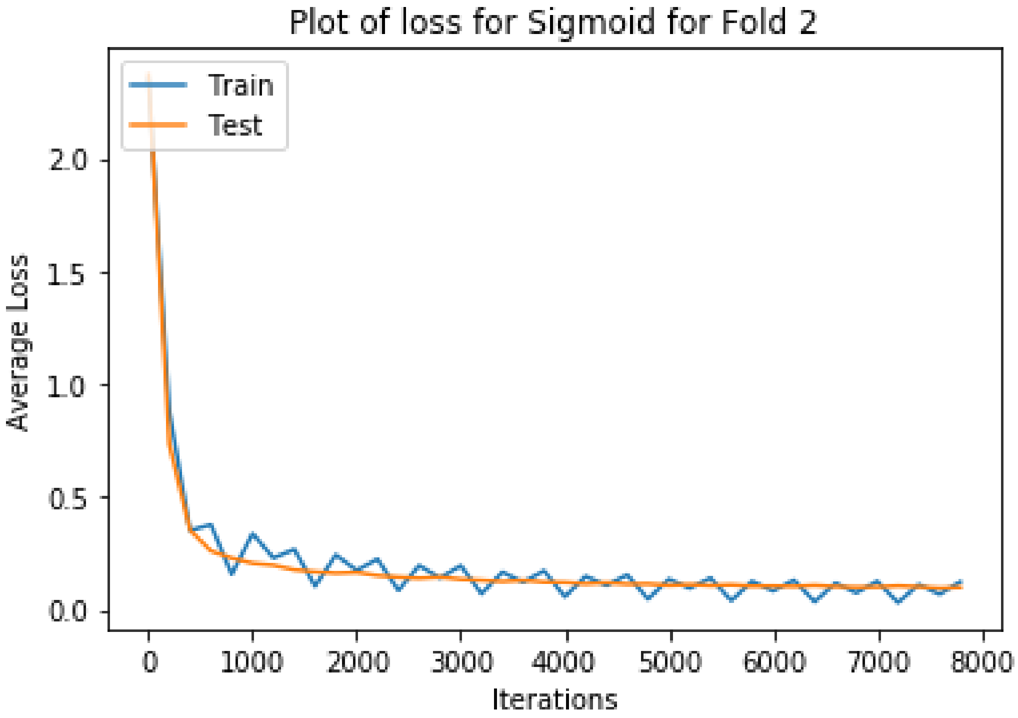


	0	1	2	3	4	5	6	7	8	9
0	1,198.0	1.0	2.0	1.0	0.0	0.0	4.0	1.0	3.0	3.0
1	1.0	1,304.0	7.0	1.0	2.0	0.0	2.0	3.0	2.0	3.0
2	2.0	3.0	1,163.0	13.0	1.0	0.0	2.0	6.0	3.0	0.0
3	0.0	0.0	11.0	1,219.0	0.0	7.0	0.0	1.0	5.0	3.0
4	3.0	1.0	3.0	0.0	1,084.0	1.0	5.0	15.0	4.0	27.0
5	0.0	0.0	3.0	20.0	1.0	1,036.0	13.0	0.0	7.0	7.0
6	2.0	1.0	0.0	0.0	1.0	5.0	1,143.0	0.0	2.0	0.0
7	1.0	1.0	16.0	10.0	11.0	0.0	0.0	1,180.0	2.0	27.0
8	5.0	7.0	10.0	24.0	1.0	8.0	12.0	2.0	1,099.0	20.0
9	2.0	0.0	1.0	6.0	3.0	1.0	1.0	13.0	7.0	1,169.0

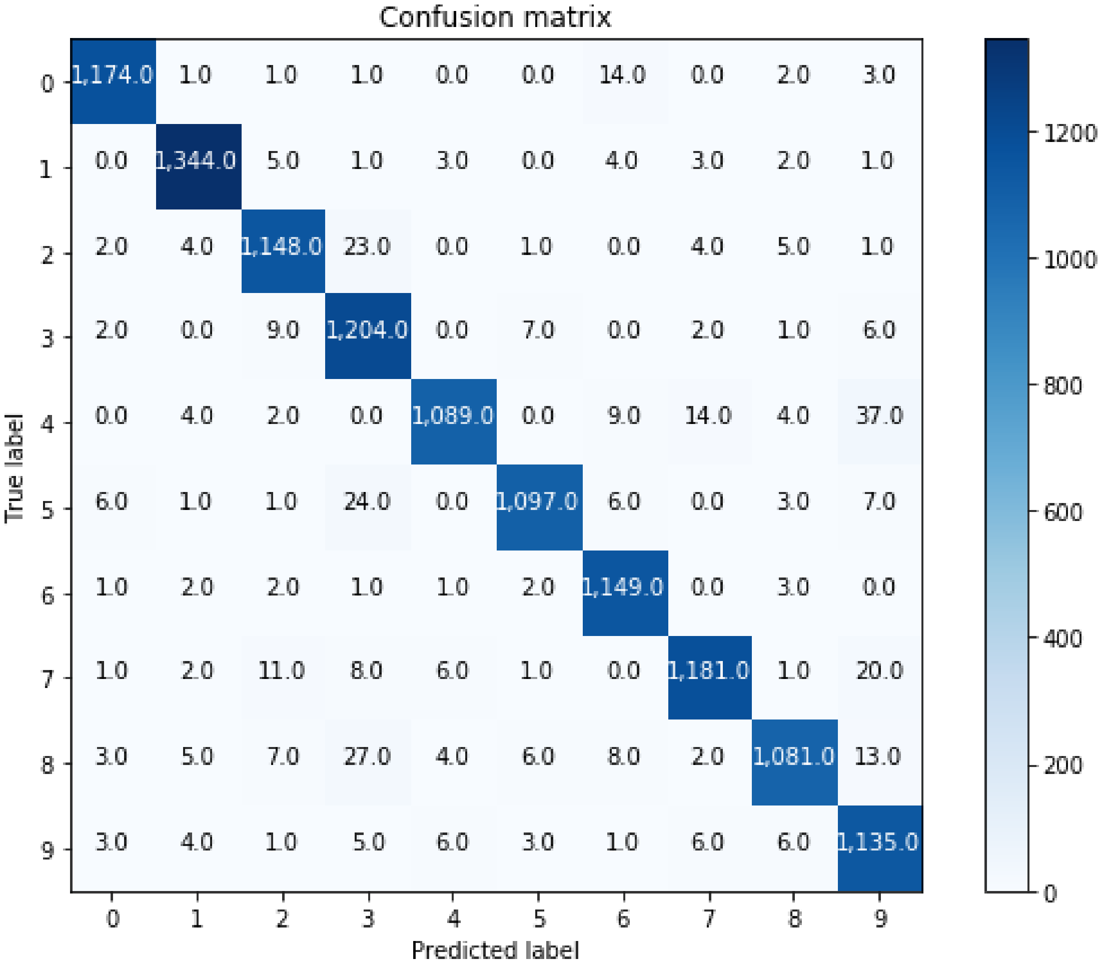
2.0.2 Sigmoid Activation: Fold - 2 , Learning Rate=1e-3

```
In [73]: model1=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=1, model=model1)
```

```
Epoch 1 Iteration 1 : Avg Loss = 2.3486130349224137
Epoch 1 Iteration 201 : Avg Loss = 0.8805077968073363
Epoch 1 Iteration 401 : Avg Loss = 0.35118200690820534
Epoch 1 Iteration 601 : Avg Loss = 0.3774719906840963
Epoch 2 Iteration 1 : Avg Loss = 0.15647418099205188
Epoch 2 Iteration 201 : Avg Loss = 0.33701204427867015
Epoch 2 Iteration 401 : Avg Loss = 0.2292529184510146
Epoch 2 Iteration 601 : Avg Loss = 0.2673917212471495
Epoch 3 Iteration 1 : Avg Loss = 0.10403024286388032
Epoch 3 Iteration 201 : Avg Loss = 0.2429156371271068
Epoch 3 Iteration 401 : Avg Loss = 0.17408084793344306
Epoch 3 Iteration 601 : Avg Loss = 0.2241918291408031
Epoch 4 Iteration 1 : Avg Loss = 0.08549408312668665
Epoch 4 Iteration 201 : Avg Loss = 0.1959961144193908
Epoch 4 Iteration 401 : Avg Loss = 0.14028176742571927
Epoch 4 Iteration 601 : Avg Loss = 0.1952009737202599
Epoch 5 Iteration 1 : Avg Loss = 0.07061118638707099
Epoch 5 Iteration 201 : Avg Loss = 0.1664451775125741
Epoch 5 Iteration 401 : Avg Loss = 0.12167676985414602
Epoch 5 Iteration 601 : Avg Loss = 0.17237019309312093
Epoch 6 Iteration 1 : Avg Loss = 0.05740647102003866
Epoch 6 Iteration 201 : Avg Loss = 0.14803157482959878
Epoch 6 Iteration 401 : Avg Loss = 0.10812726609504619
Epoch 6 Iteration 601 : Avg Loss = 0.1543118751711875
Epoch 7 Iteration 1 : Avg Loss = 0.04705788424598002
Epoch 7 Iteration 201 : Avg Loss = 0.13573084250629824
Epoch 7 Iteration 401 : Avg Loss = 0.09490796121066841
Epoch 7 Iteration 601 : Avg Loss = 0.1405772328128907
Epoch 8 Iteration 1 : Avg Loss = 0.040060624910670015
Epoch 8 Iteration 201 : Avg Loss = 0.12595932838728982
Epoch 8 Iteration 401 : Avg Loss = 0.08337007569707795
Epoch 8 Iteration 601 : Avg Loss = 0.1310908655700931
Epoch 9 Iteration 1 : Avg Loss = 0.03522286143880078
Epoch 9 Iteration 201 : Avg Loss = 0.11782556391561515
Epoch 9 Iteration 401 : Avg Loss = 0.07476002773884541
Epoch 9 Iteration 601 : Avg Loss = 0.12591232518534026
Epoch 10 Iteration 1 : Avg Loss = 0.031476027991174875
Epoch 10 Iteration 201 : Avg Loss = 0.11150040934074947
Epoch 10 Iteration 401 : Avg Loss = 0.06873461115827453
Epoch 10 Iteration 601 : Avg Loss = 0.12374684228922815
```



```
Accuracy = 0.9668333333333333
Precision =  [0.98489933 0.98317484 0.96714406 0.93044822 0.98196573 0.9820949
 0.96473552 0.97442244 0.97563177 0.92804579]
Recall =  [0.98160535 0.98606016 0.96632997 0.97806661 0.93960311 0.9580786
 0.98966408 0.95938262 0.93512111 0.97008547]
F1 Score =  [0.98324958 0.98461538 0.96673684 0.95366337 0.96031746 0.96993811
 0.97704082 0.96684404 0.954947   0.94860008]
```



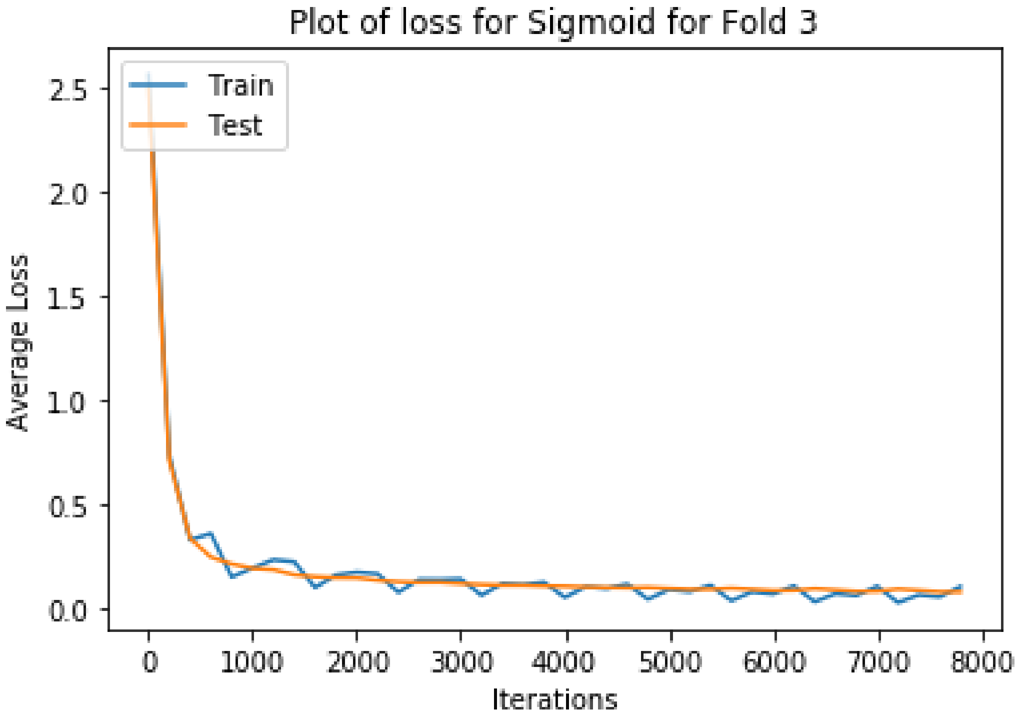
Confusion matrix

True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	1,174.0	1.0	1.0	1.0	0.0	0.0	14.0	0.0	2.0	3.0
1	0.0	1,344.0	5.0	1.0	3.0	0.0	4.0	3.0	2.0	1.0
2	2.0	4.0	1,148.0	23.0	0.0	1.0	0.0	4.0	5.0	1.0
3	2.0	0.0	9.0	1,204.0	0.0	7.0	0.0	2.0	1.0	6.0
4	0.0	4.0	2.0	0.0	1,089.0	0.0	9.0	14.0	4.0	37.0
5	6.0	1.0	1.0	24.0	0.0	1,097.0	6.0	0.0	3.0	7.0
6	1.0	2.0	2.0	1.0	1.0	2.0	1,149.0	0.0	3.0	0.0
7	1.0	2.0	11.0	8.0	6.0	1.0	0.0	1,181.0	1.0	20.0
8	3.0	5.0	7.0	27.0	4.0	6.0	8.0	2.0	1,081.0	13.0
9	3.0	4.0	1.0	5.0	6.0	3.0	1.0	6.0	6.0	1,135.0

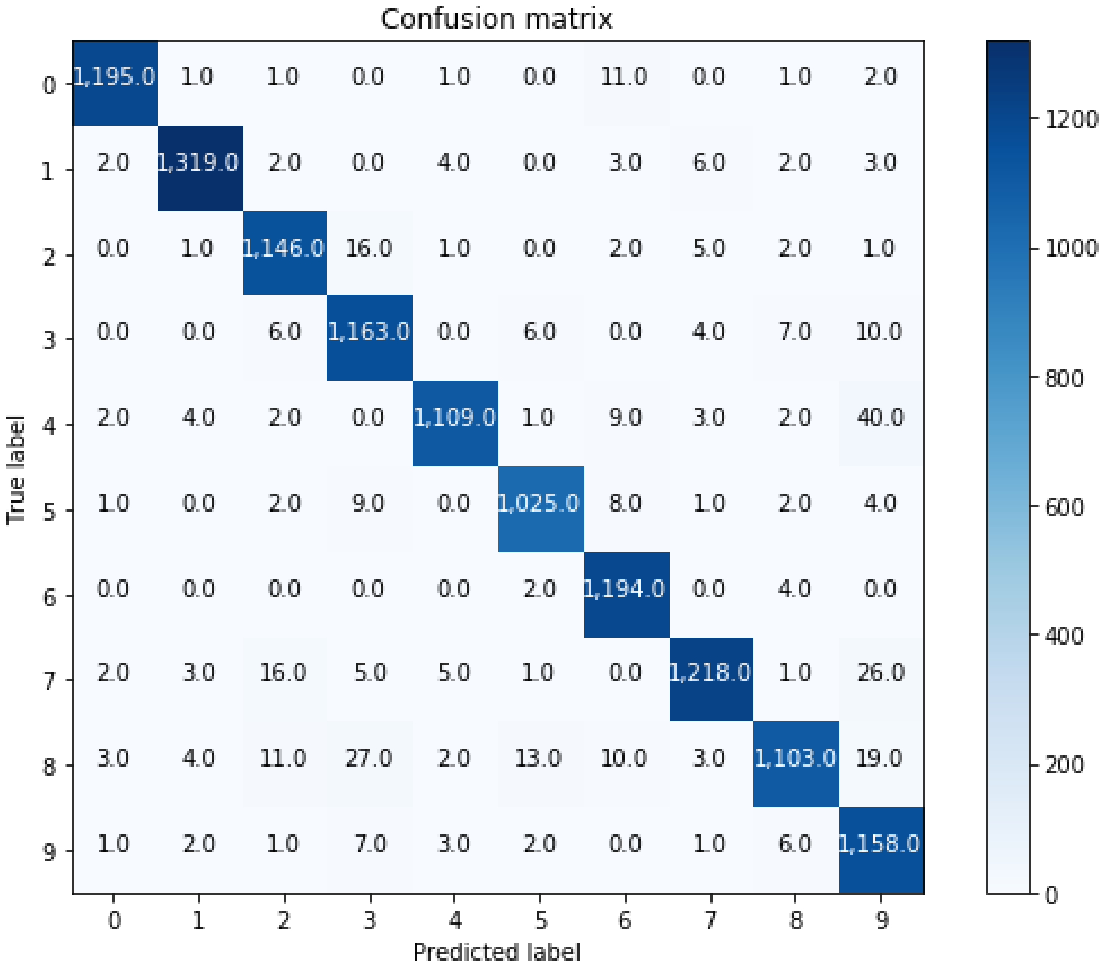
2.0.3 Sigmoid Activation: Fold - 3 , Learning Rate=1e-3

```
In [74]: model2=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=2, model=model2)
```

```
Epoch 1 Iteration 1 : Avg Loss = 2.5608770701074492
Epoch 1 Iteration 201 : Avg Loss = 0.7373994279409963
Epoch 1 Iteration 401 : Avg Loss = 0.3313299908022806
Epoch 1 Iteration 601 : Avg Loss = 0.36246301735684483
Epoch 2 Iteration 1 : Avg Loss = 0.15393990487147974
Epoch 2 Iteration 201 : Avg Loss = 0.19700579478168634
Epoch 2 Iteration 401 : Avg Loss = 0.23658798412097798
Epoch 2 Iteration 601 : Avg Loss = 0.2266668083979541
Epoch 3 Iteration 1 : Avg Loss = 0.10315344828157831
Epoch 3 Iteration 201 : Avg Loss = 0.16379690605442782
Epoch 3 Iteration 401 : Avg Loss = 0.1771364006606284
Epoch 3 Iteration 601 : Avg Loss = 0.16940896725257232
Epoch 4 Iteration 1 : Avg Loss = 0.08072841023500066
Epoch 4 Iteration 201 : Avg Loss = 0.14181660492921588
Epoch 4 Iteration 401 : Avg Loss = 0.14119408430178249
Epoch 4 Iteration 601 : Avg Loss = 0.1442354279861858
Epoch 5 Iteration 1 : Avg Loss = 0.0663022074929781
Epoch 5 Iteration 201 : Avg Loss = 0.12267082325345313
Epoch 5 Iteration 401 : Avg Loss = 0.11731573477474488
Epoch 5 Iteration 601 : Avg Loss = 0.12879216723723655
Epoch 6 Iteration 1 : Avg Loss = 0.055904690064533566
Epoch 6 Iteration 201 : Avg Loss = 0.1071905701190896
Epoch 6 Iteration 401 : Avg Loss = 0.09837409225715735
Epoch 6 Iteration 601 : Avg Loss = 0.11958445332605273
Epoch 7 Iteration 1 : Avg Loss = 0.047152172222009896
Epoch 7 Iteration 201 : Avg Loss = 0.09413525012848532
Epoch 7 Iteration 401 : Avg Loss = 0.08372216596109702
Epoch 7 Iteration 601 : Avg Loss = 0.11412888297040102
Epoch 8 Iteration 1 : Avg Loss = 0.04019930864665801
Epoch 8 Iteration 201 : Avg Loss = 0.08303096591017846
Epoch 8 Iteration 401 : Avg Loss = 0.07303894342702852
Epoch 8 Iteration 601 : Avg Loss = 0.11105829925024217
Epoch 9 Iteration 1 : Avg Loss = 0.03505150696744377
Epoch 9 Iteration 201 : Avg Loss = 0.07428206722388714
Epoch 9 Iteration 401 : Avg Loss = 0.06519559641680223
Epoch 9 Iteration 601 : Avg Loss = 0.10938989614345987
Epoch 10 Iteration 1 : Avg Loss = 0.031332867444667886
Epoch 10 Iteration 201 : Avg Loss = 0.0681132447318449
Epoch 10 Iteration 401 : Avg Loss = 0.058896756722482776
Epoch 10 Iteration 601 : Avg Loss = 0.1076279856585364
```



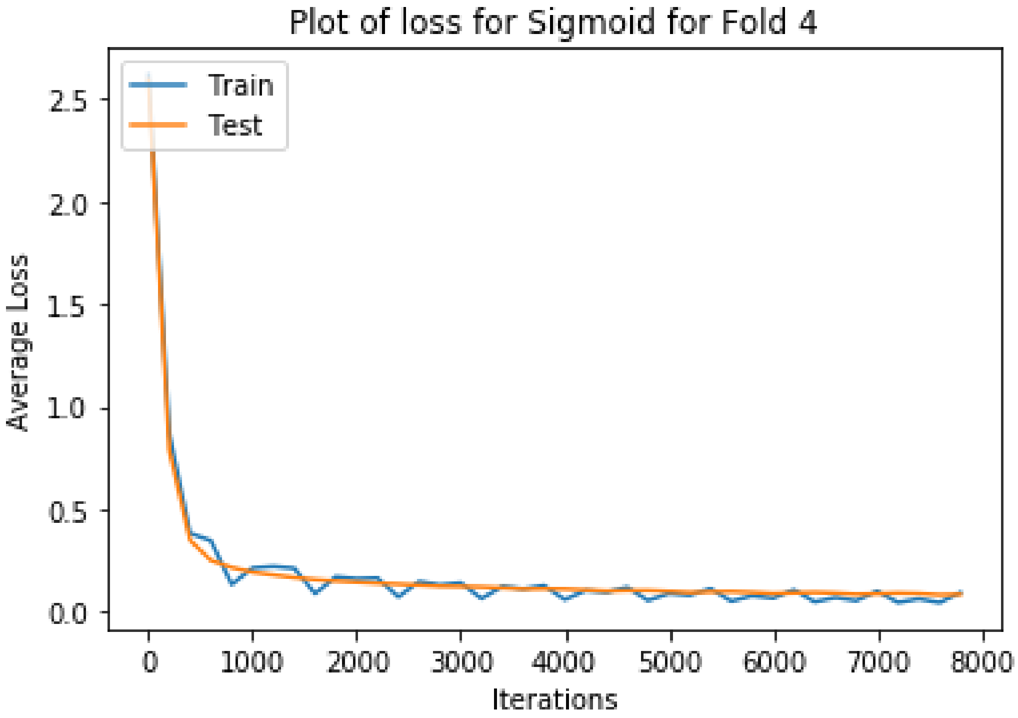
```
Accuracy = 0.9691666666666666
Precision =  [0.99087894 0.98875562 0.96545914 0.94784026 0.98577778 0.97619048
 0.96523848 0.98146656 0.97610619 0.91686461]
Recall =  [0.9859736  0.98359433 0.97614991 0.97240803 0.94624573 0.9743346
 0.995      0.95379796 0.92301255 0.98052498]
F1 Score =  [0.98842018 0.98616822 0.9707751  0.95996698 0.96560731 0.97526166
 0.97989331 0.96743447 0.9488172  0.94762684]
```



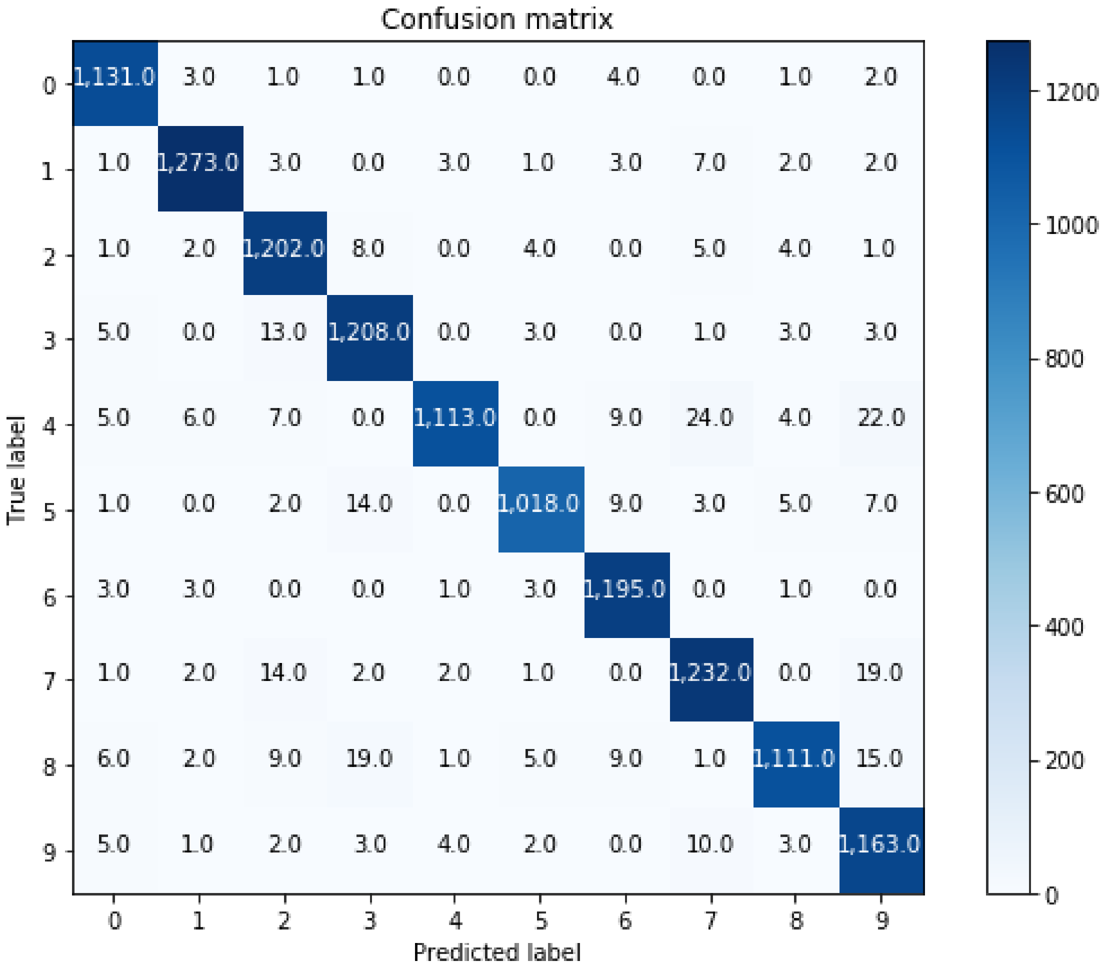
2.0.4 Sigmoid Activation: Fold - 4 , Learning Rate=1e-3

```
In [75]: model3=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=3, model=model3)
```

```
Epoch 1 Iteration 1 : Avg Loss = 2.61727318186808
Epoch 1 Iteration 201 : Avg Loss = 0.8704242156675674
Epoch 1 Iteration 401 : Avg Loss = 0.3838612418083182
Epoch 1 Iteration 601 : Avg Loss = 0.34807333461493795
Epoch 2 Iteration 1 : Avg Loss = 0.1340971846288544
Epoch 2 Iteration 201 : Avg Loss = 0.21459745950758458
Epoch 2 Iteration 401 : Avg Loss = 0.2222683176918635
Epoch 2 Iteration 601 : Avg Loss = 0.2132056718473819
Epoch 3 Iteration 1 : Avg Loss = 0.08956229127077667
Epoch 3 Iteration 201 : Avg Loss = 0.17164518048850474
Epoch 3 Iteration 401 : Avg Loss = 0.16195204657385218
Epoch 3 Iteration 601 : Avg Loss = 0.1658042036857238
Epoch 4 Iteration 1 : Avg Loss = 0.07350707623568369
Epoch 4 Iteration 201 : Avg Loss = 0.1488588781812995
Epoch 4 Iteration 401 : Avg Loss = 0.13192430967350516
Epoch 4 Iteration 601 : Avg Loss = 0.14221631945540786
Epoch 5 Iteration 1 : Avg Loss = 0.06533641702178208
Epoch 5 Iteration 201 : Avg Loss = 0.12490333848970356
Epoch 5 Iteration 401 : Avg Loss = 0.11246842146802903
Epoch 5 Iteration 601 : Avg Loss = 0.12876281625083752
Epoch 6 Iteration 1 : Avg Loss = 0.06024927838476302
Epoch 6 Iteration 201 : Avg Loss = 0.1052212343898589
Epoch 6 Iteration 401 : Avg Loss = 0.09660097093705501
Epoch 6 Iteration 601 : Avg Loss = 0.12060765625201145
Epoch 7 Iteration 1 : Avg Loss = 0.056257417432786436
Epoch 7 Iteration 201 : Avg Loss = 0.09054073301065518
Epoch 7 Iteration 401 : Avg Loss = 0.08167994378466077
Epoch 7 Iteration 601 : Avg Loss = 0.1141990344057979
Epoch 8 Iteration 1 : Avg Loss = 0.05274694745301829
Epoch 8 Iteration 201 : Avg Loss = 0.07935088781530081
Epoch 8 Iteration 401 : Avg Loss = 0.0677111509683827
Epoch 8 Iteration 601 : Avg Loss = 0.10807128947816369
Epoch 9 Iteration 1 : Avg Loss = 0.049575219308992016
Epoch 9 Iteration 201 : Avg Loss = 0.07078636261030061
Epoch 9 Iteration 401 : Avg Loss = 0.05549920232938613
Epoch 9 Iteration 601 : Avg Loss = 0.10155088564853891
Epoch 10 Iteration 1 : Avg Loss = 0.04644923545298721
Epoch 10 Iteration 201 : Avg Loss = 0.06440703772560766
Epoch 10 Iteration 401 : Avg Loss = 0.046037476503504635
Epoch 10 Iteration 601 : Avg Loss = 0.09473484911361188
```



```
Accuracy = 0.9705
Precision =  [0.97584124 0.98529412 0.95929769 0.9625498  0.99021352 0.98167792
 0.97233523 0.96024942 0.97971781 0.94246353]
Recall =  [0.98950131 0.98301158 0.9796251  0.97734628 0.93529412 0.96128423
 0.99087894 0.96779262 0.94312394 0.97485331]
F1 Score =  [0.98262381 0.98415153 0.96935484 0.96989161 0.96197061 0.97137405
 0.98151951 0.96400626 0.96107266 0.95838484]
```

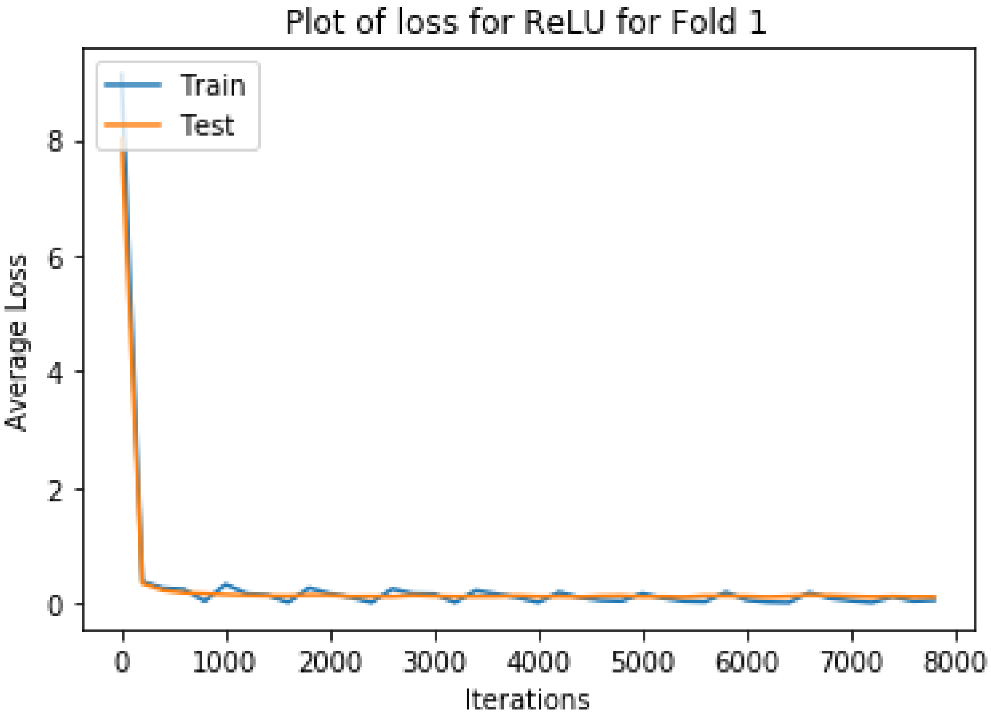


	0	1	2	3	4	5	6	7	8	9
0	1,131.0	3.0	1.0	1.0	0.0	0.0	4.0	0.0	1.0	2.0
1	1.0	1,273.0	3.0	0.0	3.0	1.0	3.0	7.0	2.0	2.0
2	1.0	2.0	1,202.0	8.0	0.0	4.0	0.0	5.0	4.0	1.0
3	5.0	0.0	13.0	1,208.0	0.0	3.0	0.0	1.0	3.0	3.0
4	5.0	6.0	7.0	0.0	1,113.0	0.0	9.0	24.0	4.0	22.0
5	1.0	0.0	2.0	14.0	0.0	1,018.0	9.0	3.0	5.0	7.0
6	3.0	3.0	0.0	0.0	1.0	3.0	1,195.0	0.0	1.0	0.0
7	1.0	2.0	14.0	2.0	2.0	1.0	0.0	1,232.0	0.0	19.0
8	6.0	2.0	9.0	19.0	1.0	5.0	9.0	1.0	1,111.0	15.0
9	5.0	1.0	2.0	3.0	4.0	2.0	0.0	10.0	3.0	1,163.0

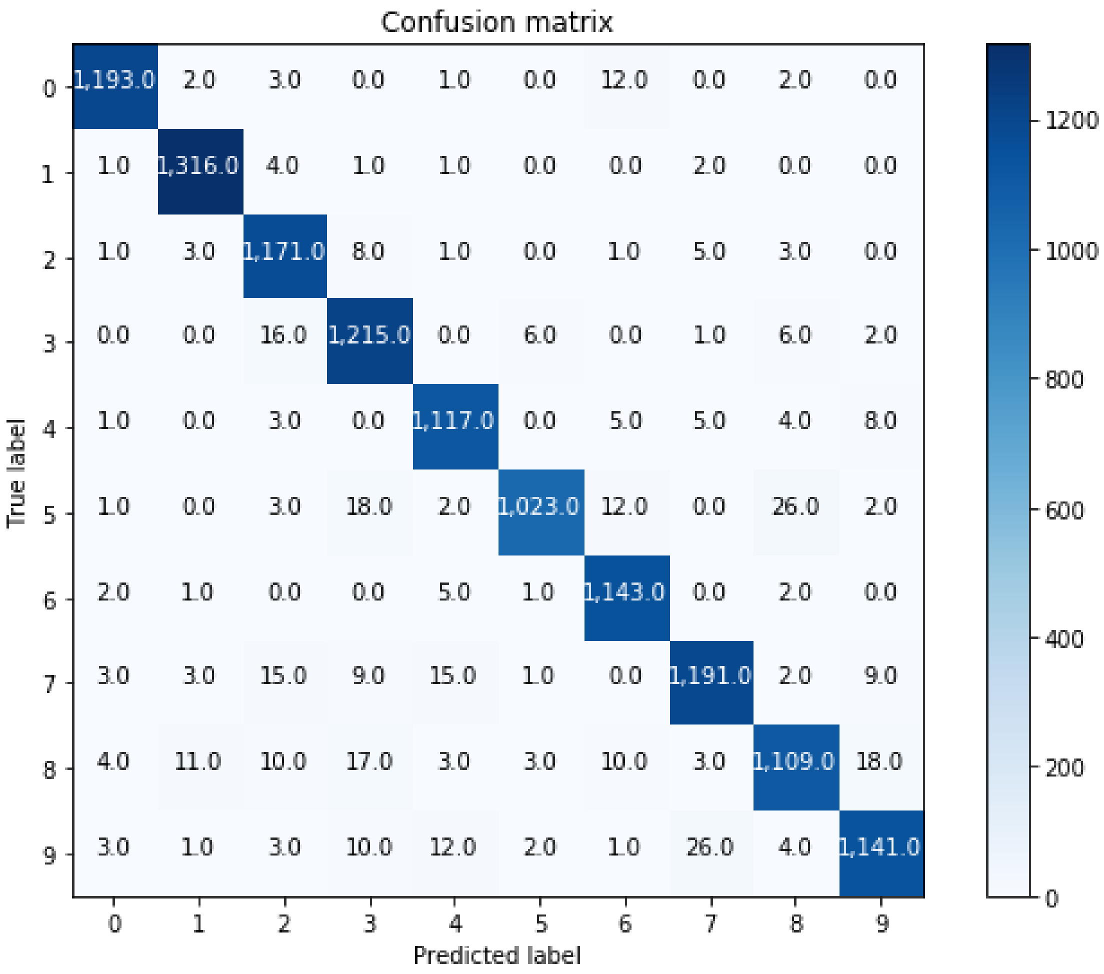
2.0.5 ReLU Activation: Fold - 1 , Learning Rate=5e-4

```
In [78]: model0=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=0, model=model0)
```

```
Epoch 1 Iteration 1 : Avg Loss = 9.120454114316132
Epoch 1 Iteration 201 : Avg Loss = 0.3695540096432324
Epoch 1 Iteration 401 : Avg Loss = 0.2563435714996035
Epoch 1 Iteration 601 : Avg Loss = 0.2329868392901371
Epoch 2 Iteration 1 : Avg Loss = 0.033891931897472014
Epoch 2 Iteration 201 : Avg Loss = 0.32103634967947414
Epoch 2 Iteration 401 : Avg Loss = 0.16219619921774103
Epoch 2 Iteration 601 : Avg Loss = 0.13502726609932056
Epoch 3 Iteration 1 : Avg Loss = 0.01103076294020759
Epoch 3 Iteration 201 : Avg Loss = 0.2586787284343916
Epoch 3 Iteration 401 : Avg Loss = 0.1622301980850681
Epoch 3 Iteration 601 : Avg Loss = 0.0983698625045581
Epoch 4 Iteration 1 : Avg Loss = 0.008943218464390768
Epoch 4 Iteration 201 : Avg Loss = 0.2421379000804832
Epoch 4 Iteration 401 : Avg Loss = 0.16164700315441405
Epoch 4 Iteration 601 : Avg Loss = 0.15484803320905666
Epoch 5 Iteration 1 : Avg Loss = 0.009118390863813414
Epoch 5 Iteration 201 : Avg Loss = 0.21582579941580404
Epoch 5 Iteration 401 : Avg Loss = 0.15499095557923986
Epoch 5 Iteration 601 : Avg Loss = 0.09493966806324963
Epoch 6 Iteration 1 : Avg Loss = 0.010490351099428197
Epoch 6 Iteration 201 : Avg Loss = 0.18654996837916796
Epoch 6 Iteration 401 : Avg Loss = 0.09390297338679306
Epoch 6 Iteration 601 : Avg Loss = 0.046540519319866865
Epoch 7 Iteration 1 : Avg Loss = 0.03453948600237958
Epoch 7 Iteration 201 : Avg Loss = 0.16441314588527928
Epoch 7 Iteration 401 : Avg Loss = 0.0819965470904304
Epoch 7 Iteration 601 : Avg Loss = 0.032412598727772135
Epoch 8 Iteration 1 : Avg Loss = 0.021400701645603367
Epoch 8 Iteration 201 : Avg Loss = 0.18457018942571835
Epoch 8 Iteration 401 : Avg Loss = 0.05032321096995107
Epoch 8 Iteration 601 : Avg Loss = 0.015144899475195855
Epoch 9 Iteration 1 : Avg Loss = 0.006296161301422975
Epoch 9 Iteration 201 : Avg Loss = 0.17762634986350748
Epoch 9 Iteration 401 : Avg Loss = 0.08052973097848583
Epoch 9 Iteration 601 : Avg Loss = 0.03981868745681544
Epoch 10 Iteration 1 : Avg Loss = 0.007603637702699635
Epoch 10 Iteration 201 : Avg Loss = 0.10729036699606607
Epoch 10 Iteration 401 : Avg Loss = 0.03250305093712224
Epoch 10 Iteration 601 : Avg Loss = 0.0492099029442807
```



```
Accuracy = 0.96825
Precision =  [0.98676592 0.98429319 0.95358306 0.95070423 0.96542783 0.98745174
 0.96537162 0.96593674 0.95768566 0.96694915]
Recall =  [0.98351195 0.99320755 0.98155909 0.97512039 0.97725284 0.94112236
 0.99046794 0.95432692 0.93350168 0.94846218]
F1 Score =  [0.98513625 0.98873028 0.96736886 0.96275753 0.97130435 0.96373057
 0.97775877 0.96009674 0.94543905 0.95761645]
```

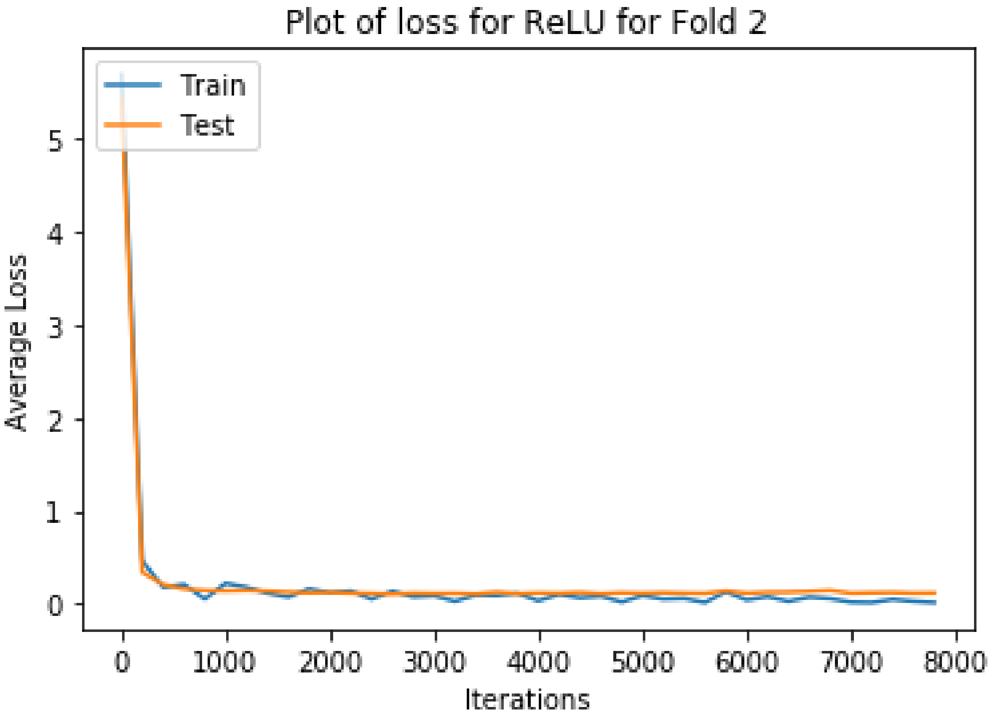


	0	1	2	3	4	5	6	7	8	9
0	1,193.0	2.0	3.0	0.0	1.0	0.0	12.0	0.0	2.0	0.0
1	1.0	1,316.0	4.0	1.0	1.0	0.0	0.0	2.0	0.0	0.0
2	1.0	3.0	1,171.0	8.0	1.0	0.0	1.0	5.0	3.0	0.0
3	0.0	0.0	16.0	1,215.0	0.0	6.0	0.0	1.0	6.0	2.0
4	1.0	0.0	3.0	0.0	1,117.0	0.0	5.0	5.0	4.0	8.0
5	1.0	0.0	3.0	18.0	2.0	1,023.0	12.0	0.0	26.0	2.0
6	2.0	1.0	0.0	0.0	5.0	1.0	1,143.0	0.0	2.0	0.0
7	3.0	3.0	15.0	9.0	15.0	1.0	0.0	1,191.0	2.0	9.0
8	4.0	11.0	10.0	17.0	3.0	3.0	10.0	3.0	1,109.0	18.0
9	3.0	1.0	3.0	10.0	12.0	2.0	1.0	26.0	4.0	1,141.0

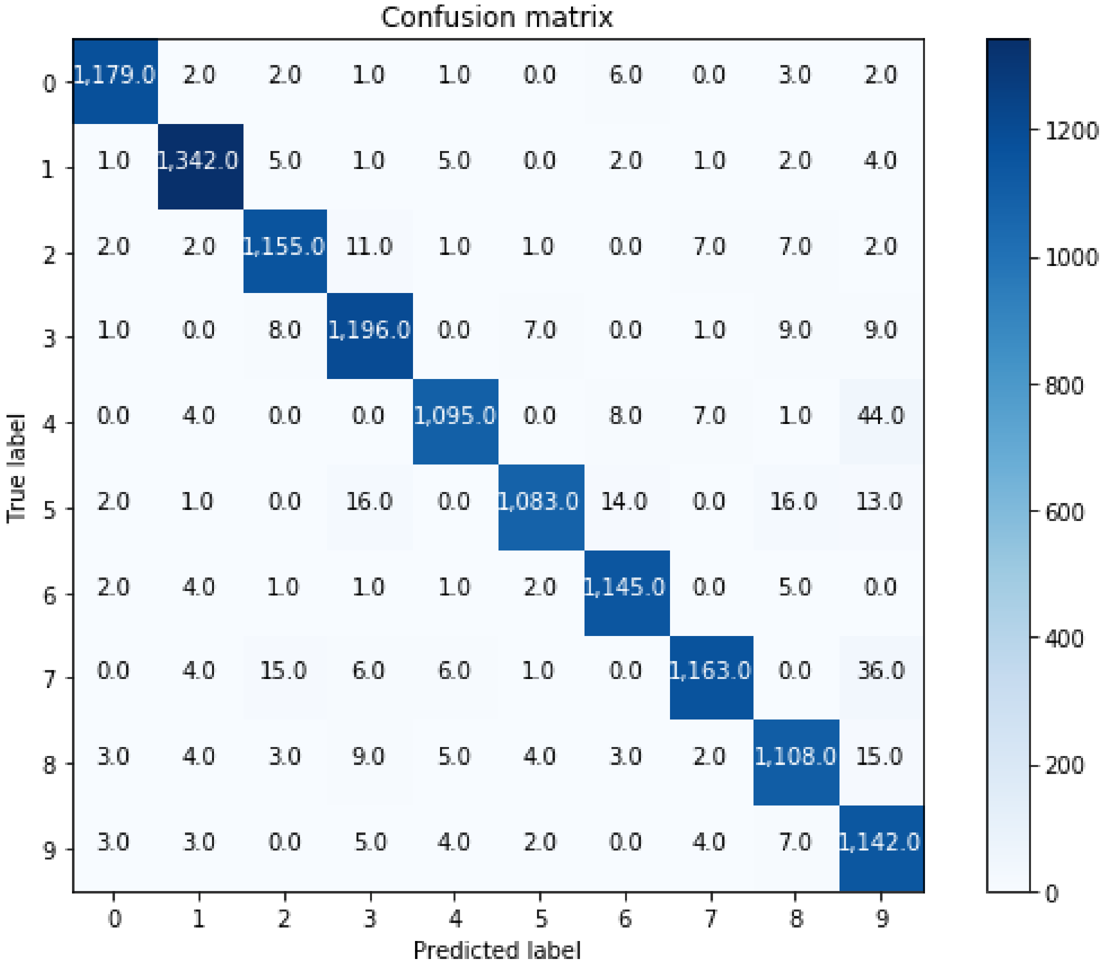
2.0.6 ReLU Activation: Fold - 2 , Learning Rate=5e-4

```
In [79]: model1=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=1, model=model1)
```

```
Epoch 1 Iteration 1 : Avg Loss = 5.686153496418004
Epoch 1 Iteration 201 : Avg Loss = 0.4629774230970316
Epoch 1 Iteration 401 : Avg Loss = 0.17903476425572615
Epoch 1 Iteration 601 : Avg Loss = 0.21012480867141517
Epoch 2 Iteration 1 : Avg Loss = 0.05348664507058814
Epoch 2 Iteration 201 : Avg Loss = 0.22230705885393404
Epoch 2 Iteration 401 : Avg Loss = 0.1792182683996405
Epoch 2 Iteration 601 : Avg Loss = 0.11641789411432661
Epoch 3 Iteration 1 : Avg Loss = 0.07433463501481997
Epoch 3 Iteration 201 : Avg Loss = 0.16261710529705697
Epoch 3 Iteration 401 : Avg Loss = 0.12071240316830212
Epoch 3 Iteration 601 : Avg Loss = 0.13939150151652466
Epoch 4 Iteration 1 : Avg Loss = 0.052717498876482194
Epoch 4 Iteration 201 : Avg Loss = 0.13673239208360083
Epoch 4 Iteration 401 : Avg Loss = 0.07667629642699361
Epoch 4 Iteration 601 : Avg Loss = 0.08814960443377175
Epoch 5 Iteration 1 : Avg Loss = 0.02887043707787417
Epoch 5 Iteration 201 : Avg Loss = 0.09594533241439655
Epoch 5 Iteration 401 : Avg Loss = 0.09157138607078297
Epoch 5 Iteration 601 : Avg Loss = 0.11366887943828569
Epoch 6 Iteration 1 : Avg Loss = 0.034516186806697255
Epoch 6 Iteration 201 : Avg Loss = 0.10210740009288341
Epoch 6 Iteration 401 : Avg Loss = 0.06621207463419
Epoch 6 Iteration 601 : Avg Loss = 0.08332966772978152
Epoch 7 Iteration 1 : Avg Loss = 0.021335071446704185
Epoch 7 Iteration 201 : Avg Loss = 0.09011580377665288
Epoch 7 Iteration 401 : Avg Loss = 0.05149619979238955
Epoch 7 Iteration 601 : Avg Loss = 0.05902780063687792
Epoch 8 Iteration 1 : Avg Loss = 0.015797329020319446
Epoch 8 Iteration 201 : Avg Loss = 0.13103298408229472
Epoch 8 Iteration 401 : Avg Loss = 0.0426423078357665
Epoch 8 Iteration 601 : Avg Loss = 0.07821632826994868
Epoch 9 Iteration 1 : Avg Loss = 0.028729327741394466
Epoch 9 Iteration 201 : Avg Loss = 0.07311715606131325
Epoch 9 Iteration 401 : Avg Loss = 0.057211775680836345
Epoch 9 Iteration 601 : Avg Loss = 0.019621590543342525
Epoch 10 Iteration 1 : Avg Loss = 0.015075036365580278
Epoch 10 Iteration 201 : Avg Loss = 0.04951006118409535
Epoch 10 Iteration 401 : Avg Loss = 0.028911340485648574
Epoch 10 Iteration 601 : Avg Loss = 0.015299236908133037
```



```
Accuracy = 0.9673333333333334
Precision =  [0.98826488 0.98243045 0.97140454 0.95987159 0.97942755 0.98454545
 0.97198642 0.9814346  0.95682211 0.90134175]
Recall =  [0.98578595 0.98459281 0.97222222 0.97156783 0.94477998 0.94585153
 0.98621878 0.94476036 0.95847751 0.97606838]
F1 Score =  [0.98702386 0.98351044 0.97181321 0.9656843  0.96179183 0.96481069
 0.97905088 0.96274834 0.95764909 0.93721789]
```

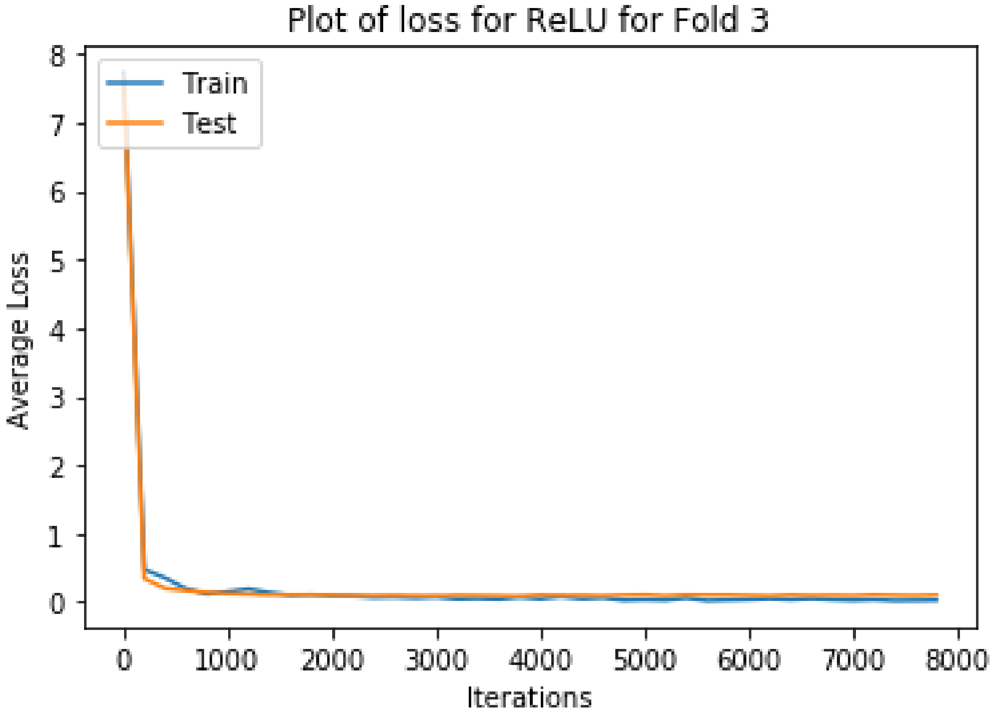


	0	1	2	3	4	5	6	7	8	9
0	1,179.0	2.0	2.0	1.0	1.0	0.0	6.0	0.0	3.0	2.0
1	1.0	1,342.0	5.0	1.0	5.0	0.0	2.0	1.0	2.0	4.0
2	2.0	2.0	1,155.0	11.0	1.0	1.0	0.0	7.0	7.0	2.0
3	1.0	0.0	8.0	1,196.0	0.0	7.0	0.0	1.0	9.0	9.0
4	0.0	4.0	0.0	0.0	1,095.0	0.0	8.0	7.0	1.0	44.0
5	2.0	1.0	0.0	16.0	0.0	1,083.0	14.0	0.0	16.0	13.0
6	2.0	4.0	1.0	1.0	1.0	2.0	1,145.0	0.0	5.0	0.0
7	0.0	4.0	15.0	6.0	6.0	1.0	0.0	1,163.0	0.0	36.0
8	3.0	4.0	3.0	9.0	5.0	4.0	3.0	2.0	1,108.0	15.0
9	3.0	3.0	0.0	5.0	4.0	2.0	0.0	4.0	7.0	1,142.0

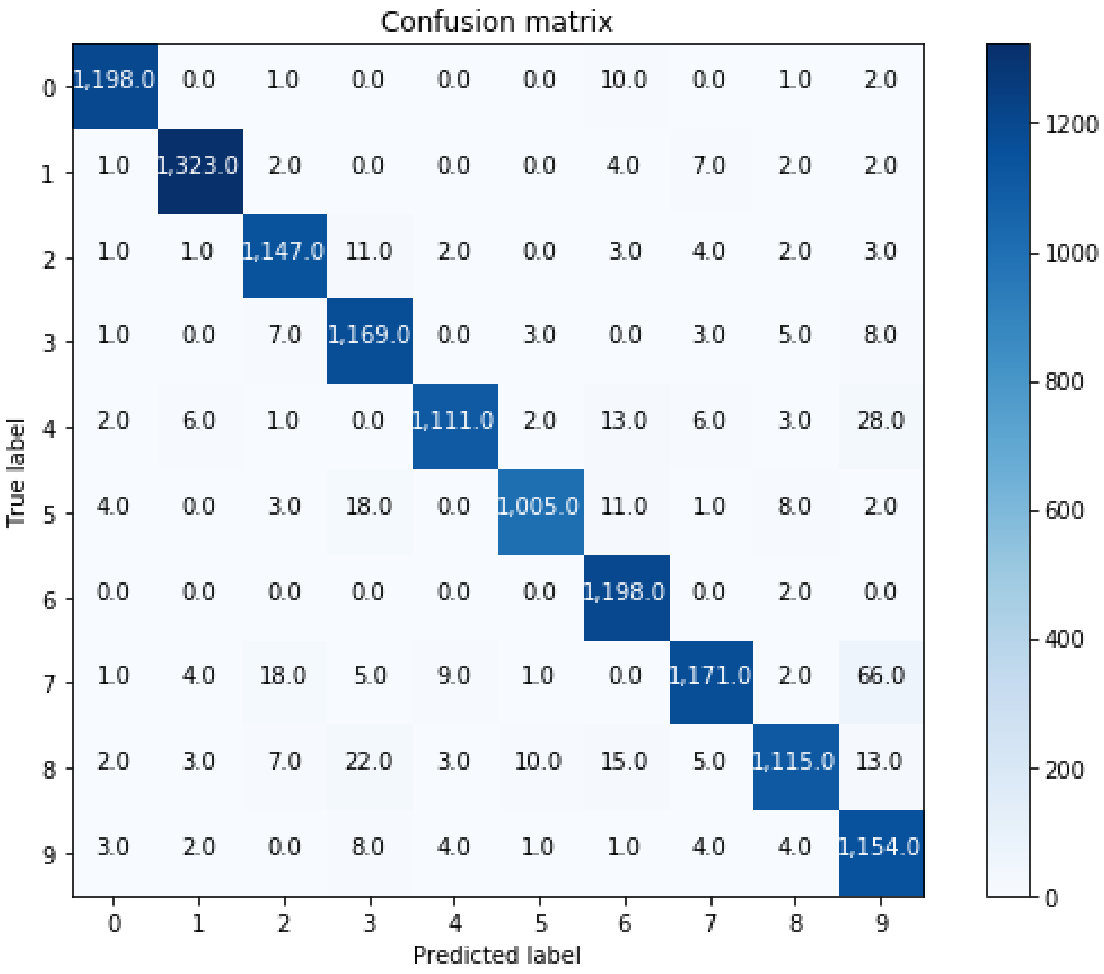
2.0.7 ReLU Activation: Fold - 3 , Learning Rate=5e-4

```
In [80]: model2=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=2, model=model2)
```

```
Epoch 1 Iteration 1 : Avg Loss = 7.7289568417855055
Epoch 1 Iteration 201 : Avg Loss = 0.4767908575414811
Epoch 1 Iteration 401 : Avg Loss = 0.3539230563369344
Epoch 1 Iteration 601 : Avg Loss = 0.19498260146854265
Epoch 2 Iteration 1 : Avg Loss = 0.11988889014426031
Epoch 2 Iteration 201 : Avg Loss = 0.15201172364305393
Epoch 2 Iteration 401 : Avg Loss = 0.1918882570027483
Epoch 2 Iteration 601 : Avg Loss = 0.1381031830695138
Epoch 3 Iteration 1 : Avg Loss = 0.10103946291874161
Epoch 3 Iteration 201 : Avg Loss = 0.10967983019753948
Epoch 3 Iteration 401 : Avg Loss = 0.09412550603307032
Epoch 3 Iteration 601 : Avg Loss = 0.08914121278336554
Epoch 4 Iteration 1 : Avg Loss = 0.07045661571137758
Epoch 4 Iteration 201 : Avg Loss = 0.07562473876215411
Epoch 4 Iteration 401 : Avg Loss = 0.0635467723364539
Epoch 4 Iteration 601 : Avg Loss = 0.07559270580111976
Epoch 5 Iteration 1 : Avg Loss = 0.04826412348498155
Epoch 5 Iteration 201 : Avg Loss = 0.05572940869537378
Epoch 5 Iteration 401 : Avg Loss = 0.04486971997321076
Epoch 5 Iteration 601 : Avg Loss = 0.07322558322903758
Epoch 6 Iteration 1 : Avg Loss = 0.0504528623033276
Epoch 6 Iteration 201 : Avg Loss = 0.0810150363080389
Epoch 6 Iteration 401 : Avg Loss = 0.051240136667419355
Epoch 6 Iteration 601 : Avg Loss = 0.06541453024892892
Epoch 7 Iteration 1 : Avg Loss = 0.028444536923683754
Epoch 7 Iteration 201 : Avg Loss = 0.03727082052452081
Epoch 7 Iteration 401 : Avg Loss = 0.030093062872504614
Epoch 7 Iteration 601 : Avg Loss = 0.06203271373423305
Epoch 8 Iteration 1 : Avg Loss = 0.024002034205703626
Epoch 8 Iteration 201 : Avg Loss = 0.03248176947002494
Epoch 8 Iteration 401 : Avg Loss = 0.03921060631717962
Epoch 8 Iteration 601 : Avg Loss = 0.052338320996058875
Epoch 9 Iteration 1 : Avg Loss = 0.03871390241072076
Epoch 9 Iteration 201 : Avg Loss = 0.05260957846975152
Epoch 9 Iteration 401 : Avg Loss = 0.03800104195705883
Epoch 9 Iteration 601 : Avg Loss = 0.02829026006163452
Epoch 10 Iteration 1 : Avg Loss = 0.03762736967892924
Epoch 10 Iteration 201 : Avg Loss = 0.022726322083105755
Epoch 10 Iteration 401 : Avg Loss = 0.025565760971025427
Epoch 10 Iteration 601 : Avg Loss = 0.02957233267804254
```



```
Accuracy = 0.9659166666666666
Precision =  [0.98763397 0.98805078 0.96711636 0.94809408 0.98405669 0.98336595
 0.95458167 0.97502082 0.97465035 0.9029734 ]
Recall =  [0.98844884 0.98657718 0.9770017  0.97742475 0.94795222 0.95532319
 0.99833333 0.91699295 0.93305439 0.97713802]
F1 Score =  [0.98804124 0.98731343 0.9720339  0.96253602 0.9656671  0.96914176
 0.97596741 0.94511703 0.95339889 0.93859292]
```

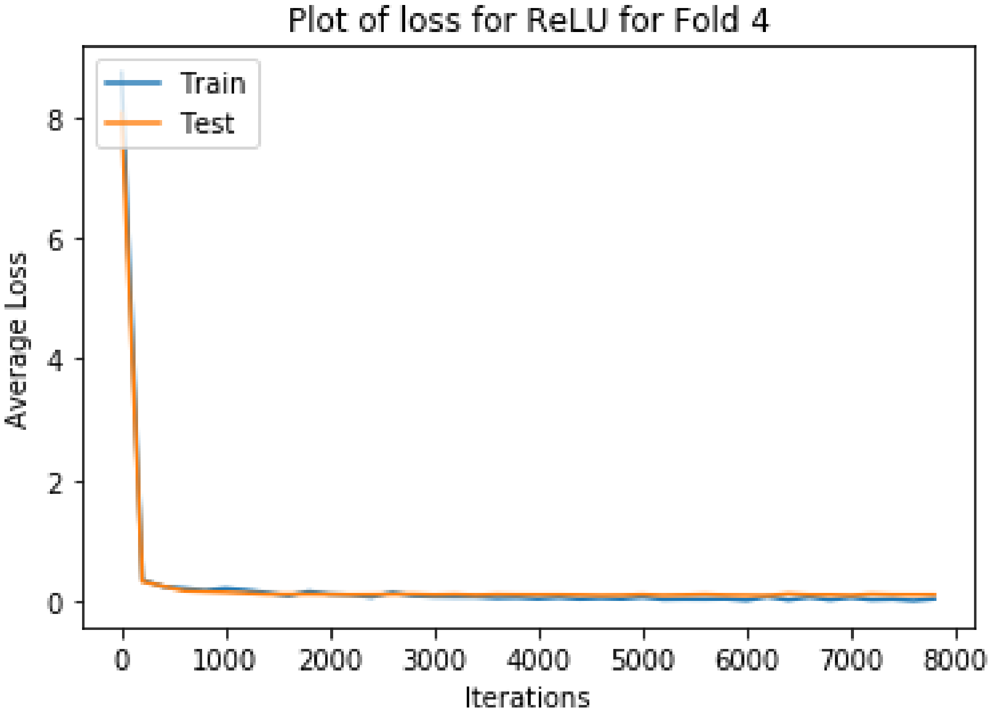


	0	1	2	3	4	5	6	7	8	9
0	1,198.0	0.0	1.0	0.0	0.0	0.0	10.0	0.0	1.0	2.0
1	1.0	1,323.0	2.0	0.0	0.0	0.0	4.0	7.0	2.0	2.0
2	1.0	1.0	1,147.0	11.0	2.0	0.0	3.0	4.0	2.0	3.0
3	1.0	0.0	7.0	1,169.0	0.0	3.0	0.0	3.0	5.0	8.0
4	2.0	6.0	1.0	0.0	1,111.0	2.0	13.0	6.0	3.0	28.0
5	4.0	0.0	3.0	18.0	0.0	1,005.0	11.0	1.0	8.0	2.0
6	0.0	0.0	0.0	0.0	0.0	0.0	1,198.0	0.0	2.0	0.0
7	1.0	4.0	18.0	5.0	9.0	1.0	0.0	1,171.0	2.0	66.0
8	2.0	3.0	7.0	22.0	3.0	10.0	15.0	5.0	1,115.0	13.0
9	3.0	2.0	0.0	8.0	4.0	1.0	1.0	4.0	4.0	1,154.0

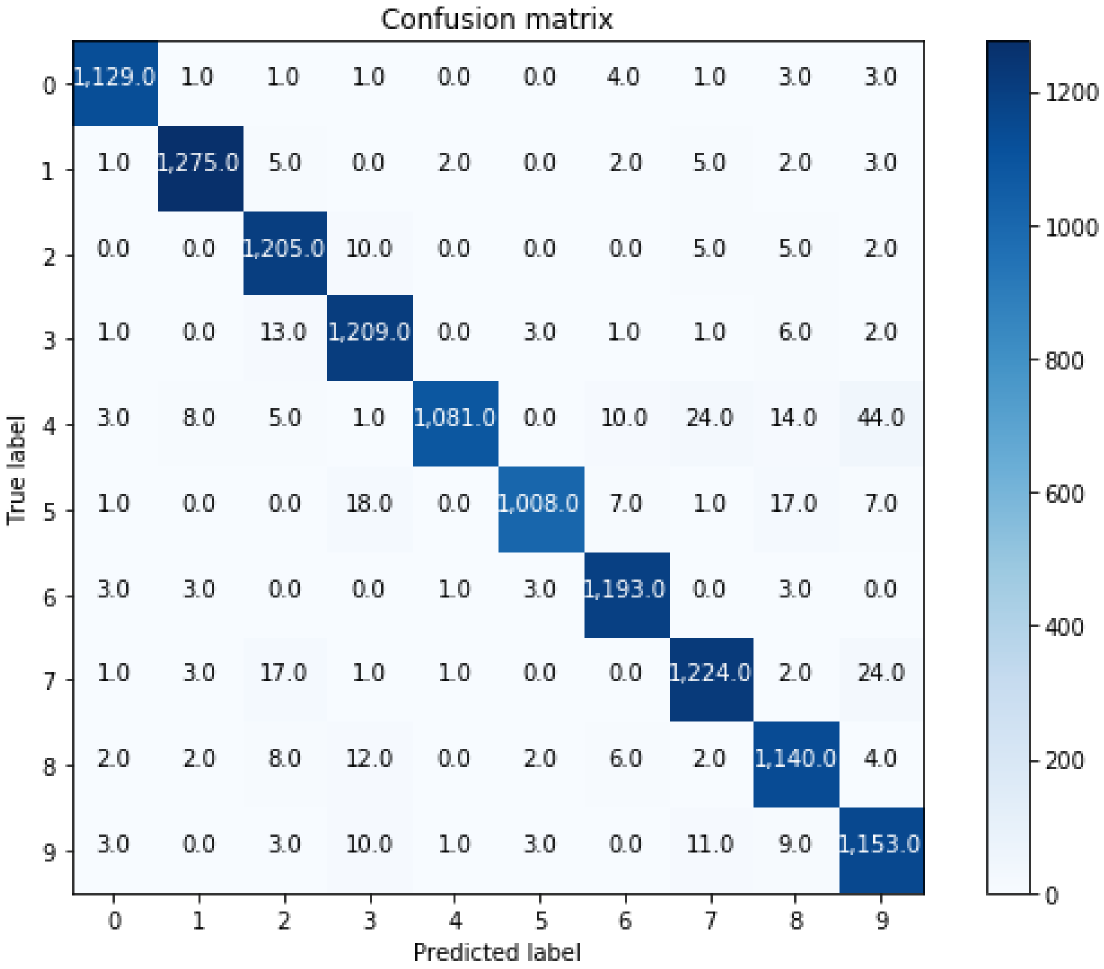
2.0.8 ReLU Activation: Fold - 4 , Learning Rate=5e-4

```
In [81]: model3=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=3, model=model3)
```

```
Epoch 1 Iteration 1 : Avg Loss = 8.73148664539806
Epoch 1 Iteration 201 : Avg Loss = 0.3428013177751035
Epoch 1 Iteration 401 : Avg Loss = 0.23369188384478579
Epoch 1 Iteration 601 : Avg Loss = 0.2042677366311289
Epoch 2 Iteration 1 : Avg Loss = 0.17110737835237658
Epoch 2 Iteration 201 : Avg Loss = 0.20369544499754108
Epoch 2 Iteration 401 : Avg Loss = 0.17257317536873873
Epoch 2 Iteration 601 : Avg Loss = 0.13164172663356488
Epoch 3 Iteration 1 : Avg Loss = 0.08116140438953583
Epoch 3 Iteration 201 : Avg Loss = 0.15247239997726222
Epoch 3 Iteration 401 : Avg Loss = 0.10891679864831445
Epoch 3 Iteration 601 : Avg Loss = 0.0962537658825236
Epoch 4 Iteration 1 : Avg Loss = 0.0628560789125494
Epoch 4 Iteration 201 : Avg Loss = 0.13475861205303025
Epoch 4 Iteration 401 : Avg Loss = 0.08095876305624854
Epoch 4 Iteration 601 : Avg Loss = 0.06913590105589297
Epoch 5 Iteration 1 : Avg Loss = 0.06665181353210972
Epoch 5 Iteration 201 : Avg Loss = 0.0662147601957997
Epoch 5 Iteration 401 : Avg Loss = 0.04630521229031184
Epoch 5 Iteration 601 : Avg Loss = 0.05317881752069942
Epoch 6 Iteration 1 : Avg Loss = 0.03444957516862797
Epoch 6 Iteration 201 : Avg Loss = 0.053649075961792816
Epoch 6 Iteration 401 : Avg Loss = 0.03012052388546444
Epoch 6 Iteration 601 : Avg Loss = 0.043634725720439915
Epoch 7 Iteration 1 : Avg Loss = 0.031534681066791005
Epoch 7 Iteration 201 : Avg Loss = 0.058949691252450684
Epoch 7 Iteration 401 : Avg Loss = 0.023746311551862208
Epoch 7 Iteration 601 : Avg Loss = 0.02947613191008299
Epoch 8 Iteration 1 : Avg Loss = 0.028301773012016883
Epoch 8 Iteration 201 : Avg Loss = 0.03299403901221629
Epoch 8 Iteration 401 : Avg Loss = 0.008379397744295641
Epoch 8 Iteration 601 : Avg Loss = 0.06611256263442201
Epoch 9 Iteration 1 : Avg Loss = 0.0140484036381062
Epoch 9 Iteration 201 : Avg Loss = 0.05860864870432068
Epoch 9 Iteration 401 : Avg Loss = 0.016484641139775404
Epoch 9 Iteration 601 : Avg Loss = 0.05300744201430871
Epoch 10 Iteration 1 : Avg Loss = 0.017310534388658882
Epoch 10 Iteration 201 : Avg Loss = 0.02800197666738998
Epoch 10 Iteration 401 : Avg Loss = 0.0027657788361300883
Epoch 10 Iteration 601 : Avg Loss = 0.032562322922983486
```



```
Accuracy = 0.9680833333333333
Precision =  [0.98688811 0.98684211 0.95863166 0.95800317 0.99539595 0.9892051
 0.97547016 0.96075353 0.94920899 0.92834138]
Recall =  [0.98775153 0.98455598 0.98207009 0.97815534 0.90840336 0.95184136
 0.98922056 0.96150825 0.96774194 0.96647108]
F1 Score =  [0.98731963 0.98569772 0.97020934 0.96797438 0.94991213 0.97016362
 0.98229724 0.96113074 0.95838588 0.94702259]
```

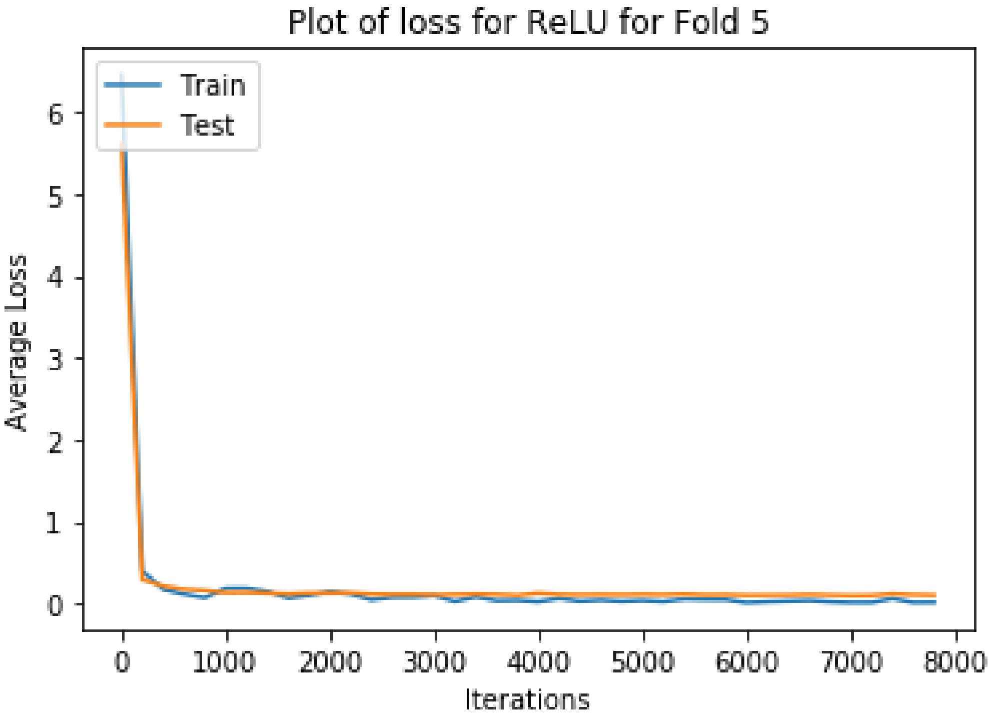


	0	1	2	3	4	5	6	7	8	9
0	1,129.0	1.0	1.0	1.0	0.0	0.0	4.0	1.0	3.0	3.0
1	1.0	1,275.0	5.0	0.0	2.0	0.0	2.0	5.0	2.0	3.0
2	0.0	0.0	1,205.0	10.0	0.0	0.0	0.0	5.0	5.0	2.0
3	1.0	0.0	13.0	1,209.0	0.0	3.0	1.0	1.0	6.0	2.0
4	3.0	8.0	5.0	1.0	1,081.0	0.0	10.0	24.0	14.0	44.0
5	1.0	0.0	0.0	18.0	0.0	1,008.0	7.0	1.0	17.0	7.0
6	3.0	3.0	0.0	0.0	1.0	3.0	1,193.0	0.0	3.0	0.0
7	1.0	3.0	17.0	1.0	1.0	0.0	0.0	1,224.0	2.0	24.0
8	2.0	2.0	8.0	12.0	0.0	2.0	6.0	2.0	1,140.0	4.0
9	3.0	0.0	3.0	10.0	1.0	3.0	0.0	11.0	9.0	1,153.0

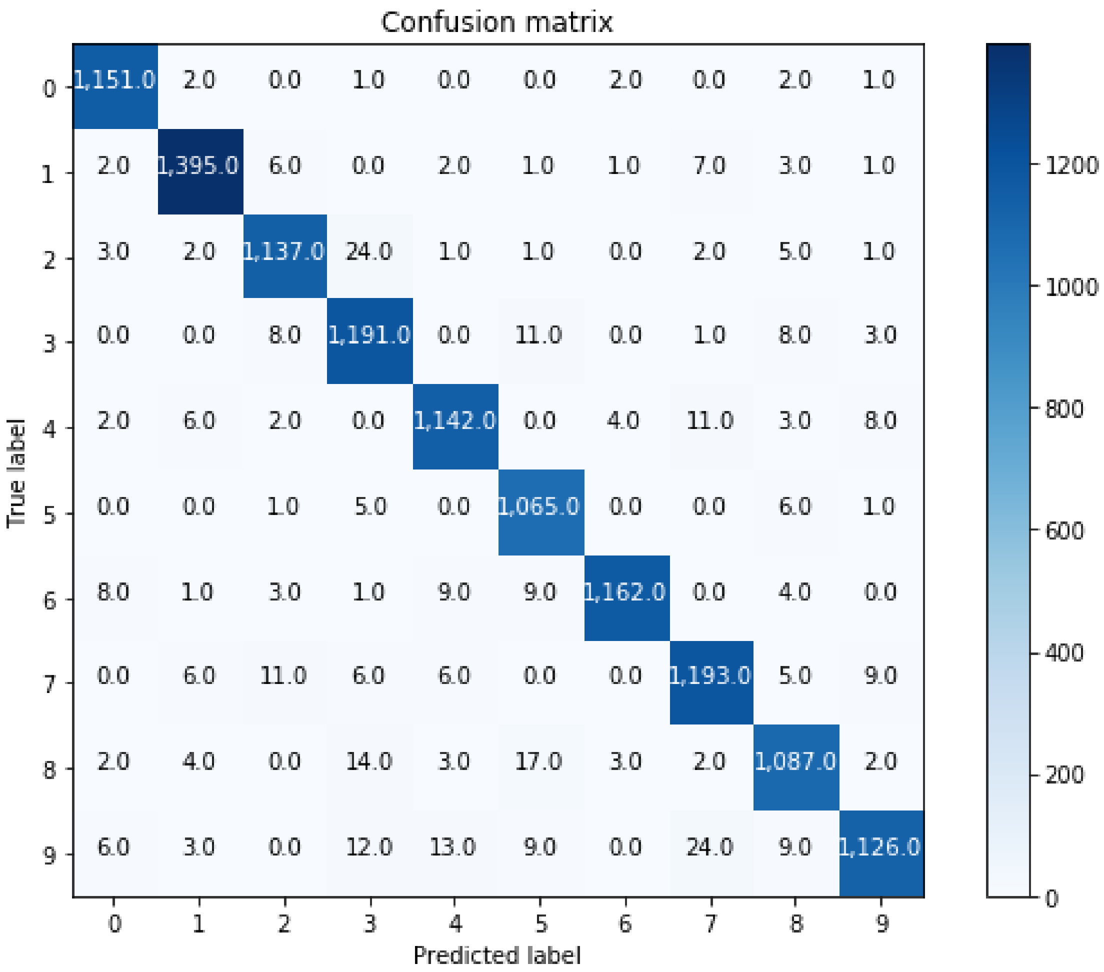
2.0.9 ReLU Activation: Fold - 5 , Learning Rate=5e-4

```
In [82]: model4=MLP(input_size, h1_size, h2_size, output_size)
         train(images=hog_img, labels=labels, epochs=10, fold_index=4, model=model4)
```

```
Epoch 1 Iteration 1 : Avg Loss = 6.452419998397894
Epoch 1 Iteration 201 : Avg Loss = 0.40023061029722135
Epoch 1 Iteration 401 : Avg Loss = 0.17510700787350808
Epoch 1 Iteration 601 : Avg Loss = 0.11678254612431363
Epoch 2 Iteration 1 : Avg Loss = 0.07297036524886916
Epoch 2 Iteration 201 : Avg Loss = 0.1881079525044066
Epoch 2 Iteration 401 : Avg Loss = 0.1888615387786949
Epoch 2 Iteration 601 : Avg Loss = 0.14828309971065878
Epoch 3 Iteration 1 : Avg Loss = 0.0691296822765721
Epoch 3 Iteration 201 : Avg Loss = 0.10173555418989014
Epoch 3 Iteration 401 : Avg Loss = 0.14566269985428876
Epoch 3 Iteration 601 : Avg Loss = 0.11688379121844969
Epoch 4 Iteration 1 : Avg Loss = 0.04856110558598667
Epoch 4 Iteration 201 : Avg Loss = 0.07830608669722441
Epoch 4 Iteration 401 : Avg Loss = 0.07990529918493103
Epoch 4 Iteration 601 : Avg Loss = 0.09795277032129626
Epoch 5 Iteration 1 : Avg Loss = 0.028402865971453164
Epoch 5 Iteration 201 : Avg Loss = 0.08302671915807842
Epoch 5 Iteration 401 : Avg Loss = 0.038439255179968365
Epoch 5 Iteration 601 : Avg Loss = 0.04206275166613032
Epoch 6 Iteration 1 : Avg Loss = 0.024634110359063113
Epoch 6 Iteration 201 : Avg Loss = 0.0641232264861061
Epoch 6 Iteration 401 : Avg Loss = 0.03140132256054211
Epoch 6 Iteration 601 : Avg Loss = 0.04534162271382405
Epoch 7 Iteration 1 : Avg Loss = 0.02920686127899648
Epoch 7 Iteration 201 : Avg Loss = 0.04026503338404281
Epoch 7 Iteration 401 : Avg Loss = 0.02642523893575877
Epoch 7 Iteration 601 : Avg Loss = 0.056157282918288504
Epoch 8 Iteration 1 : Avg Loss = 0.05163475866612013
Epoch 8 Iteration 201 : Avg Loss = 0.05030746820019917
Epoch 8 Iteration 401 : Avg Loss = 0.012630129906847025
Epoch 8 Iteration 601 : Avg Loss = 0.02424433438057641
Epoch 9 Iteration 1 : Avg Loss = 0.03185902907025416
Epoch 9 Iteration 201 : Avg Loss = 0.03967608046544924
Epoch 9 Iteration 401 : Avg Loss = 0.023369219452937105
Epoch 9 Iteration 601 : Avg Loss = 0.014198182768065767
Epoch 10 Iteration 1 : Avg Loss = 0.014902764776699171
Epoch 10 Iteration 201 : Avg Loss = 0.062476652899227415
Epoch 10 Iteration 401 : Avg Loss = 0.014880773452666824
Epoch 10 Iteration 601 : Avg Loss = 0.016509734007795866
```



```
Accuracy = 0.97075
Precision =  [0.98040886 0.98308668 0.9734589  0.94976077 0.97108844 0.95687332
 0.99146758 0.96209677 0.96024735 0.97743056]
Recall =  [0.9930975  0.98377997 0.96683673 0.97463175 0.96943973 0.98794063
 0.97076023 0.96521036 0.95855379 0.93677205]
F1 Score =  [0.98671239 0.9834332  0.97013652 0.96203554 0.97026338 0.97215883
 0.98100464 0.96365105 0.95939982 0.9566695 ]
```



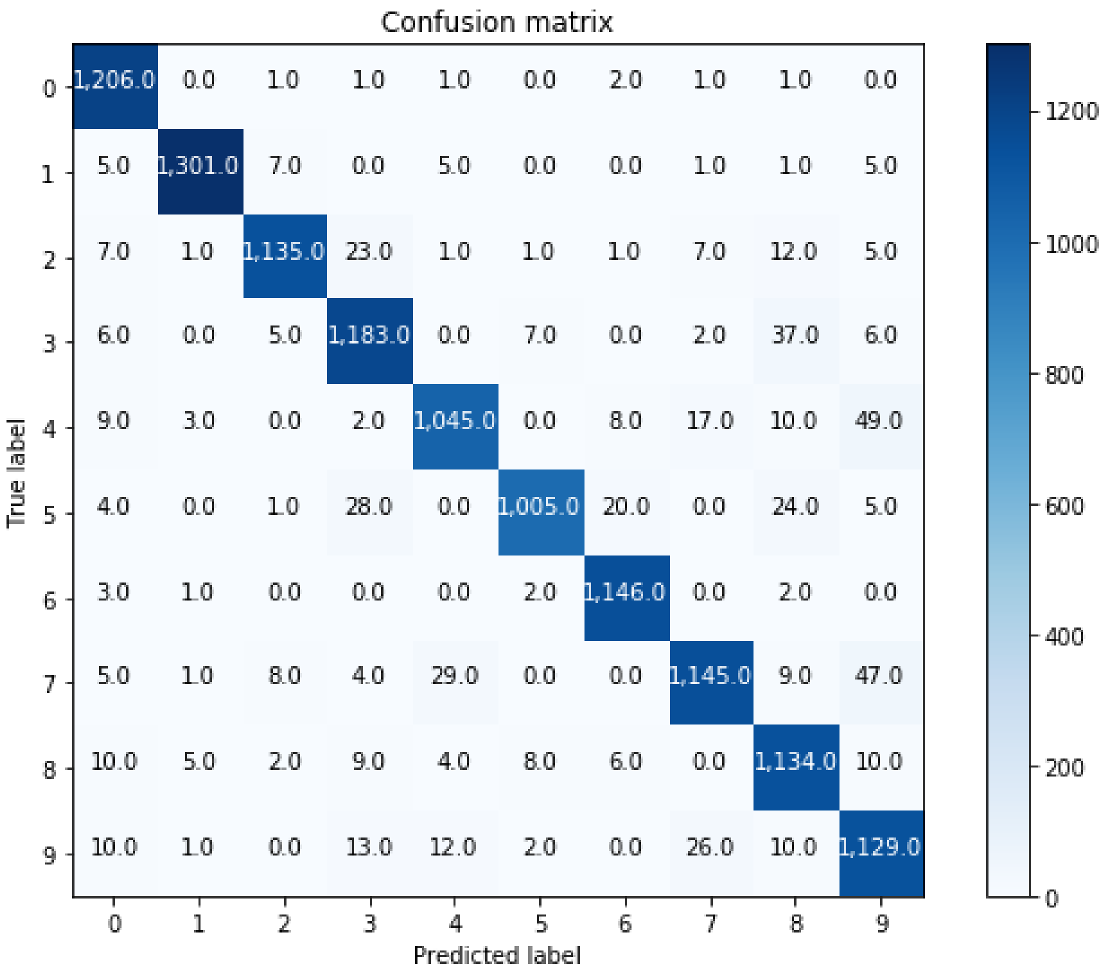
Confusion matrix

	0	1	2	3	4	5	6	7	8	9
0	1,151.0	2.0	0.0	1.0	0.0	0.0	2.0	0.0	2.0	1.0
1	2.0	1,395.0	6.0	0.0	2.0	1.0	1.0	7.0	3.0	1.0
2	3.0	2.0	1,137.0	24.0	1.0	1.0	0.0	2.0	5.0	1.0
3	0.0	0.0	8.0	1,191.0	0.0	11.0	0.0	1.0	8.0	3.0
4	2.0	6.0	2.0	0.0	1,142.0	0.0	4.0	11.0	3.0	8.0
5	0.0	0.0	1.0	5.0	0.0	1,065.0	0.0	0.0	6.0	1.0
6	8.0	1.0	3.0	1.0	9.0	9.0	1,162.0	0.0	4.0	0.0
7	0.0	6.0	11.0	6.0	6.0	0.0	0.0	1,193.0	5.0	9.0
8	2.0	4.0	0.0	14.0	3.0	17.0	3.0	2.0	1,087.0	2.0
9	6.0	3.0	0.0	12.0	13.0	9.0	0.0	24.0	9.0	1,126.0

2.0.10 KNN: Fold-1

```
In [117]: train_KNN(hog_img, labels, fold_index=0)
```

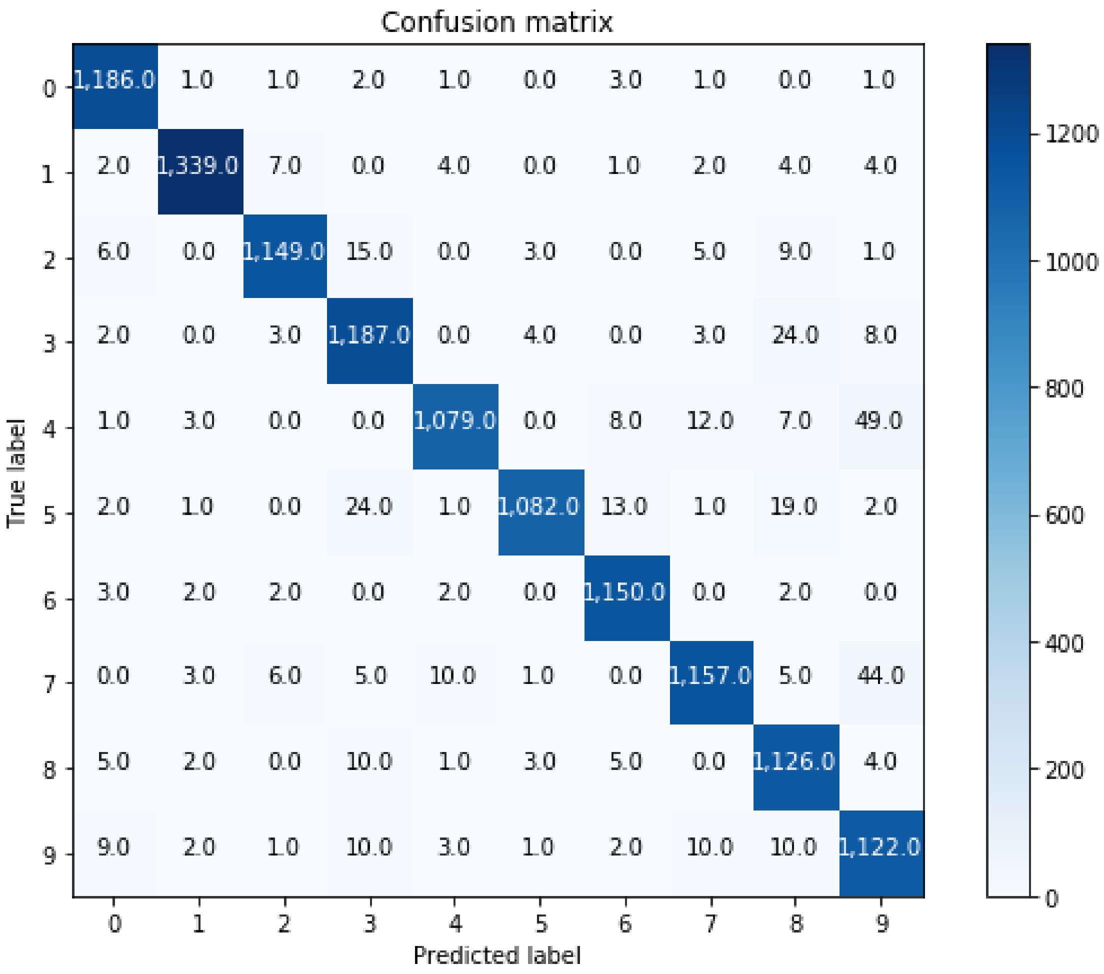
```
Accuracy = 0.9524166666666667
Precision =  [0.95335968 0.99086062 0.97929249 0.93665875 0.95259799 0.9804878
 0.96872358 0.95496247 0.91451613 0.89888535]
Recall =  [0.99422918 0.98188679 0.95138307 0.9494382  0.91426072 0.92456302
 0.99306759 0.91746795 0.95454545 0.93848712]
F1 Score =  [0.97336562 0.9863533  0.96513605 0.94300518 0.93303571 0.95170455
 0.98074454 0.9358398  0.93410214 0.91825946]
```



2.0.11 KNN: Fold-2

```
In [120]: train_KNN(hog_img, labels, fold_index=1)
```

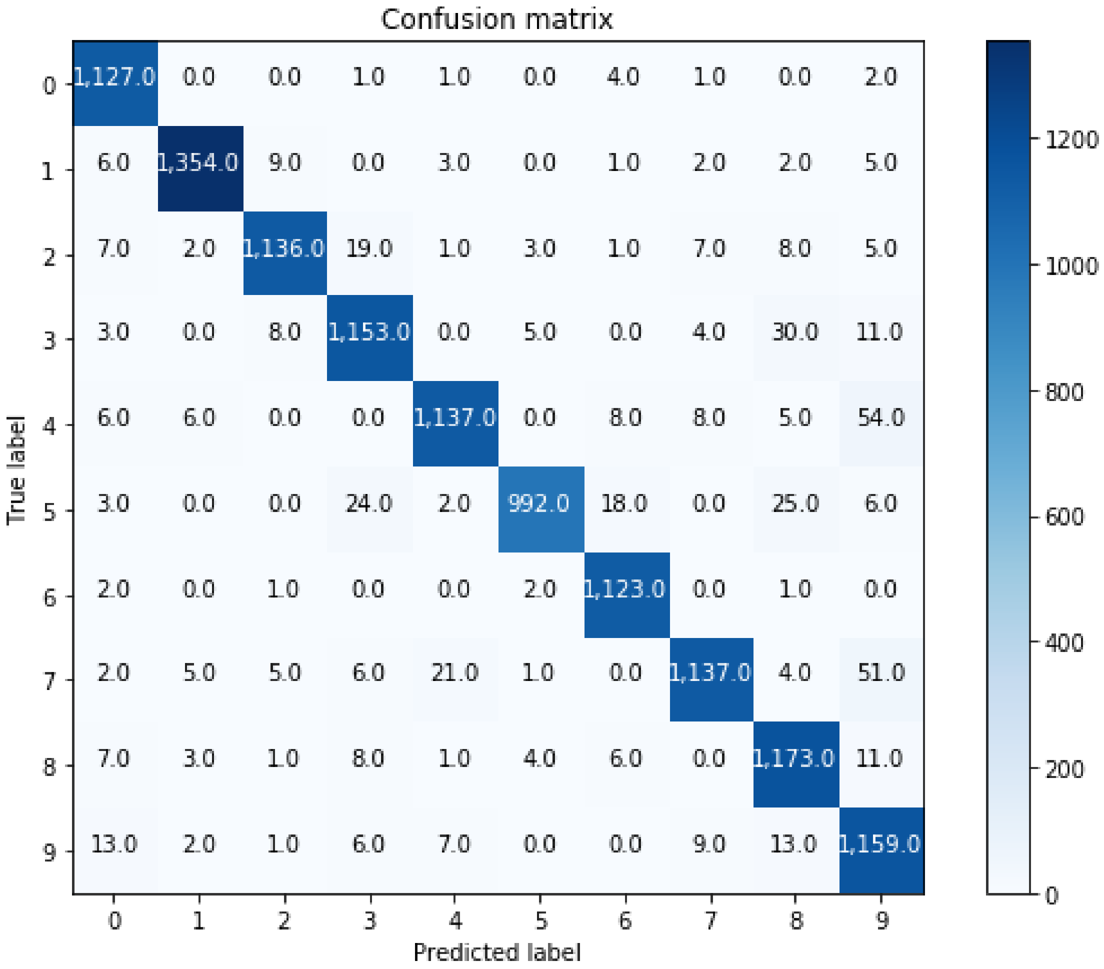
```
Accuracy = 0.96475
Precision =  [0.97532895 0.98965262 0.98289136 0.94732642 0.98001817 0.98903108
 0.97292724 0.97145256 0.93366501 0.90850202]
Recall =  [0.9916388  0.98239178 0.96717172 0.9642567  0.93097498 0.94497817
 0.99052541 0.93988627 0.97404844 0.95897436]
F1 Score =  [0.98341625 0.98600884 0.97496818 0.95571659 0.95486726 0.9665029
 0.98164746 0.95540875 0.9534293  0.93305613]
```



2.0.12 KNN: Fold-3

```
In [19]: train_KNN(hog_img, labels, fold_index=2)
```

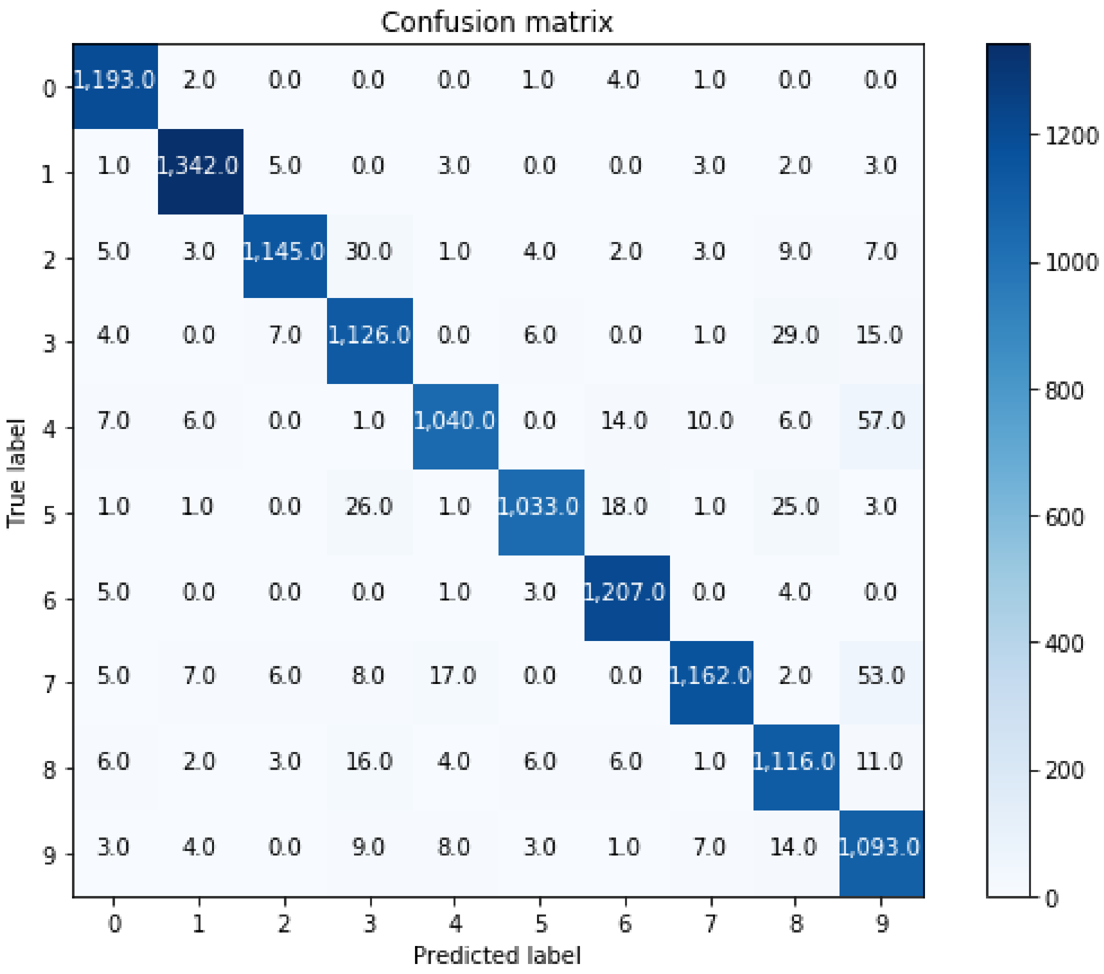
```
Accuracy = 0.9575833333333333
Precision =  [0.95833333 0.98688047 0.97846684 0.94741167 0.96930946 0.98510427
 0.9672696  0.9734589  0.93021412 0.88880368]
Recall =  [0.99207746 0.97973951 0.95542473 0.94975288 0.92892157 0.9271028
 0.99468556 0.92288961 0.96622735 0.95785124]
F1 Score =  [0.97491349 0.98329702 0.96680851 0.94858083 0.94868586 0.95522388
 0.98078603 0.9475     0.94787879 0.9220366 ]
```



2.0.13 KNN: Fold-4

```
In [21]: train_KNN(hog_img, labels, fold_index=3)
```

```
Accuracy = 0.95475
Precision =  [0.9699187  0.98171178 0.98198971 0.92598684 0.96744186 0.9782197
 0.96405751 0.97729184 0.92460646 0.88003221]
Recall =  [0.99333888 0.9874908  0.94706369 0.94781145 0.91148116 0.93146979
 0.98934426 0.92222222 0.9530316  0.95709282]
F1 Score =  [0.9814891  0.98459281 0.96421053 0.93677205 0.93862816 0.95427252
 0.97653722 0.94895876 0.93860387 0.91694631]
```

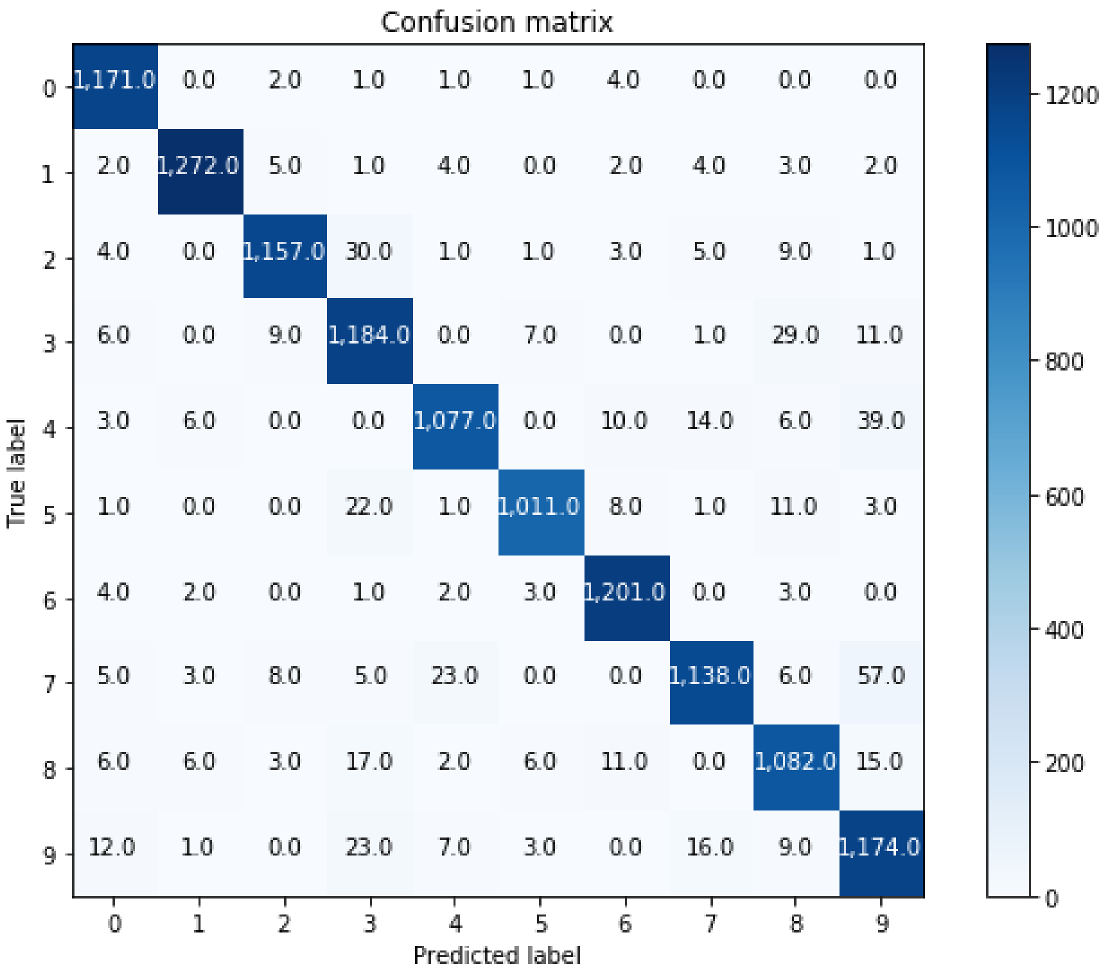


	0	1	2	3	4	5	6	7	8	9
0	1,193.0	2.0	0.0	0.0	0.0	1.0	4.0	1.0	0.0	0.0
1	1.0	1,342.0	5.0	0.0	3.0	0.0	0.0	3.0	2.0	3.0
2	5.0	3.0	1,145.0	30.0	1.0	4.0	2.0	3.0	9.0	7.0
3	4.0	0.0	7.0	1,126.0	0.0	6.0	0.0	1.0	29.0	15.0
4	7.0	6.0	0.0	1.0	1,040.0	0.0	14.0	10.0	6.0	57.0
5	1.0	1.0	0.0	26.0	1.0	1,033.0	18.0	1.0	25.0	3.0
6	5.0	0.0	0.0	0.0	1.0	3.0	1,207.0	0.0	4.0	0.0
7	5.0	7.0	6.0	8.0	17.0	0.0	0.0	1,162.0	2.0	53.0
8	6.0	2.0	3.0	16.0	4.0	6.0	6.0	1.0	1,116.0	11.0
9	3.0	4.0	0.0	9.0	8.0	3.0	1.0	7.0	14.0	1,093.0

2.0.14 KNN: Fold-5

```
In [23]: train_KNN(hog_img, labels, fold_index=4)
```

```
Accuracy = 0.9555833333333333
Precision =  [0.9645799  0.98604651 0.97719595 0.92211838 0.96332737 0.97965116
 0.9693301  0.96522477 0.9343696  0.90168971]
Recall =  [0.99237288 0.98223938 0.95540875 0.94947875 0.93246753 0.95557656
 0.98766447 0.91405622 0.94250871 0.94297189]
F1 Score =  [0.97827903 0.98413926 0.96617954 0.93559858 0.94764628 0.96746411
 0.97841141 0.93894389 0.93842151 0.92186887]
```

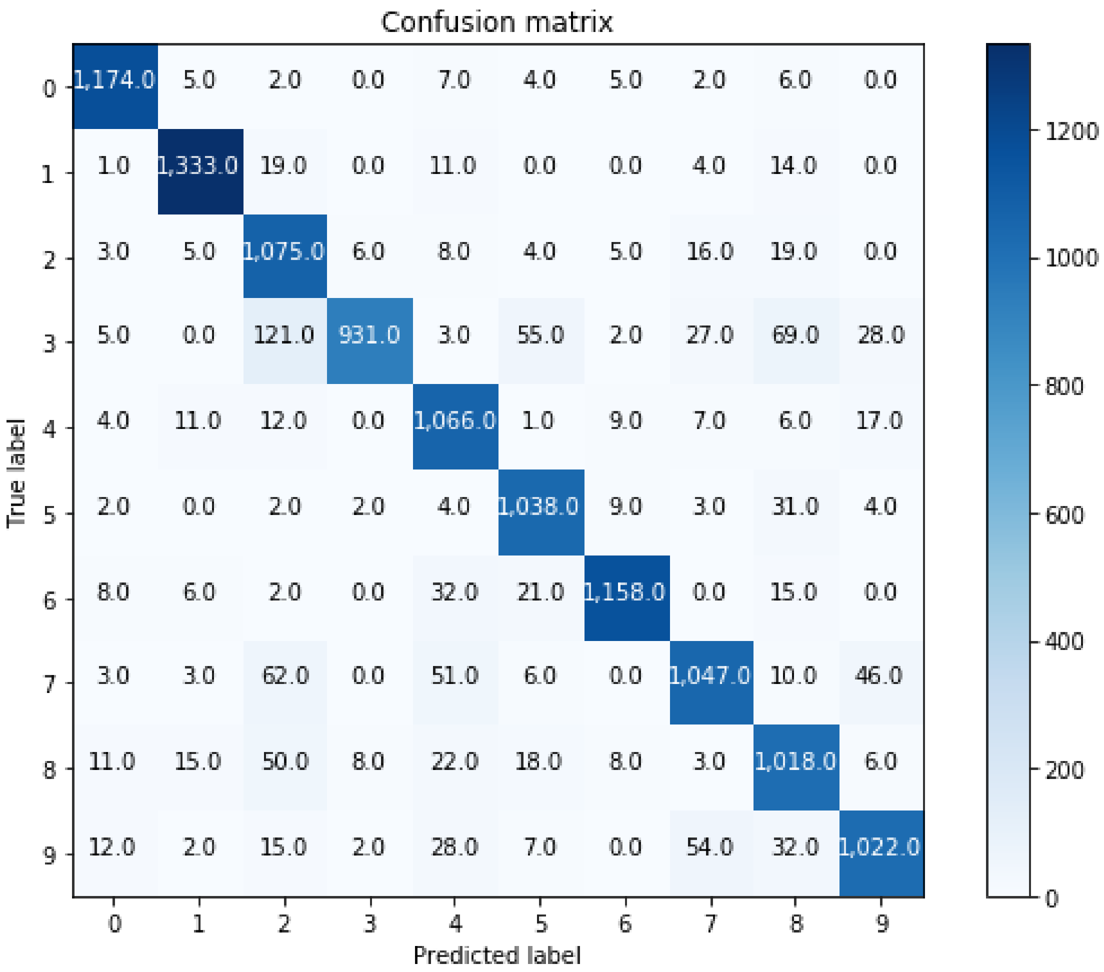


True label \ Predicted label	0	1	2	3	4	5	6	7	8	9
0	1,171.0	0.0	2.0	1.0	1.0	1.0	4.0	0.0	0.0	0.0
1	2.0	1,272.0	5.0	1.0	4.0	0.0	2.0	4.0	3.0	2.0
2	4.0	0.0	1,157.0	30.0	1.0	1.0	3.0	5.0	9.0	1.0
3	6.0	0.0	9.0	1,184.0	0.0	7.0	0.0	1.0	29.0	11.0
4	3.0	6.0	0.0	0.0	1,077.0	0.0	10.0	14.0	6.0	39.0
5	1.0	0.0	0.0	22.0	1.0	1,011.0	8.0	1.0	11.0	3.0
6	4.0	2.0	0.0	1.0	2.0	3.0	1,201.0	0.0	3.0	0.0
7	5.0	3.0	8.0	5.0	23.0	0.0	0.0	1,138.0	6.0	57.0
8	6.0	6.0	3.0	17.0	2.0	6.0	11.0	0.0	1,082.0	15.0
9	12.0	1.0	0.0	23.0	7.0	3.0	0.0	16.0	9.0	1,174.0

2.0.15 SVM: Fold-1

```
In [14]: train_SVM(hog_img, labels, 0)
```

```
Accuracy = 0.9051666666666667
Precision =  [0.95993459 0.96594203 0.79044118 0.98103267 0.86525974 0.89948007
 0.96822742 0.90025795 0.83442623 0.91006233]
Recall =  [0.97427386 0.96454414 0.942156   0.75020145 0.94086496 0.94794521
 0.93236715 0.85260586 0.8783434  0.87052811]
F1 Score =  [0.96705107 0.96524258 0.85965614 0.85022831 0.90147992 0.92307692
 0.94995898 0.87578419 0.85582177 0.88985633]
```

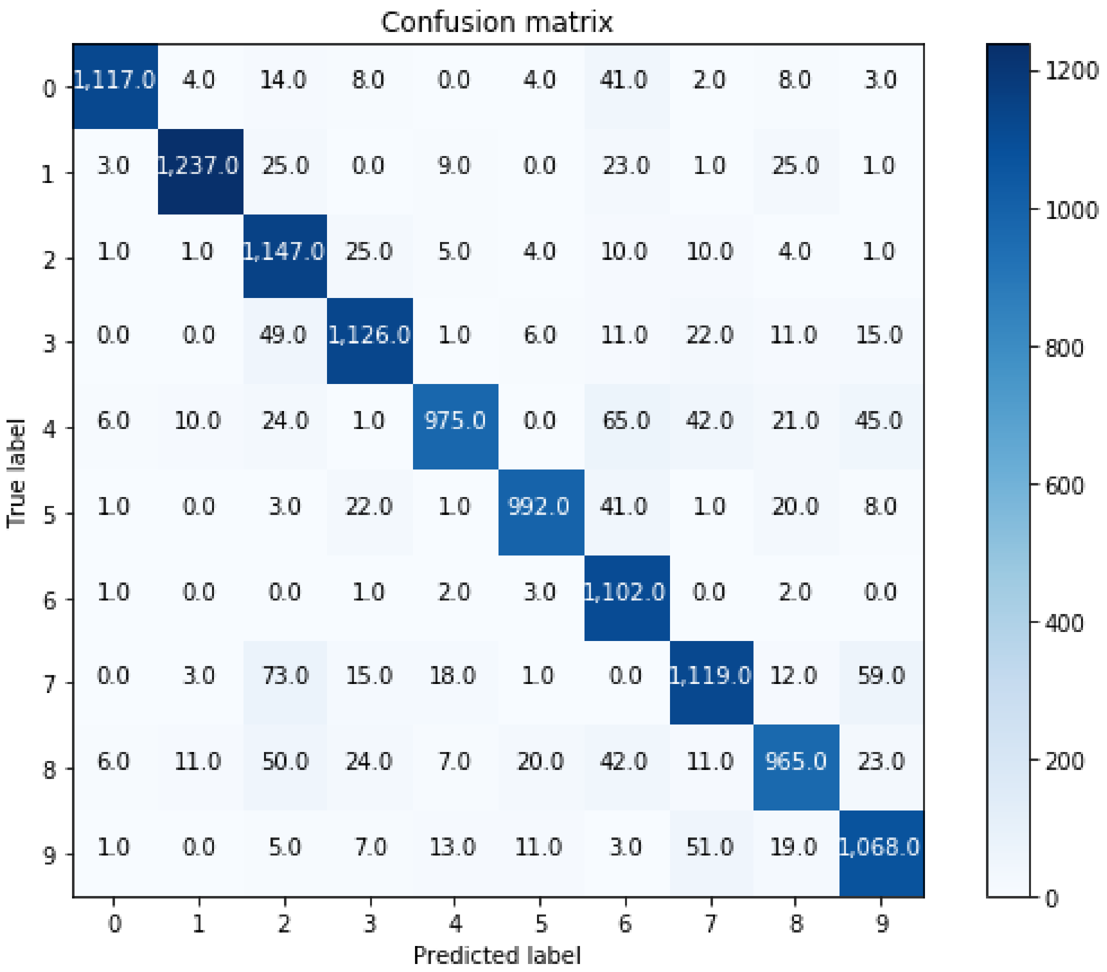


	0	1	2	3	4	5	6	7	8	9
0	1,174.0	5.0	2.0	0.0	7.0	4.0	5.0	2.0	6.0	0.0
1	1.0	1,333.0	19.0	0.0	11.0	0.0	0.0	4.0	14.0	0.0
2	3.0	5.0	1,075.0	6.0	8.0	4.0	5.0	16.0	19.0	0.0
3	5.0	0.0	121.0	931.0	3.0	55.0	2.0	27.0	69.0	28.0
4	4.0	11.0	12.0	0.0	1,066.0	1.0	9.0	7.0	6.0	17.0
5	2.0	0.0	2.0	2.0	4.0	1,038.0	9.0	3.0	31.0	4.0
6	8.0	6.0	2.0	0.0	32.0	21.0	1,158.0	0.0	15.0	0.0
7	3.0	3.0	62.0	0.0	51.0	6.0	0.0	1,047.0	10.0	46.0
8	11.0	15.0	50.0	8.0	22.0	18.0	8.0	3.0	1,018.0	6.0
9	12.0	2.0	15.0	2.0	28.0	7.0	0.0	54.0	32.0	1,022.0

2.0.16 SVM: Fold-2

```
In [15]: train_SVM(hog_img, labels, 1)
```

```
Accuracy = 0.904
Precision =  [0.98327465 0.97709321 0.82517986 0.91619203 0.9456838  0.95292988
 0.82361734 0.88880064 0.88776449 0.87326247]
Recall =  [0.93005828 0.93429003 0.94950331 0.9073328  0.82001682 0.91092746
 0.99189919 0.86076923 0.83261432 0.90662139]
F1 Score =  [0.9559264  0.95521236 0.88298691 0.91174089 0.87837838 0.9314554
 0.89995917 0.87456038 0.85930543 0.88962932]
```

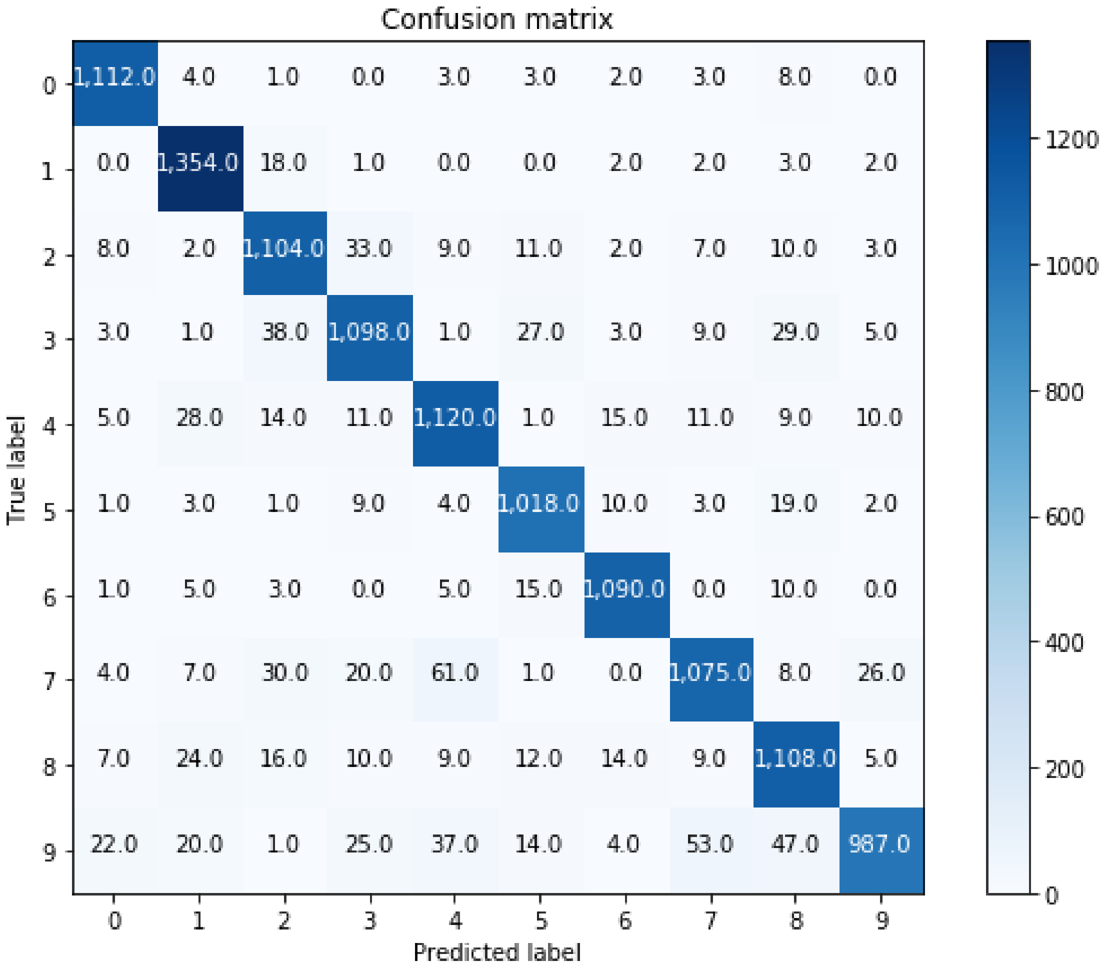


	0	1	2	3	4	5	6	7	8	9
0	1,117.0	4.0	14.0	8.0	0.0	4.0	41.0	2.0	8.0	3.0
1	3.0	1,237.0	25.0	0.0	9.0	0.0	23.0	1.0	25.0	1.0
2	1.0	1.0	1,147.0	25.0	5.0	4.0	10.0	10.0	4.0	1.0
3	0.0	0.0	49.0	1,126.0	1.0	6.0	11.0	22.0	11.0	15.0
4	6.0	10.0	24.0	1.0	975.0	0.0	65.0	42.0	21.0	45.0
5	1.0	0.0	3.0	22.0	1.0	992.0	41.0	1.0	20.0	8.0
6	1.0	0.0	0.0	1.0	2.0	3.0	1,102.0	0.0	2.0	0.0
7	0.0	3.0	73.0	15.0	18.0	1.0	0.0	1,119.0	12.0	59.0
8	6.0	11.0	50.0	24.0	7.0	20.0	42.0	11.0	965.0	23.0
9	1.0	0.0	5.0	7.0	13.0	11.0	3.0	51.0	19.0	1,068.0

2.0.17 SVM: Fold-3

```
In [20]: train_SVM(hog_img, labels, 2)
```

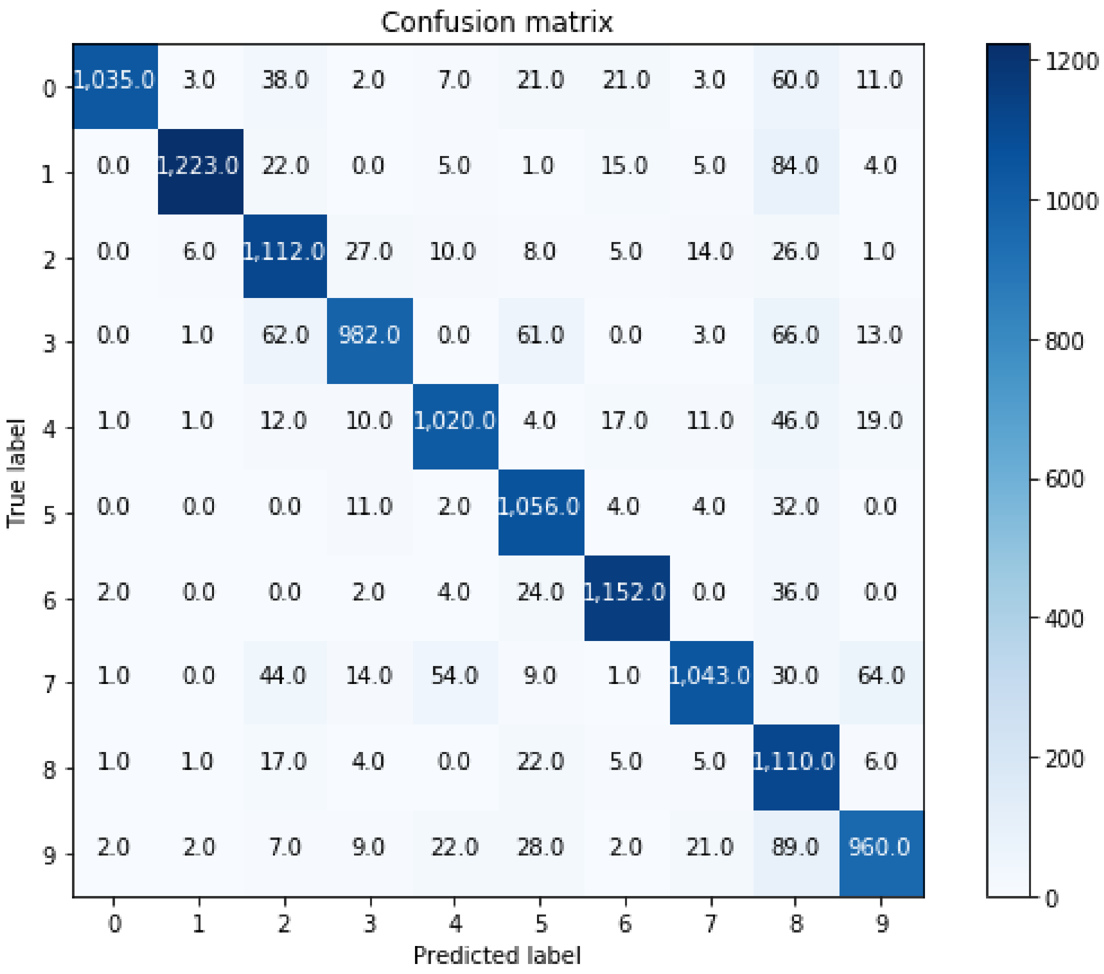
```
Accuracy = 0.9221666666666667
Precision =  [0.95614789 0.93508287 0.9004894  0.90969345 0.89671737 0.92377495
 0.95446585 0.91723549 0.88569145 0.94903846]
Recall =  [0.97887324 0.97973951 0.92851135 0.90444811 0.91503268 0.95140187
 0.96545616 0.87256494 0.91268534 0.81570248]
F1 Score =  [0.96737712 0.95689046 0.91428571 0.9070632  0.90578245 0.9373849
 0.95992955 0.89434276 0.8989858  0.87733333]
```



2.0.18 SVM: Fold-4

```
In [22]: train_SVM(hog_img, labels, 3)
```

```
Accuracy = 0.8910833333333333
Precision =  [0.99328215 0.9886823  0.84627093 0.92554194 0.90747331 0.85575365
 0.94271686 0.94048693 0.70297657 0.89053803]
Recall =  [0.86178185 0.89992642 0.9197684  0.82659933 0.89395267 0.9522092
 0.9442623  0.82777778 0.94790777 0.84063047]
F1 Score =  [0.92287115 0.9422188  0.88149029 0.87327701 0.90066225 0.90140845
 0.94348894 0.88054031 0.80727273 0.86486486]
```

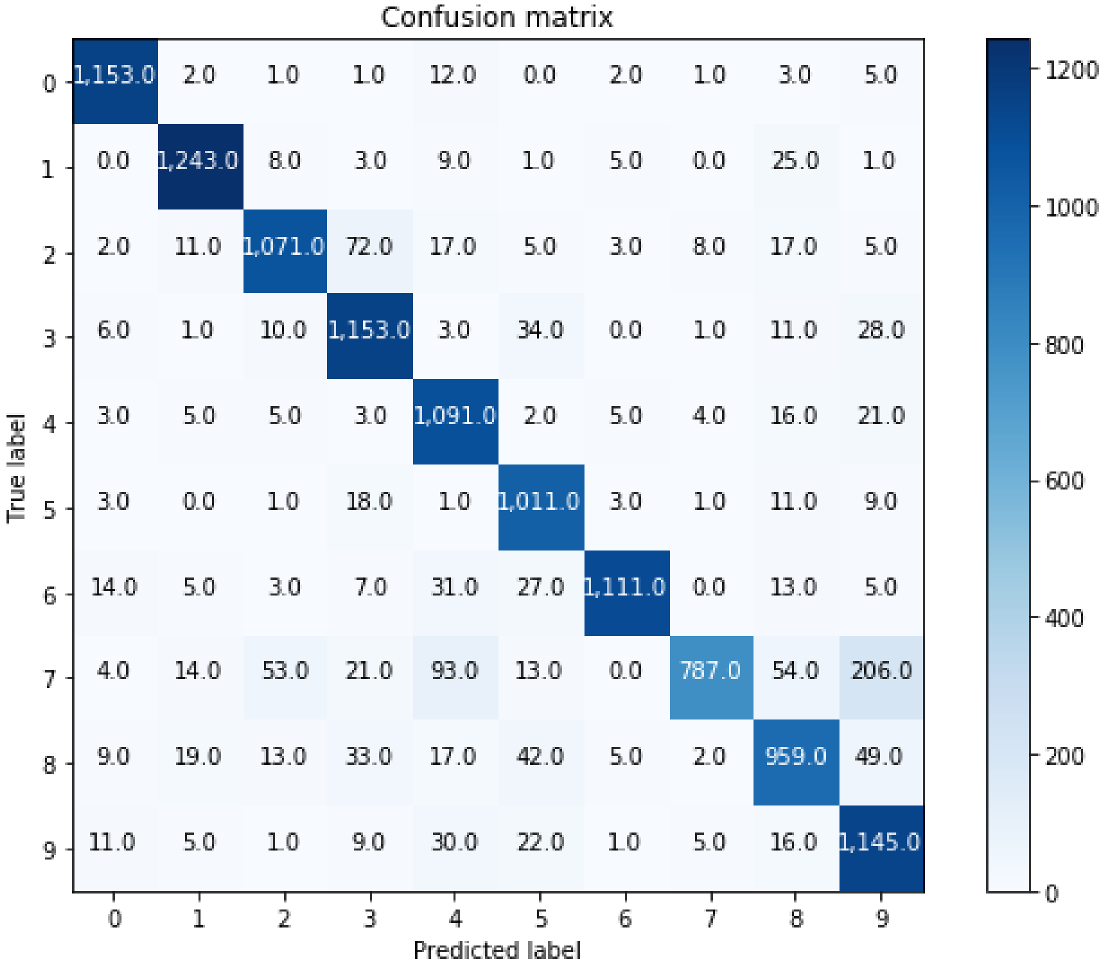


	0	1	2	3	4	5	6	7	8	9
0	1,035.0	3.0	38.0	2.0	7.0	21.0	21.0	3.0	60.0	11.0
1	0.0	1,223.0	22.0	0.0	5.0	1.0	15.0	5.0	84.0	4.0
2	0.0	6.0	1,112.0	27.0	10.0	8.0	5.0	14.0	26.0	1.0
3	0.0	1.0	62.0	982.0	0.0	61.0	0.0	3.0	66.0	13.0
4	1.0	1.0	12.0	10.0	1,020.0	4.0	17.0	11.0	46.0	19.0
5	0.0	0.0	0.0	11.0	2.0	1,056.0	4.0	4.0	32.0	0.0
6	2.0	0.0	0.0	2.0	4.0	24.0	1,152.0	0.0	36.0	0.0
7	1.0	0.0	44.0	14.0	54.0	9.0	1.0	1,043.0	30.0	64.0
8	1.0	1.0	17.0	4.0	0.0	22.0	5.0	5.0	1,110.0	6.0
9	2.0	2.0	7.0	9.0	22.0	28.0	2.0	21.0	89.0	960.0

2.0.19 SVM: Fold-5

```
In [24]: train_SVM(hog_img, labels, 4)
```

```
Accuracy = 0.8936666666666667
Precision =  [0.95684647 0.95249042 0.91852487 0.87348485 0.83665644 0.87381158
 0.97885463 0.97280593 0.85244444 0.77679783]
Recall =  [0.97711864 0.95984556 0.88439306 0.92461909 0.94458874 0.95557656
 0.91365132 0.63212851 0.83536585 0.91967871]
F1 Score =  [0.96687631 0.95615385 0.90113589 0.89832489 0.88735258 0.91286682
 0.94512973 0.76630964 0.84381874 0.8422214 ]
```


Confusion matrix
True label
Predicted label
0 1 2 3 4 5 6 7 8 9
0 1,153.0 2.0 1.0 1.0 12.0 0.0 2.0 1.0 3.0 5.0
1 0.0 1,243.0 8.0 3.0 9.0 1.0 5.0 0.0 25.0 1.0
2 2.0 11.0 1,071.0 72.0 17.0 5.0 3.0 8.0 17.0 5.0
3 6.0 1.0 10.0 1,153.0 3.0 34.0 0.0 1.0 11.0 28.0
4 3.0 5.0 5.0 3.0 1,091.0 2.0 5.0 4.0 16.0 21.0
5 3.0 0.0 1.0 18.0 1.0 1,011.0 3.0 1.0 11.0 9.0
6 14.0 5.0 3.0 7.0 31.0 27.0 1,111.0 0.0 13.0 5.0
7 4.0 14.0 53.0 21.0 93.0 13.0 0.0 787.0 54.0 206.0
8 9.0 19.0 13.0 33.0 17.0 42.0 5.0 2.0 959.0 49.0
9 11.0 5.0 1.0 9.0 30.0 22.0 1.0 5.0 16.0 1,145.0
1200
1000
800
600
400
200
0