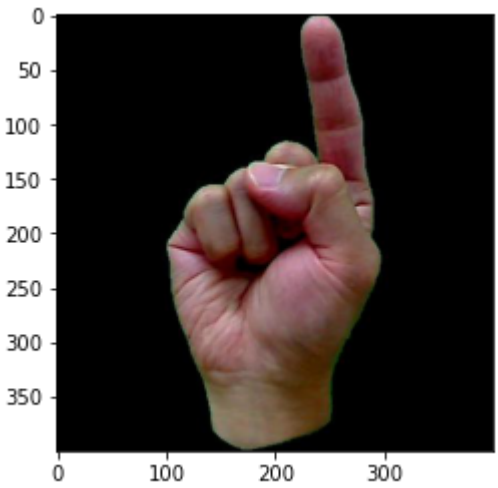
In [ ]:
```
from matplotlib import image
from matplotlib import pyplot
import os
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn import model_selection
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDi
```

DataPreProcessing:

In [ ]:
```
path = './asl_dataset/'
data,rawLabel = [],[]
labelTypes=[]
for root, dirs, files in os.walk(path):
    key = os.path.basename(root)
    if key=="": continue
    labelTypes.append(key)
    print(key, end=' ')
    for file in files:
        full_file_path = os.path.join(root,file)
        img = image.imread(full_file_path)
        data.append(img)
        rawLabel.append(key)

data = np.array(data)/255.0
rawLabel = np.array(rawLabel)
labelNum=len(labelTypes)
print(img.dtype,img.shape)
pyplot.imshow(img)
print(data.shape,rawLabel.shape,labelNum)
print(labelTypes)
```

```
0 1 2 3 4 5 6 7 8 9 a b c d e f g h i j k l m n o p q r s t u v w x y z uint8 (400, 400, 3)
(2515, 400, 400, 3) (2515,) 36
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
```



In [ ]:
```
label=np.array([[rawLabel[i]==labelTypes[j] for j in range(labelNum)]for i in range(l
print(label.shape)
print(label[0])
print(label[-1])
```

```
(2515, 36)
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

split data into train and test for cross validation

In [ ]:
```
X_train,X_use,y_train,y_use  = model_selection.train_test_split(data, label, train_size
X_test, X_val, y_test, y_val = model_selection.train_test_split(X_use, y_use, train_siz
print(X_train.shape, X_test.shape,X_val.shape, y_train.shape, y_test.shape,y_val.shape)
```

```
(2012, 400, 400, 3) (251, 400, 400, 3) (252, 400, 400, 3) (2012, 36) (251, 36) (252, 3
6)
```

In [ ]:
```
X_trainTensor=tf.constant(X_train)
X_testTensor=tf.constant(X_test)
X_valTensor=tf.constant(X_val)
Y_trainTensor=tf.constant(y_train)
Y_testTensor=tf.constant(y_test)
Y_valTensor=tf.constant(y_val)
print(X_trainTensor.shape,Y_trainTensor.shape)
```

```
(2012, 400, 400, 3) (2012, 36)
```

Data Analysis:

By observing the following histgram,

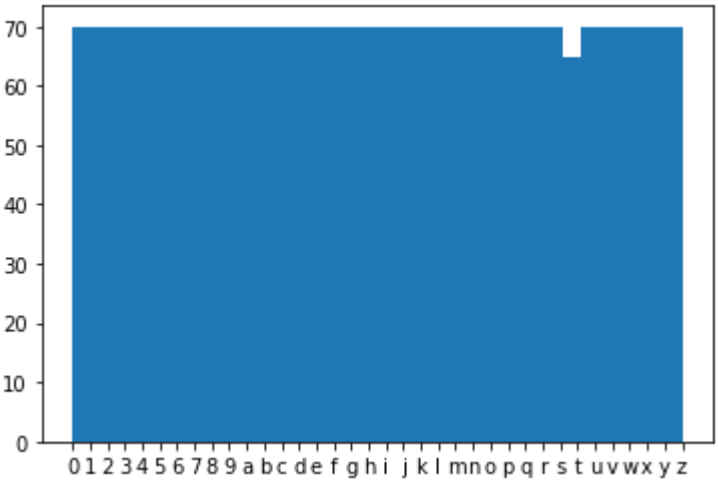
we find that each type of images are almost the same amount(70).

By looking through each data type folder,

we find that the qulities of each type of data is presistent.

In [ ]:
```
pyplot.hist(rawLabel,bins=labelNum)
```

Out[ ]:
```
(array([70., 70., 70., 70., 70., 70., 70., 70., 70., 70., 70., 70.,
        70., 70., 70., 70., 70., 70., 70., 70., 70., 70., 70., 70.,
        70., 70., 70., 65., 70., 70., 70., 70., 70., 70.]),
 array([ 0.        ,  0.97222222,  1.94444444,  2.91666667,  3.88888889,
         4.86111111,  5.83333333,  6.80555556,  7.77777778,  8.75      ,
         9.72222222, 10.69444444, 11.66666667, 12.63888889, 13.61111111,
        14.58333333, 15.55555556, 16.52777778, 17.5       , 18.47222222,
        19.44444444, 20.41666667, 21.38888889, 22.36111111, 23.33333333,
        24.30555556, 25.27777778, 26.25      , 27.22222222, 28.19444444,
        29.16666667, 30.13888889, 31.11111111, 32.08333333, 33.05555556,
        34.02777778, 35.        ]),
 <BarContainer object of 36 artists>)
```



One-Channel-AutoEncoders:

Here we'll build 3 different One-Channel-AutoEncoders which encode original 400*400*3 graphs into:

1) 20x20x1

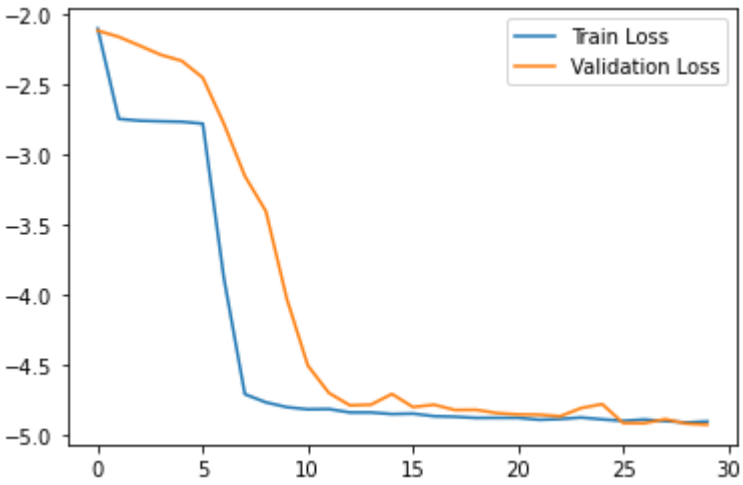
2) 10x10x1

3) 5x5x1

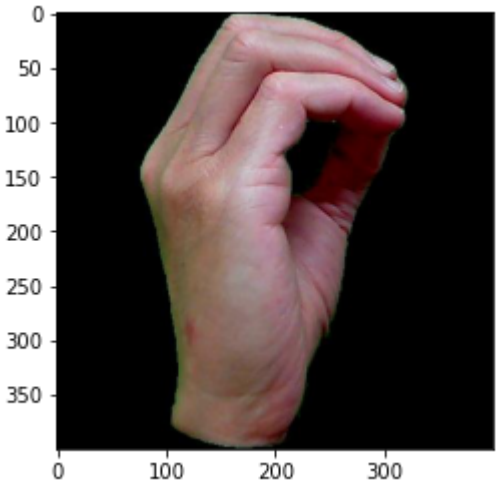
In [ ]:
```
class AutoEncoder1(keras.Model):
  def __init__(self):
    super(AutoEncoder1, self).__init__()
    self.encoder = tf.keras.models.Sequential([
      layers.MaxPool2D(pool_size=(5, 5),strides=(5, 5), padding='same'),
      layers.Conv2D(2, (3,3), activation = 'relu',strides=1,padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      layers.Conv2D(1, (3,3), activation = 'sigmoid',strides=1,padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      ])
    self.decoder = tf.keras.models.Sequential([
      layers.Conv2DTranspose(1,(5,5),strides=5,activation='relu', padding='same'),
      layers.Conv2DTranspose(4,(5,5),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(3,3),strides=2,activation='relu', padding='same'),
      layers.BatchNormalization(),
      layers.Conv2D(8, kernel_size=(3, 3), activation='relu', padding='same'),
      layers.Conv2D(3, kernel_size=(3, 3), activation='sigmoid', padding='same')
      ])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder1=AutoEncoder1()
autoencoder1.compile(optimizer='adam',loss=keras.losses.MeanSquaredError())
history1 = autoencoder1.fit(X_trainTensor, X_trainTensor,
                        validation_data=(X_valTensor,X_valTensor),
                        epochs=30,shuffle=True)
```

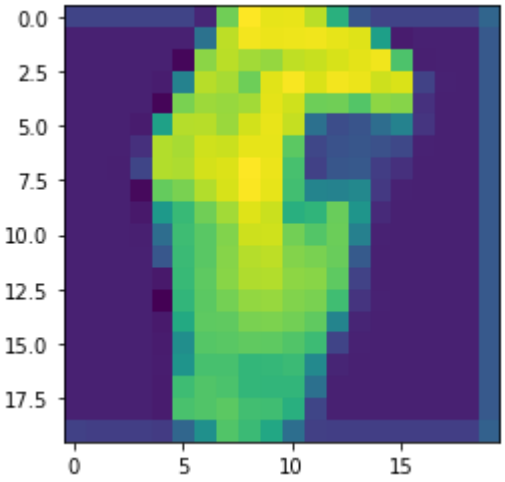
In [ ]:
```
autoencoder1.summary()
pyplot.plot(np.log(history1.history['loss']), label='Train Loss')
pyplot.plot(np.log(history1.history['val_loss']), label='Validation Loss')
pyplot.legend()
pyplot.show()
```

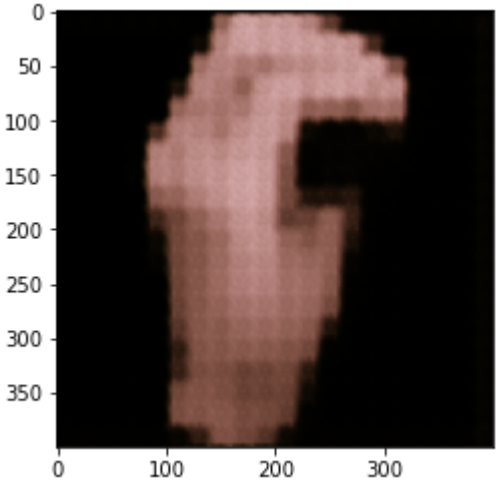
```
Model: "auto_encoder1_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_42 (Sequential)  (None, 20, 20, 1)         75

 sequential_43 (Sequential)  (None, 400, 400, 3)       1261

=================================================================
Total params: 1,336
Trainable params: 1,320
Non-trainable params: 16
_________________________________________________________________
```



In [ ]:
```
pyplot.imshow(X_train[0]);
pyplot.show()
middleImage=autoencoder1.encoder(X_train[[0]])
pyplot.imshow(middleImage[0].numpy());
pyplot.show()
newImage=autoencoder1.decoder(middleImage)[0].numpy()
pyplot.imshow(newImage);
```





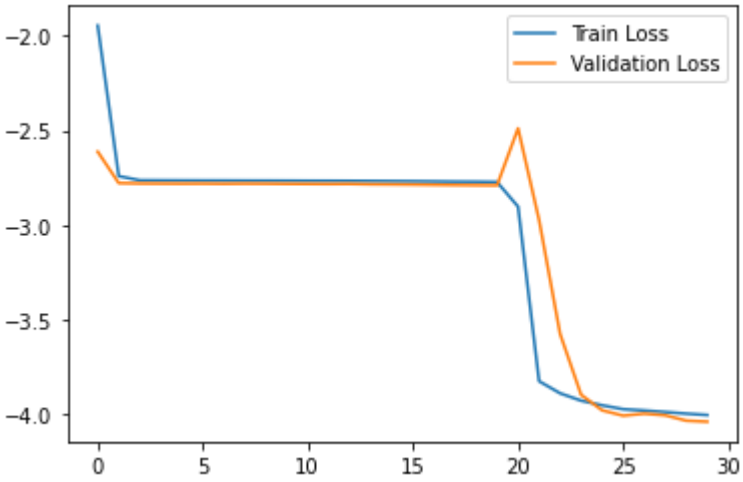


In [ ]:
```
class AutoEncoder2(keras.Model):
  def __init__(self):
    super(AutoEncoder2, self).__init__()
    self.encoder = tf.keras.models.Sequential([
      layers.MaxPool2D(pool_size=(10, 10),strides=(10, 10), padding='same'),
      layers.Conv2D(2, (3,3),strides=1,activation='relu', padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      layers.Conv2D(1, (3,3),strides=1,activation="sigmoid",padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      ])
    self.decoder = tf.keras.models.Sequential([
      layers.Conv2DTranspose(1,(5,5),strides=5,activation='relu', padding='same'),
      layers.Conv2DTranspose(4,(5,5),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(5,5),strides=2,activation='relu', padding='same'),
      layers.BatchNormalization(),
      layers.Conv2DTranspose(8,(3,3),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(3,3),strides=1,activation='relu', padding='same'),
      layers.Conv2D(3, kernel_size=(3, 3), activation='sigmoid', padding='same')
      ])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder2=AutoEncoder2()
autoencoder2.compile(optimizer='adam',loss=keras.losses.MeanSquaredError())
history2 = autoencoder2.fit(X_trainTensor, X_trainTensor,
```

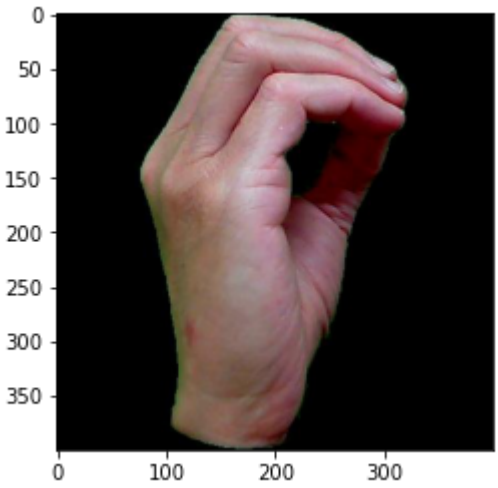
```
                        validation_data=(X_valTensor,X_valTensor),
                        epochs=30,shuffle=True)
```

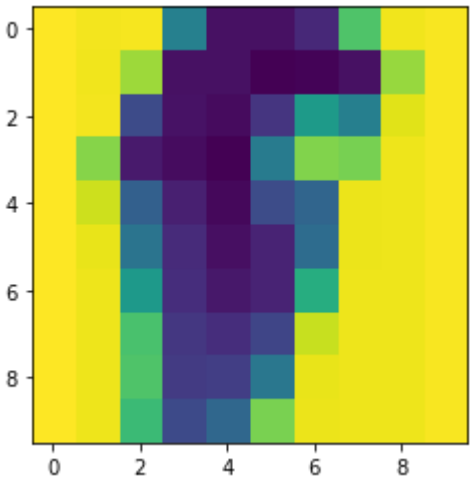
In [ ]:
```
autoencoder2.summary()
pyplot.plot(np.log(history2.history['loss']), label='Train Loss')
pyplot.plot(np.log(history2.history['val_loss']), label='Validation Loss')
pyplot.legend()
pyplot.show()
```

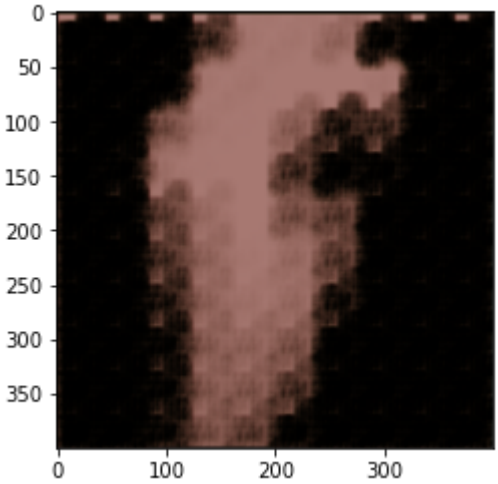
```
Model: "auto_encoder2_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_44 (Sequential)  (None, 10, 10, 1)         75

 sequential_45 (Sequential)  (None, 400, 400, 3)       2357

=================================================================
Total params: 2,432
Trainable params: 2,416
Non-trainable params: 16
_________________________________________________________________
```



In [ ]:
```
pyplot.imshow(X_train[0]);
pyplot.show()
middleImage=autoencoder2.encoder(X_train[[0]])
pyplot.imshow(middleImage[0].numpy());
pyplot.show()
newImage=autoencoder2.decoder(middleImage)[0].numpy()
pyplot.imshow(newImage);
```





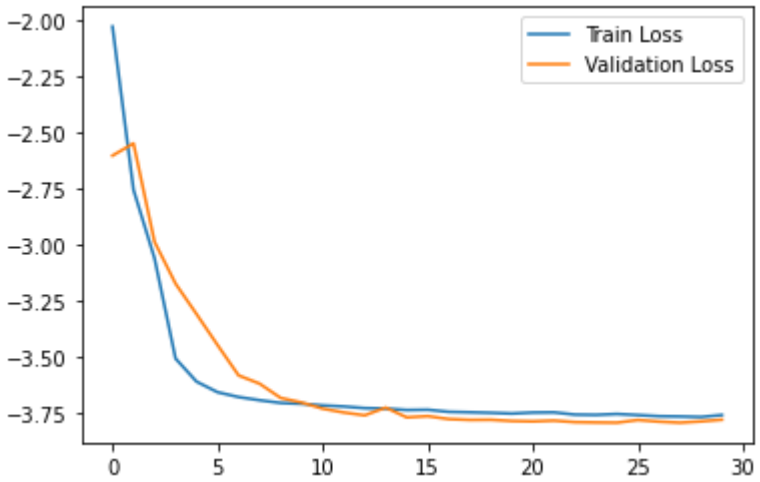


In [ ]:
```
class AutoEncoder3(keras.Model):
  def __init__(self):
    super(AutoEncoder3, self).__init__()
    self.encoder = tf.keras.models.Sequential([
      layers.MaxPool2D(pool_size=(10, 10),strides=(10, 10), padding='same'),
      layers.Conv2D(4, (3,3),strides=1,activation='relu', padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      layers.Conv2D(2, (3,3),strides=1,activation="relu",padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      layers.Conv2D(1, (3,3),strides=1,activation="sigmoid",padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      ])
```

```
    self.decoder = tf.keras.models.Sequential([
      layers.Conv2DTranspose(1,(5,5),strides=5,activation='relu', padding='same'),
      layers.Conv2DTranspose(4,(5,5),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(4,(5,5),strides=2,activation='relu', padding='same'),
      layers.BatchNormalization(),
      layers.Conv2DTranspose(8,(3,3),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(3,3),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(3,3),strides=1,activation='relu', padding='same'),
      layers.Conv2D(3, kernel_size=(3, 3), activation='sigmoid', padding='same')])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

autoencoder3=AutoEncoder3()
autoencoder3.compile(optimizer='adam',loss=keras.losses.MeanSquaredError())
history3 = autoencoder3.fit(X_trainTensor, X_trainTensor,
                            validation_data=(X_valTensor,X_valTensor),
                            epochs=30,shuffle=True)
```

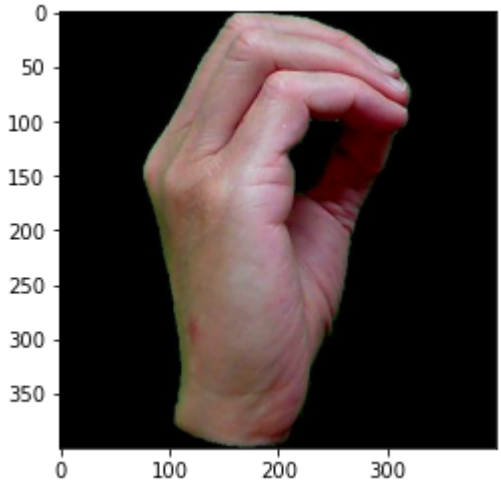
In [ ]:
```
autoencoder3.summary()
pyplot.plot(np.log(history3.history['loss']), label='Train Loss')
pyplot.plot(np.log(history3.history['val_loss']), label='Validation Loss')
pyplot.legend()
pyplot.show()
```

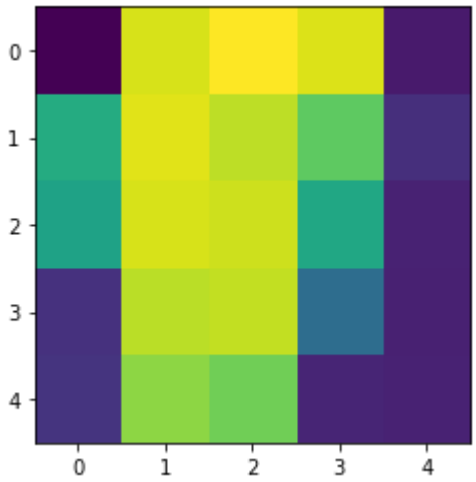
```
Model: "auto_encoder3_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_46 (Sequential)  (None, 5, 5, 1)           205

 sequential_47 (Sequential)  (None, 400, 400, 3)       2233

=================================================================
Total params: 2,438
Trainable params: 2,430
Non-trainable params: 8
_________________________________________________________________
```

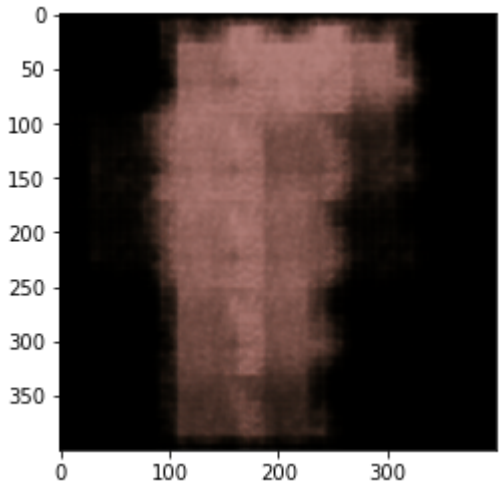


In [ ]:
```
pyplot.imshow(X_train[0]);
pyplot.show()
middleImage=autoencoder3.encoder(X_train[[0]])
```

```
pyplot.imshow(middleImage[0].numpy());
pyplot.show()
newImage=autoencoder3.decoder(middleImage)[0].numpy()
pyplot.imshow(newImage);
```







summary:

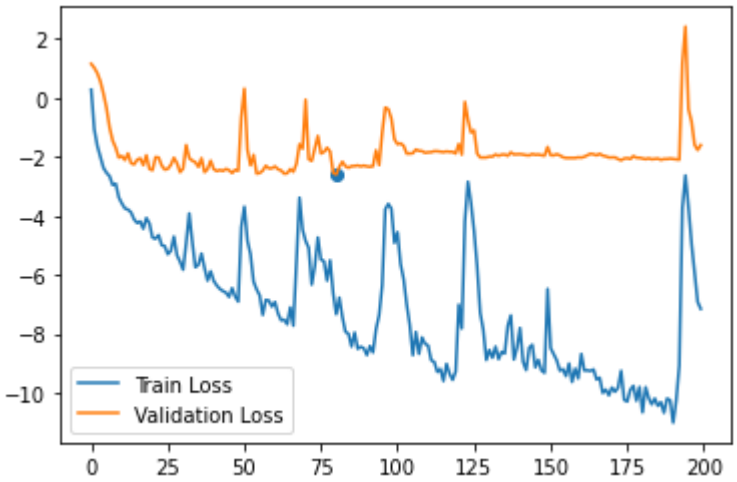
Classifiers: now we'll build 3 different classifiers based on different AutoEncoders with slightly different structure.

In [ ]:
```
class Clasiffier1(keras.Model):
  def __init__(self):
    super(Clasiffier1, self).__init__()
    self.linear = tf.keras.models.Sequential([
        layers.Conv2D(4, (5,5), activation = 'relu',strides=1,padding='same'),
        layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
        layers.Conv2D(16, (3,3), activation = 'relu',strides=1,padding='same'),
        layers.Flatten(),
        layers.BatchNormalization(),
        layers.Dense(36,activation='softmax')
      ])

  def call(self, x):
    encoded = autoencoder1.encoder(x)
    return self.linear(encoded)

clasiffier1=Clasiffier1()

class Clasiffier2(keras.Model):
    def __init__(self):
      super(Clasiffier2, self).__init__()
      self.linear = tf.keras.models.Sequential([
        layers.Conv2D(4, (5,5), activation = 'relu',strides=1,padding='same'),
        layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
        layers.Conv2D(16, (3,3), activation = 'relu',strides=1,padding='same'),
        layers.Flatten(),
        layers.BatchNormalization(),
        layers.Dense(36,activation='softmax')
      ])

    def call(self, x):
      encoded = autoencoder2.encoder(x)
      return self.linear(encoded)

clasiffier2=Clasiffier2()

class Clasiffier3(keras.Model):
    def __init__(self):
      super(Clasiffier3, self).__init__()
      self.linear = tf.keras.models.Sequential([
        layers.Conv2D(4, (3,3), activation = 'relu',strides=1,padding='same'),
        layers.Conv2D(16, (3,3), activation = 'relu',strides=1,padding='same'),
        layers.Flatten(),
        layers.BatchNormalization(),
        layers.Dense(36,activation='softmax')
      ])

    def call(self, x):
      encoded = autoencoder3.encoder(x)
      return self.linear(encoded)

clasiffier3=Clasiffier3()
```

In [ ]:
```
clasiffier1.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy
history4 = clasiffier1.fit(X_trainTensor, Y_trainTensor, epochs=200,
                           validation_data=(X_valTensor,Y_valTensor),shuffle=True)
```

In [ ]:
```
clasiffier1.summary()
pyplot.plot(np.log(history4.history['loss']), label='Train Loss')
pyplot.plot(np.log(history4.history['val_loss']), label='Validation Loss')
lowestIndex=np.argmin(history4.history['val_loss'])
pyplot.scatter(lowestIndex,np.log(history4.history['val_loss'][lowestIndex]))
pyplot.legend()
pyplot.show()
```

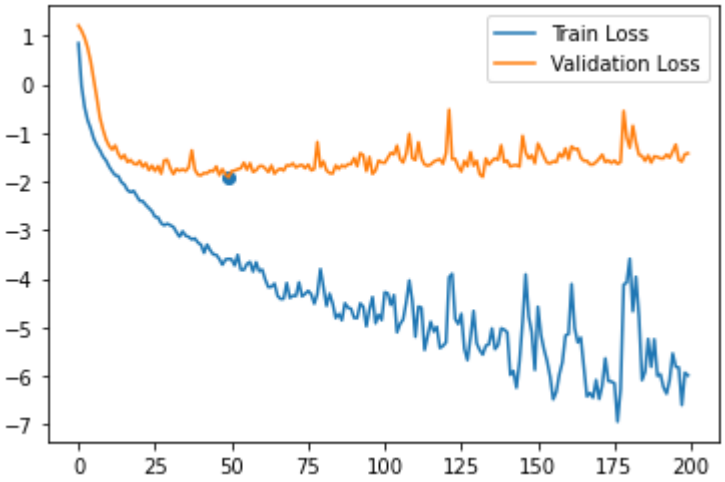
```
Model: "clasiffier1_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_72 (Sequential)  (None, 36)                64732

=================================================================
Total params: 64,732
Trainable params: 61,532
Non-trainable params: 3,200
_________________________________________________________________
```



In [ ]:
```
clasiffier2.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy
history5 = clasiffier2.fit(X_trainTensor, Y_trainTensor, epochs=200,
                           validation_data=(X_valTensor,Y_valTensor),shuffle=True)
```

In [ ]:
```
clasiffier2.summary()
pyplot.plot(np.log(history5.history['loss']), label='Train Loss')
pyplot.plot(np.log(history5.history['val_loss']), label='Validation Loss')
lowestIndex=np.argmin(history5.history['val_loss'])
pyplot.scatter(lowestIndex,np.log(history5.history['val_loss'][lowestIndex]))
pyplot.legend()
pyplot.show()
```

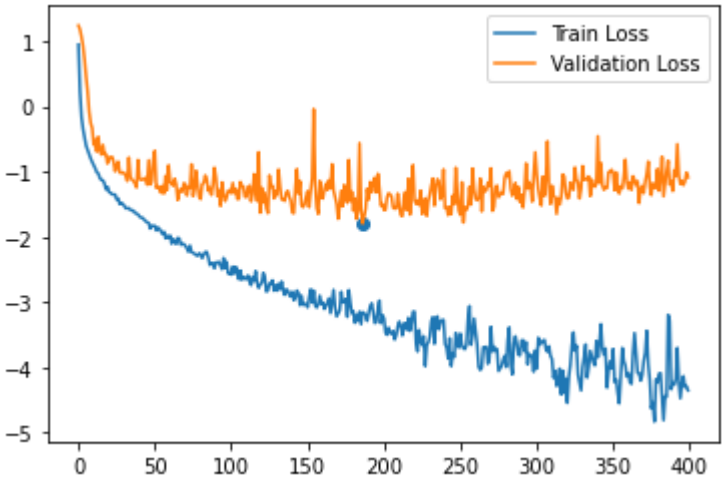
```
Model: "clasiffier2_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_73 (Sequential)  (None, 36)                16732

=================================================================
Total params: 16,732
Trainable params: 15,932
Non-trainable params: 800
_________________________________________________________________
```



In [ ]:
```
clasiffier3.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy
history6 = clasiffier3.fit(X_trainTensor, Y_trainTensor, epochs=400,
                           validation_data=(X_valTensor,Y_valTensor),shuffle=True)
```

In [ ]:
```
clasiffier3.summary()
pyplot.plot(np.log(history6.history['loss']), label='Train Loss')
pyplot.plot(np.log(history6.history['val_loss']), label='Validation Loss')
lowestIndex=np.argmin(history6.history['val_loss'])
pyplot.scatter(lowestIndex,np.log(history6.history['val_loss'][lowestIndex]))
pyplot.legend()
pyplot.show()
```

```
Model: "clasiffier3_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_74 (Sequential)  (None, 36)                16668

=================================================================
Total params: 16,668
Trainable params: 15,868
Non-trainable params: 800
_________________________________________________________________
```



summary:

during the training, accurading to val accuracy, all three models are able to beat 90% accuracy.

we also find clue about the points that each model will start to overfit.

as a result, we'll pick model 1 and 3, as the most accurate one and the smallest one.

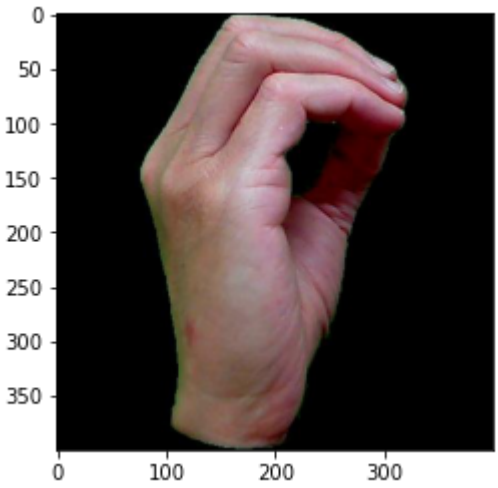
I'll retrain them near the stable range(i.e. 175,190) we got here and then determine the accuracy.

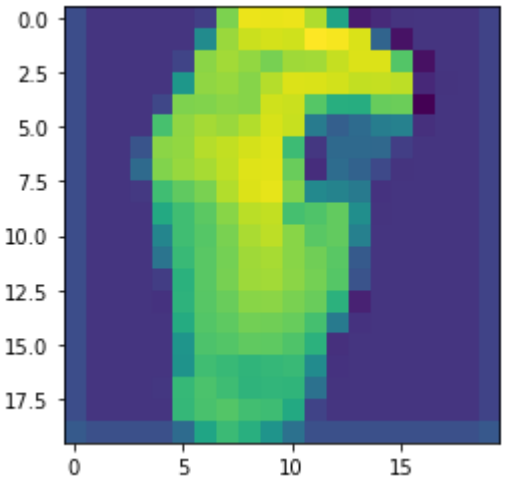
Testing:

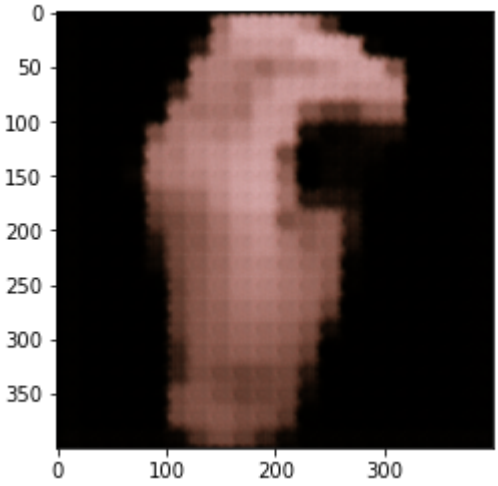
accurate model:

In [ ]:
```
class Model1AutoEncoder(keras.Model):
  def __init__(self):
    super(Model1AutoEncoder, self).__init__()
    self.encoder = tf.keras.models.Sequential([
      layers.MaxPool2D(pool_size=(5, 5),strides=(5, 5), padding='same'),
      layers.Conv2D(2, (3,3), activation = 'relu',strides=1,padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      layers.Conv2D(1, (3,3), activation = 'sigmoid',strides=1,padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      ])
    self.decoder = tf.keras.models.Sequential([
      layers.Conv2DTranspose(1,(5,5),strides=5,activation='relu', padding='same'),
      layers.Conv2DTranspose(4,(5,5),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(3,3),strides=2,activation='relu', padding='same'),
      layers.BatchNormalization(),
      layers.Conv2D(8, kernel_size=(3, 3), activation='relu', padding='same'),
      layers.Conv2D(3, kernel_size=(3, 3), activation='sigmoid', padding='same')
      ])

  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

model1AutoEncoder=Model1AutoEncoder()
model1AutoEncoder.compile(optimizer='adam',loss=keras.losses.MeanSquaredError())
history7 = model1AutoEncoder.fit(X_trainTensor, X_trainTensor,
                         validation_data=(X_valTensor,X_valTensor),
                         epochs=60,shuffle=True)
```

In [ ]:
```
pyplot.imshow(X_train[0]);
pyplot.show()
middleImage=model1AutoEncoder.encoder(X_train[[0]])
pyplot.imshow(middleImage[0].numpy());
pyplot.show()
newImage=model1AutoEncoder.decoder(middleImage)[0].numpy()
pyplot.imshow(newImage);
```

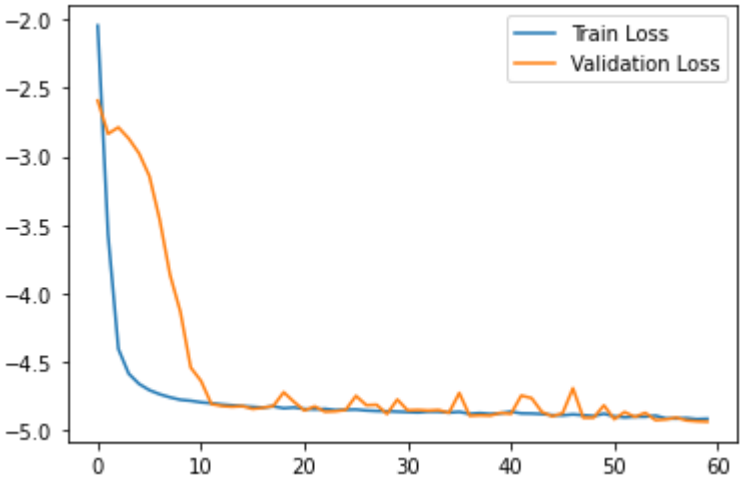






In [ ]:
```
modellAutoEncoder.summary()
pyplot.plot(np.log(history7.history['loss']), label='Train Loss')
pyplot.plot(np.log(history7.history['val_loss']), label='Validation Loss')
pyplot.legend()
pyplot.show()
```

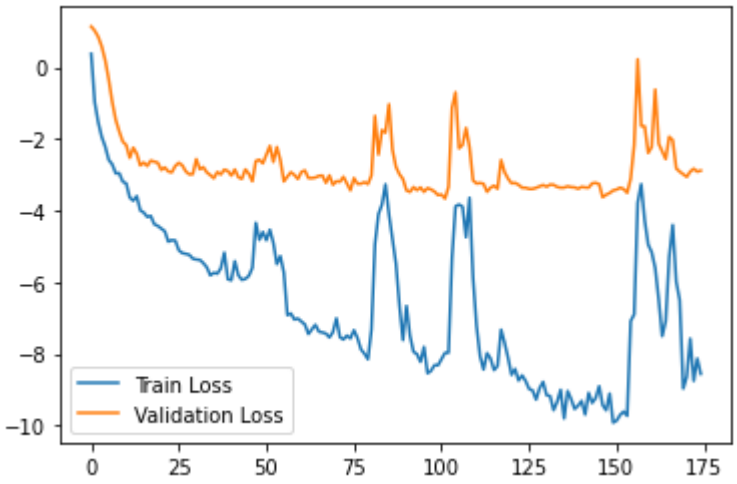
```
Model: "model1_auto_encoder_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_77 (Sequential)  (None, 20, 20, 1)         75

 sequential_78 (Sequential)  (None, 400, 400, 3)       1261

=================================================================
Total params: 1,336
Trainable params: 1,320
Non-trainable params: 16
_________________________________________________________________
```



In [ ]:
```
class Model1Clasiffier(keras.Model):
  def __init__(self):
    super(Model1Clasiffier, self).__init__()
    self.linear = tf.keras.models.Sequential([
        layers.Conv2D(4, (5,5), activation = 'relu',strides=1,padding='same'),
        layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
        layers.Conv2D(16, (3,3), activation = 'relu',strides=1,padding='same'),
        layers.Flatten(),
        layers.BatchNormalization(),
        layers.Dense(36,activation='softmax')
      ])

  def call(self, x):
    encoded = model1AutoEncoder.encoder(x)
    return self.linear(encoded)

model1 = Model1Clasiffier()
model1.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
history8 = model1.fit(X_trainTensor, Y_trainTensor, epochs=175,
                      validation_data=(X_valTensor,Y_valTensor),shuffle=True)
```

In [ ]:
```
model1.summary()
pyplot.plot(np.log(history8.history['loss']), label='Train Loss')
pyplot.plot(np.log(history8.history['val_loss']), label='Validation Loss')
pyplot.legend()
pyplot.show()
model1.evaluate(X_test,y_test)
```

```
Model: "model1_clasiffier_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_88 (Sequential)  (None, 36)                64732

=================================================================
Total params: 64,732
Trainable params: 61,532
Non-trainable params: 3,200
_________________________________________________________________
```




```
8/8 [==============================] - 0s 15ms/step - loss: 0.1877 - accuracy: 0.9721
```

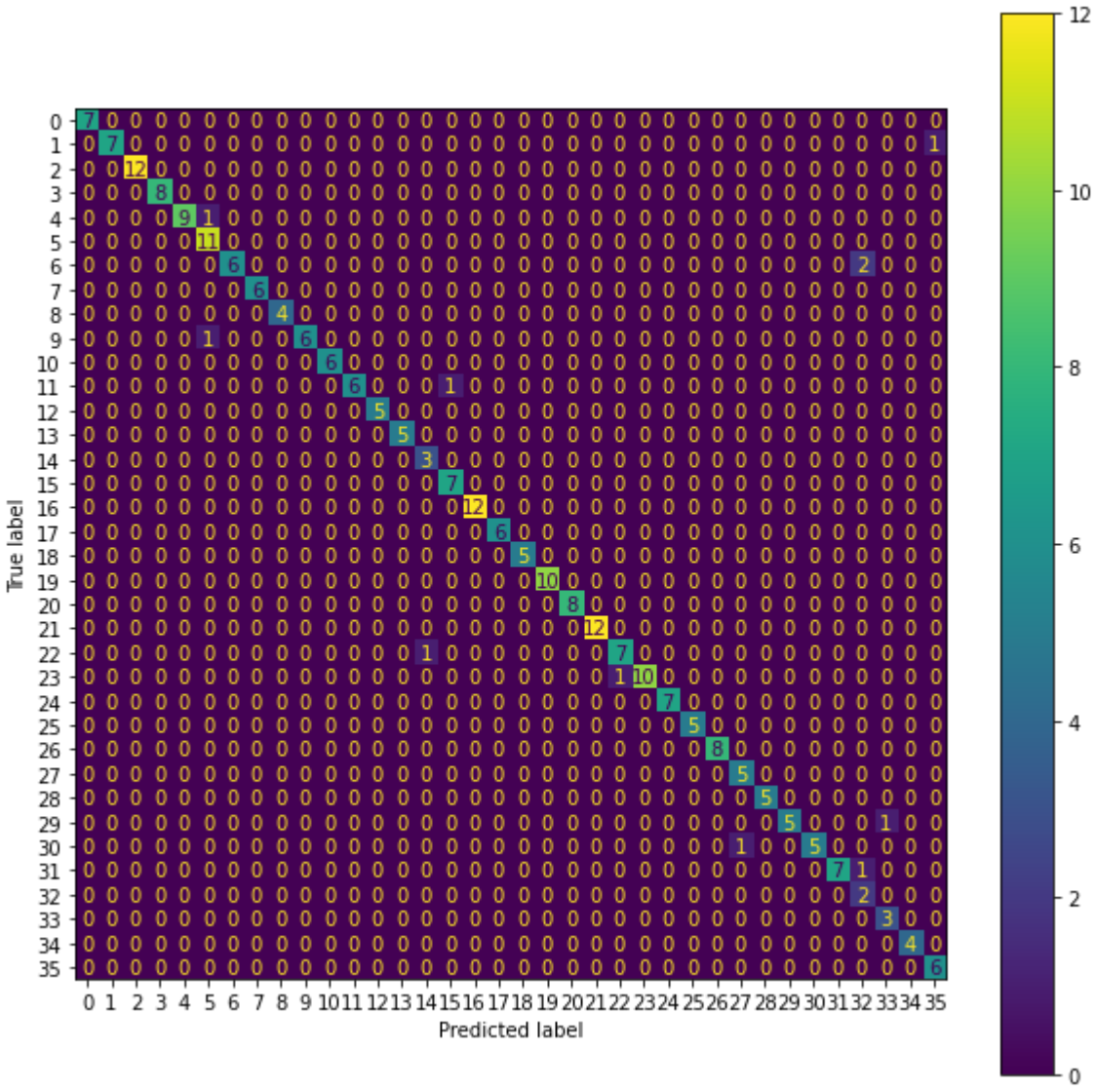
Out[ ]:
```
[0.18773959577083588, 0.9721115827560425]
```

In [ ]:
```
class_test=np.argmax(y_test, axis=1)
y_pred=model1.predict(X_test)
y_pred=np.argmax(y_pred, axis=1)
print(classification_report(y_pred, class_test))
```

```
8/8 [==============================] - 0s 16ms/step
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7
1	1.00	0.78	0.88	9
2	1.00	1.00	1.00	12
3	1.00	1.00	1.00	8
4	1.00	0.90	0.95	10
5	0.85	1.00	0.92	11
6	1.00	0.86	0.92	7
7	1.00	1.00	1.00	6
8	1.00	1.00	1.00	4
9	1.00	1.00	1.00	6
10	1.00	1.00	1.00	6
11	1.00	1.00	1.00	6
12	1.00	1.00	1.00	5
13	1.00	1.00	1.00	5
14	1.00	1.00	1.00	4
15	1.00	1.00	1.00	8
16	1.00	1.00	1.00	12
17	1.00	1.00	1.00	6
18	1.00	1.00	1.00	5
19	1.00	1.00	1.00	10
20	1.00	1.00	1.00	8
21	1.00	1.00	1.00	12
22	0.88	1.00	0.93	7
23	1.00	0.91	0.95	11
24	1.00	0.88	0.93	8
25	1.00	1.00	1.00	5
26	1.00	1.00	1.00	8
27	1.00	1.00	1.00	6
28	0.80	1.00	0.89	4
29	1.00	1.00	1.00	5
30	1.00	1.00	1.00	5
31	1.00	1.00	1.00	7
32	0.80	1.00	0.89	4
33	0.75	1.00	0.86	3
34	1.00	1.00	1.00	4
35	0.86	0.86	0.86	7
accuracy			0.97	251
macro avg	0.97	0.98	0.97	251
weighted avg	0.98	0.97	0.97	251

In [ ]:
```
cm=ConfusionMatrixDisplay(confusion_matrix(y_pred, class_test))
fig, ax = pyplot.subplots(figsize=(10,10))
cm.plot(ax=ax);
```

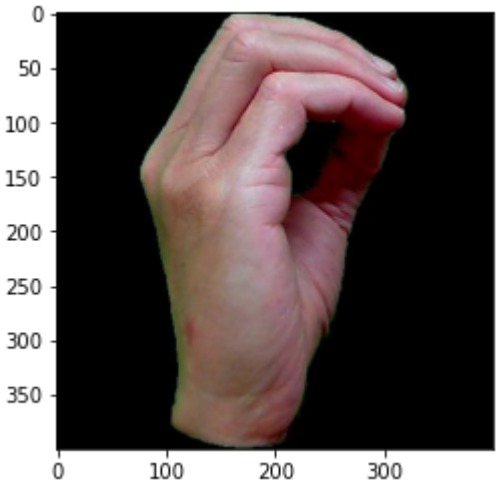


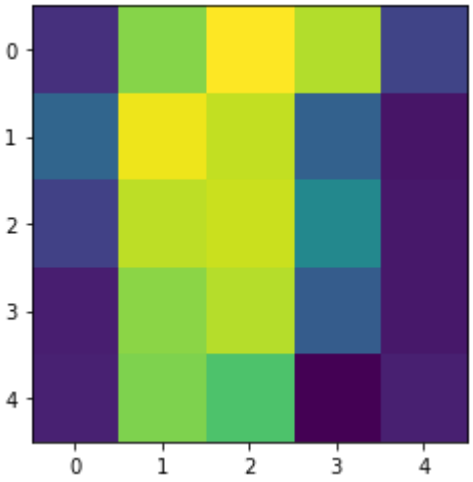
SmallModel

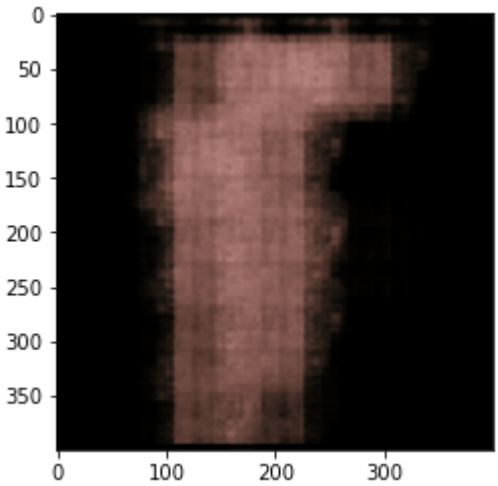
In [ ]:
```
class Model2AutoEncoder(keras.Model):
  def __init__(self):
    super(Model2AutoEncoder, self).__init__()
    self.encoder = tf.keras.models.Sequential([
      layers.MaxPool2D(pool_size=(10, 10),strides=(10, 10), padding='same'),
      layers.Conv2D(4, (3,3),strides=1,activation='relu', padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      layers.Conv2D(2, (3,3),strides=1,activation="relu",padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      layers.Conv2D(1, (3,3),strides=1,activation="sigmoid",padding='same'),
      layers.MaxPool2D(pool_size=(2, 2),strides=(2, 2), padding='same'),
      ])
    self.decoder = tf.keras.models.Sequential([
      layers.Conv2DTranspose(1,(5,5),strides=5,activation='relu', padding='same'),
      layers.Conv2DTranspose(4,(5,5),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(4,(5,5),strides=2,activation='relu', padding='same'),
      layers.BatchNormalization(),
      layers.Conv2DTranspose(8,(3,3),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(3,3),strides=2,activation='relu', padding='same'),
      layers.Conv2DTranspose(8,(3,3),strides=1,activation='relu', padding='same'),
      layers.Conv2D(3, kernel_size=(3, 3), activation='sigmoid', padding='same')])
```

```
  def call(self, x):
    encoded = self.encoder(x)
    decoded = self.decoder(encoded)
    return decoded

model2AutoEncoder=Model2AutoEncoder()
model2AutoEncoder.compile(optimizer='adam',loss=keras.losses.MeanSquaredError())
history9 = model2AutoEncoder.fit(X_trainTensor, X_trainTensor,
                         validation_data=(X_valTensor,X_valTensor),
                         epochs=60,shuffle=True)
```

In [ ]:
```
pyplot.imshow(X_train[0]);
pyplot.show()
middleImage=model2AutoEncoder.encoder(X_train[[0]])
pyplot.imshow(middleImage[0].numpy());
pyplot.show()
newImage=model2AutoEncoder.decoder(middleImage)[0].numpy()
pyplot.imshow(newImage);
```

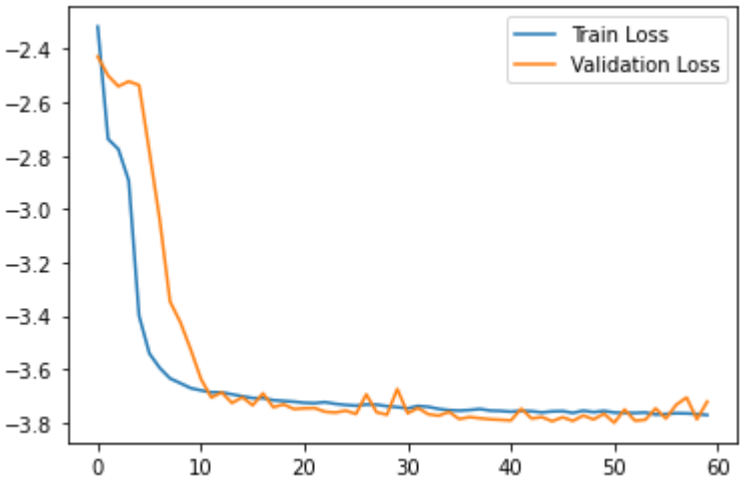






In [ ]:
```
model2AutoEncoder.summary()
pyplot.plot(np.log(history9.history['loss']), label='Train Loss')
pyplot.plot(np.log(history9.history['val_loss']), label='Validation Loss')
pyplot.legend()
pyplot.show()
```

```
Model: "model2_auto_encoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_82 (Sequential)  (None, 5, 5, 1)           205

 sequential_83 (Sequential)  (None, 400, 400, 3)       2233

=================================================================
Total params: 2,438
Trainable params: 2,430
Non-trainable params: 8
_________________________________________________________________
```



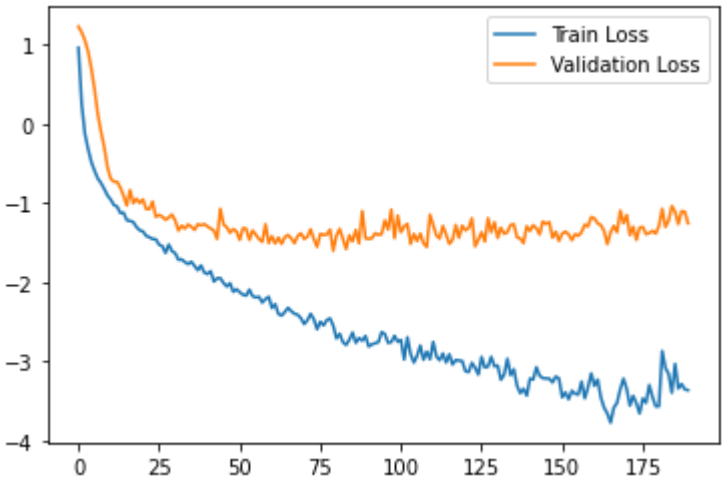
Out[ ]:
```
8/8 [==============================] - 0s 12ms/step - loss: 3.5849 - accuracy: 0.0159
[3.5848546028137207, 0.01593625545501709]
```

In [ ]:
```
class Model2Clasiffier(keras.Model):
    def __init__(self):
      super(Model2Clasiffier, self).__init__()
      self.linear = tf.keras.models.Sequential([
        layers.Conv2D(4, (3,3), activation = 'relu',strides=1,padding='same'),
```

```
        layers.Conv2D(16, (3,3), activation = 'relu',strides=1,padding='same'),
        layers.Flatten(),
        layers.BatchNormalization(),
        layers.Dense(36,activation='softmax')
      ])

    def call(self, x):
      encoded = model2AutoEncoder.encoder(x)
      return self.linear(encoded)

model2 = Model2Clasiffier()
model2.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
history10 = model2.fit(X_trainTensor, Y_trainTensor, epochs=190,
                       validation_data=(X_valTensor,Y_valTensor),shuffle=True)
```

In [ ]:
```
model2.summary()
pyplot.plot(np.log(history10.history['loss']), label='Train Loss')
pyplot.plot(np.log(history10.history['val_loss']), label='Validation Loss')
pyplot.legend()
pyplot.show()
model2.evaluate(X_test,y_test)
```

```
Model: "model2_clasiffier_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_86 (Sequential)  (None, 36)                16668

=================================================================
Total params: 16,668
Trainable params: 15,868
Non-trainable params: 800
_________________________________________________________________
```



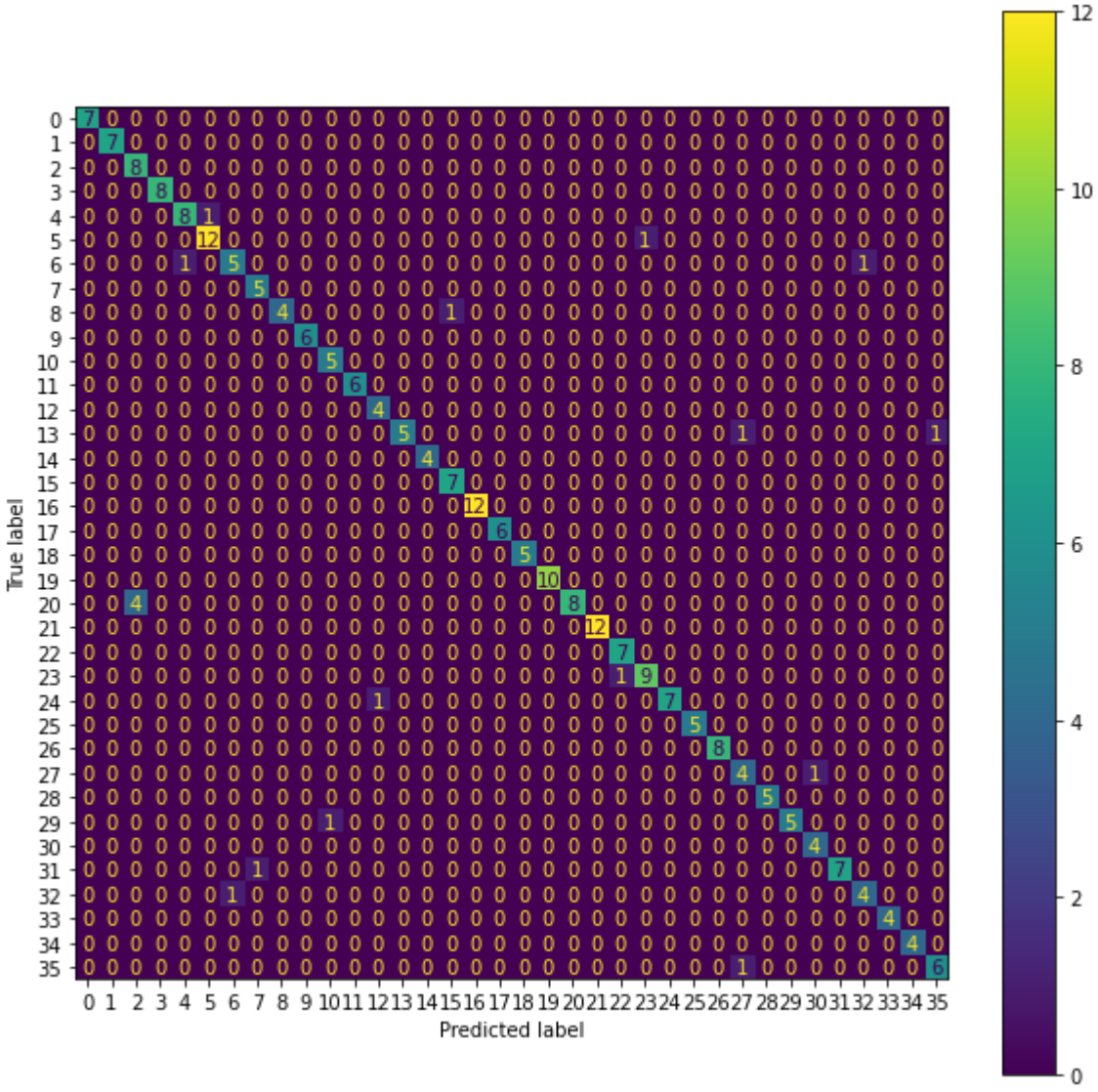
Out[ ]:
```
8/8 [==============================] - 0s 12ms/step - loss: 0.3020 - accuracy: 0.9283
[0.3019736111164093, 0.9282868504524231]
```

In [ ]:
```
class_test=np.argmax(y_test, axis=1)
y_pred=model2.predict(X_test)
y_pred=np.argmax(y_pred, axis=1)
print(classification_report(y_pred, class_test))
```

```
8/8 [==============================] - 0s 12ms/step
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7
1	1.00	1.00	1.00	7
2	0.67	1.00	0.80	8
3	1.00	1.00	1.00	8
4	0.89	0.89	0.89	9
5	0.92	0.92	0.92	13
6	0.83	0.71	0.77	7
7	0.83	1.00	0.91	5
8	1.00	0.80	0.89	5
9	1.00	1.00	1.00	6
10	0.83	1.00	0.91	5
11	1.00	1.00	1.00	6
12	0.80	1.00	0.89	4
13	1.00	0.71	0.83	7
14	1.00	1.00	1.00	4
15	0.88	1.00	0.93	7
16	1.00	1.00	1.00	12
17	1.00	1.00	1.00	6
18	1.00	1.00	1.00	5
19	1.00	1.00	1.00	10
20	1.00	0.67	0.80	12
21	1.00	1.00	1.00	12
22	0.88	1.00	0.93	7
23	0.90	0.90	0.90	10
24	1.00	0.88	0.93	8
25	1.00	1.00	1.00	5
26	1.00	1.00	1.00	8
27	0.67	0.80	0.73	5
28	1.00	1.00	1.00	5
29	1.00	0.83	0.91	6
30	0.80	1.00	0.89	4
31	1.00	0.88	0.93	8
32	0.80	0.80	0.80	5
33	1.00	1.00	1.00	4
34	1.00	1.00	1.00	4
35	0.86	0.86	0.86	7
accuracy			0.93	251
macro avg	0.93	0.93	0.93	251
weighted avg	0.94	0.93	0.93	251

In [ ]:
```
cm=ConfusionMatrixDisplay(confusion_matrix(y_pred, class_test))
fig, ax = pyplot.subplots(figsize=(10,10))
cm.plot(ax=ax);
```


True label
Predicted label
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
12
10
8
6
4
2
0