```
import numpy as np
import pandas as pd
import tensorflow as tf


trainX = np.load("trainX.npy")
trainY = np.load("trainY.npy")
testX = np.load("testX.npy")
testY = np.load("testY.npy")

print(trainX.shape)
print(trainY.shape)
print(testX.shape)
print(testY.shape)

trainY = tf.keras.utils.to_categorical(trainY, num_classes=20)
testY = tf.keras.utils.to_categorical(testY, num_classes=20)
```

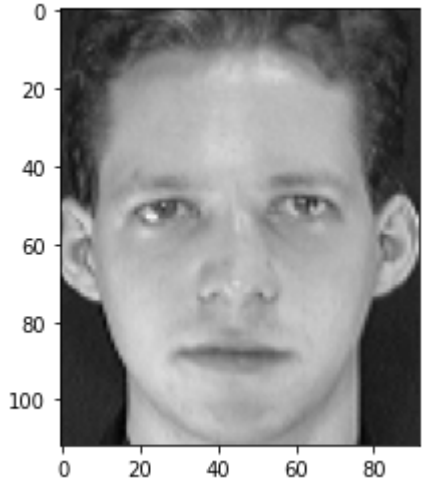
```
(240, 10304)
(240,)
(160, 10304)
(160,)
```

```
trainX = trainX.reshape(240,112,92,1)
testX = testX.reshape(160,112,92,1)


trainX.shape
```

```
(240, 112, 92, 1)
```

```
import matplotlib.pyplot as plt

plt.imshow(trainX.reshape(240,112,92)[0],cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x7fa0a30dc810>
```




```
import tensorflow as tf
```

```
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt

model = models.Sequential()
model.add(layers.Reshape((112,92,1),input_shape=(112,92,1)))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(112,92,1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(20, activation='softmax'))


model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy


model.summary()
```

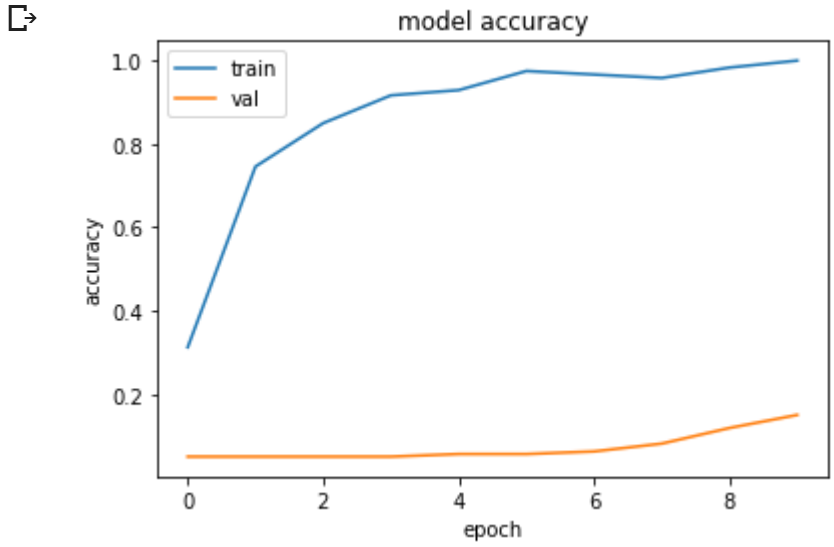
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 reshape_1 (Reshape)          (None, 112, 92, 1)        0

 batch_normalization_1 (Batc  (None, 112, 92, 1)       4
 hNormalization)

 conv2d_3 (Conv2D)            (None, 110, 90, 32)       320

 max_pooling2d_2 (MaxPooling  (None, 55, 45, 32)       0
 2D)

 conv2d_4 (Conv2D)            (None, 53, 43, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 26, 21, 64)       0
 2D)

 conv2d_5 (Conv2D)            (None, 24, 19, 64)        36928

 flatten_1 (Flatten)          (None, 29184)             0

 dense_1 (Dense)              (None, 20)                583700

=================================================================
Total params: 639,448
Trainable params: 639,446
Non-trainable params: 2
_________________________________________________________________
```

```
history = model.fit(trainX,trainY,
          validation_data=(testX,testY),
          epochs=10,
          batch_size=16)
```

```
Epoch 1/10
15/15 [==============================] - 5s 315ms/step - loss: 2.4035 - accur
```

```
Epoch 2/10
15/15 [==============================] - 5s 303ms/step - loss: 0.7899 - accur
Epoch 3/10
15/15 [==============================] - 5s 306ms/step - loss: 0.3920 - accur
Epoch 4/10
15/15 [==============================] - 5s 304ms/step - loss: 0.2331 - accur
Epoch 5/10
15/15 [==============================] - 4s 300ms/step - loss: 0.1794 - accur
Epoch 6/10
15/15 [==============================] - 4s 300ms/step - loss: 0.0883 - accur
Epoch 7/10
15/15 [==============================] - 4s 303ms/step - loss: 0.0646 - accur
Epoch 8/10
15/15 [==============================] - 4s 300ms/step - loss: 0.1083 - accur
Epoch 9/10
15/15 [==============================] - 4s 298ms/step - loss: 0.0677 - accur
Epoch 10/10
15/15 [==============================] - 4s 303ms/step - loss: 0.0123 - accur
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



✓ 0s completed at 08:30