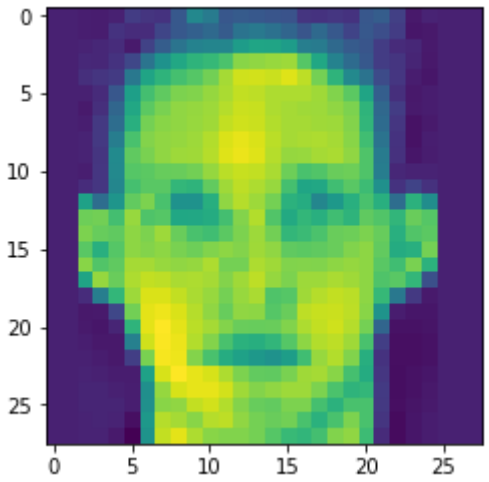
Show Code

```
Using TensorFlow backend.
```

Step 1: Load the att_faces dataset.

A data loading function is already provided in loader.py. It uses the first 8 images for each identity as training images; the 9th image for validation, and the 10th image for testing. The images are downsampled and padded to have size 28x28 (same as MNIST).



Step 2: Calculate the average intensity for the training images and subtract it from all images.

```
Average intensity: 0.40
```

Step 3: Set up a LeNet-style neural network for multi-class classification.

The model should have these layers in order:

- 20 5x5 Convolutions
- 2x2 Max Pooling with stride of 2
- 50 5x5 Convolutions
- 2x2 Max Pooling with stride of 2
- Fully connected layer: 512 outputs with ReLU activation
- Fully connected layer: 40 outputs (one per identity class) with Softmax activation

```
print(model.summary())
```

will print a text summary of the model.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 28, 28)            0         
_________________________________________________________________
reshape_1 (Reshape)          (None, 28, 28, 1)         0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 20)        520       
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 20)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 50)        25050     
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 50)          0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 2450)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1254912   
_________________________________________________________________
dense_2 (Dense)              (None, 40)                20520     
=================================================================
Total params: 1,301,002
Trainable params: 1,301,002
Non-trainable params: 0
_________________________________________________________________
```

Step 4: Train the model on the training data.

Use the 'categorical_crossentropy' loss from Keras -- this is the cross-entropy loss for multi-class classification with Softmax.

Use the SGD optimizer with a learning rate of 1e-2, 60 epochs, and a batch size of 32.

Pass the validation data as well and tell the model to compute the 'categorical_accuracy' metric.

```
Train on 320 samples, validate on 40 samples
Epoch 1/60
320/320 [==============================] - 0s - loss: 3.7161 - categori
cal_accuracy: 0.0000e+00 - val_loss: 3.6739 - val_categorical_accuracy:
0.0000e+00
Epoch 2/60
320/320 [==============================] - 0s - loss: 3.6500 - categori
cal_accuracy: 0.0281 - val_loss: 3.6228 - val_categorical_accuracy: 0.0
500
Epoch 3/60
320/320 [==============================] - 0s - loss: 3.5979 - categori
cal_accuracy: 0.0500 - val_loss: 3.5796 - val_categorical_accuracy: 0.1
000
Epoch 4/60
320/320 [==============================] - 0s - loss: 3.5529 - categori
cal_accuracy: 0.1187 - val_loss: 3.5402 - val_categorical_accuracy: 0.1
750
Epoch 5/60
320/320 [==============================] - 0s - loss: 3.5089 - categori
```

```
cal_accuracy: 0.1906 - val_loss: 3.4997 - val_categorical_accuracy: 0.2
000
Epoch 6/60
320/320 [==============================] - 0s - loss: 3.4653 - categori
cal_accuracy: 0.1844 - val_loss: 3.4567 - val_categorical_accuracy: 0.2
000
Epoch 7/60
320/320 [==============================] - 0s - loss: 3.4125 - categori
cal_accuracy: 0.2750 - val_loss: 3.4098 - val_categorical_accuracy: 0.2
500
Epoch 8/60
320/320 [==============================] - 0s - loss: 3.3602 - categori
cal_accuracy: 0.2687 - val_loss: 3.3593 - val_categorical_accuracy: 0.3
250
Epoch 9/60
320/320 [==============================] - 0s - loss: 3.3008 - categori
cal_accuracy: 0.3031 - val_loss: 3.3031 - val_categorical_accuracy: 0.3
250
Epoch 10/60
320/320 [==============================] - 0s - loss: 3.2333 - categori
cal_accuracy: 0.3719 - val_loss: 3.2375 - val_categorical_accuracy: 0.3
250
Epoch 11/60
320/320 [==============================] - 0s - loss: 3.1620 - categori
cal_accuracy: 0.3094 - val_loss: 3.1628 - val_categorical_accuracy: 0.3
250
Epoch 12/60
320/320 [==============================] - 0s - loss: 3.0777 - categori
cal_accuracy: 0.3719 - val_loss: 3.0786 - val_categorical_accuracy: 0.3
750
Epoch 13/60
320/320 [==============================] - 0s - loss: 2.9841 - categori
cal_accuracy: 0.4156 - val_loss: 2.9824 - val_categorical_accuracy: 0.3
750
Epoch 14/60
320/320 [==============================] - 0s - loss: 2.8791 - categori
cal_accuracy: 0.4188 - val_loss: 2.8770 - val_categorical_accuracy: 0.4
000
Epoch 15/60
320/320 [==============================] - 0s - loss: 2.7577 - categori
cal_accuracy: 0.4500 - val_loss: 2.7582 - val_categorical_accuracy: 0.3
750
Epoch 16/60
320/320 [==============================] - 0s - loss: 2.6204 - categori
cal_accuracy: 0.5094 - val_loss: 2.6346 - val_categorical_accuracy: 0.3
750
Epoch 17/60
320/320 [==============================] - 0s - loss: 2.4793 - categori
cal_accuracy: 0.4688 - val_loss: 2.4877 - val_categorical_accuracy: 0.5
250
Epoch 18/60
320/320 [==============================] - 0s - loss: 2.3294 - categori
cal_accuracy: 0.5938 - val_loss: 2.3358 - val_categorical_accuracy: 0.5
500
Epoch 19/60
320/320 [==============================] - 0s - loss: 2.1844 - categori
cal_accuracy: 0.6063 - val_loss: 2.1988 - val_categorical_accuracy: 0.4
```

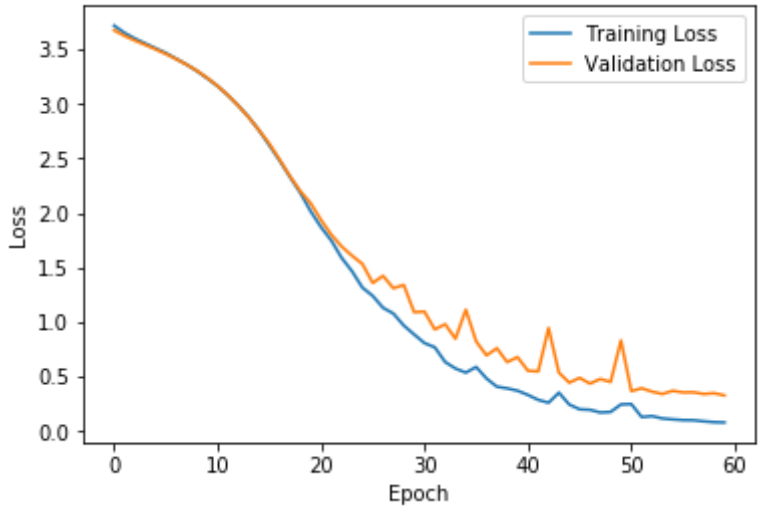
```
750
Epoch 20/60
320/320 [==============================] - 0s - loss: 2.0147 - categori
cal_accuracy: 0.6469 - val_loss: 2.0884 - val_categorical_accuracy: 0.5
250
Epoch 21/60
320/320 [==============================] - 0s - loss: 1.8699 - categori
cal_accuracy: 0.6594 - val_loss: 1.9354 - val_categorical_accuracy: 0.6
000
Epoch 22/60
320/320 [==============================] - 0s - loss: 1.7463 - categori
cal_accuracy: 0.6312 - val_loss: 1.7991 - val_categorical_accuracy: 0.5
750
Epoch 23/60
320/320 [==============================] - 0s - loss: 1.5874 - categori
cal_accuracy: 0.7188 - val_loss: 1.6922 - val_categorical_accuracy: 0.6
250
Epoch 24/60
320/320 [==============================] - 0s - loss: 1.4653 - categori
cal_accuracy: 0.7375 - val_loss: 1.6073 - val_categorical_accuracy: 0.6
750
Epoch 25/60
320/320 [==============================] - 0s - loss: 1.3157 - categori
cal_accuracy: 0.7812 - val_loss: 1.5325 - val_categorical_accuracy: 0.6
000
Epoch 26/60
320/320 [==============================] - 0s - loss: 1.2392 - categori
cal_accuracy: 0.7344 - val_loss: 1.3586 - val_categorical_accuracy: 0.6
750
Epoch 27/60
320/320 [==============================] - 0s - loss: 1.1303 - categori
cal_accuracy: 0.7906 - val_loss: 1.4223 - val_categorical_accuracy: 0.7
000
Epoch 28/60
320/320 [==============================] - 0s - loss: 1.0774 - categori
cal_accuracy: 0.7812 - val_loss: 1.3099 - val_categorical_accuracy: 0.6
000
Epoch 29/60
320/320 [==============================] - 0s - loss: 0.9681 - categori
cal_accuracy: 0.8156 - val_loss: 1.3398 - val_categorical_accuracy: 0.6
000
Epoch 30/60
320/320 [==============================] - 0s - loss: 0.8861 - categori
cal_accuracy: 0.8313 - val_loss: 1.0904 - val_categorical_accuracy: 0.8
000
Epoch 31/60
320/320 [==============================] - 0s - loss: 0.8073 - categori
cal_accuracy: 0.8438 - val_loss: 1.0953 - val_categorical_accuracy: 0.7
250
Epoch 32/60
320/320 [==============================] - 0s - loss: 0.7683 - categori
cal_accuracy: 0.8531 - val_loss: 0.9323 - val_categorical_accuracy: 0.7
750
Epoch 33/60
320/320 [==============================] - 0s - loss: 0.6341 - categori
cal_accuracy: 0.9219 - val_loss: 0.9797 - val_categorical_accuracy: 0.7
500
```

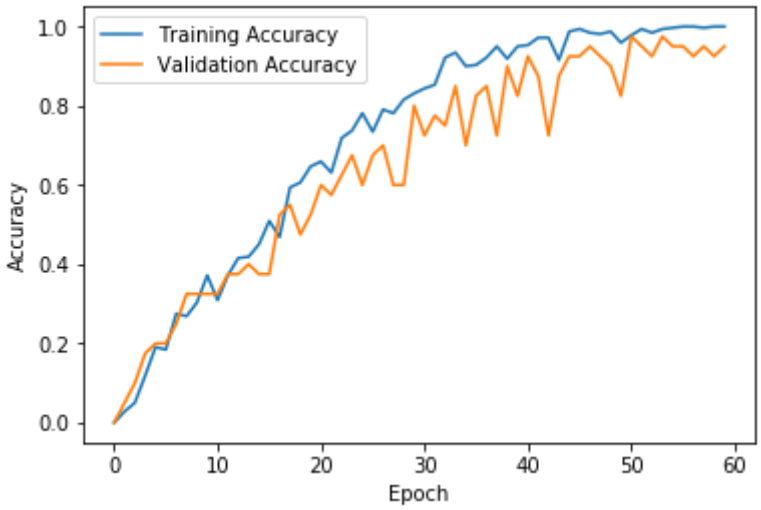
```
Epoch 34/60
320/320 [==============================] - 0s - loss: 0.5739 - categori
cal_accuracy: 0.9344 - val_loss: 0.8471 - val_categorical_accuracy: 0.8
500
Epoch 35/60
320/320 [==============================] - 0s - loss: 0.5362 - categori
cal_accuracy: 0.9000 - val_loss: 1.1124 - val_categorical_accuracy: 0.7
000
Epoch 36/60
320/320 [==============================] - 0s - loss: 0.5880 - categori
cal_accuracy: 0.9031 - val_loss: 0.8222 - val_categorical_accuracy: 0.8
250
Epoch 37/60
320/320 [==============================] - 0s - loss: 0.4821 - categori
cal_accuracy: 0.9219 - val_loss: 0.6944 - val_categorical_accuracy: 0.8
500
Epoch 38/60
320/320 [==============================] - 0s - loss: 0.4074 - categori
cal_accuracy: 0.9500 - val_loss: 0.7596 - val_categorical_accuracy: 0.7
250
Epoch 39/60
320/320 [==============================] - 0s - loss: 0.3921 - categori
cal_accuracy: 0.9187 - val_loss: 0.6330 - val_categorical_accuracy: 0.9
000
Epoch 40/60
320/320 [==============================] - 0s - loss: 0.3705 - categori
cal_accuracy: 0.9500 - val_loss: 0.6780 - val_categorical_accuracy: 0.8
250
Epoch 41/60
320/320 [==============================] - 0s - loss: 0.3320 - categori
cal_accuracy: 0.9531 - val_loss: 0.5540 - val_categorical_accuracy: 0.9
250
Epoch 42/60
320/320 [==============================] - 0s - loss: 0.2865 - categori
cal_accuracy: 0.9719 - val_loss: 0.5471 - val_categorical_accuracy: 0.8
750
Epoch 43/60
320/320 [==============================] - 0s - loss: 0.2603 - categori
cal_accuracy: 0.9719 - val_loss: 0.9453 - val_categorical_accuracy: 0.7
250
Epoch 44/60
320/320 [==============================] - 0s - loss: 0.3528 - categori
cal_accuracy: 0.9156 - val_loss: 0.5356 - val_categorical_accuracy: 0.8
750
Epoch 45/60
320/320 [==============================] - 0s - loss: 0.2439 - categori
cal_accuracy: 0.9875 - val_loss: 0.4442 - val_categorical_accuracy: 0.9
250
Epoch 46/60
320/320 [==============================] - 0s - loss: 0.2001 - categori
cal_accuracy: 0.9938 - val_loss: 0.4881 - val_categorical_accuracy: 0.9
250
Epoch 47/60
320/320 [==============================] - 0s - loss: 0.1954 - categori
cal_accuracy: 0.9844 - val_loss: 0.4358 - val_categorical_accuracy: 0.9
500
Epoch 48/60
```

```
320/320 [==============================] - 0s - loss: 0.1709 - categori
cal_accuracy: 0.9812 - val_loss: 0.4773 - val_categorical_accuracy: 0.9
250
Epoch 49/60
320/320 [==============================] - 0s - loss: 0.1760 - categori
cal_accuracy: 0.9875 - val_loss: 0.4512 - val_categorical_accuracy: 0.9
000
Epoch 50/60
320/320 [==============================] - 0s - loss: 0.2448 - categori
cal_accuracy: 0.9594 - val_loss: 0.8311 - val_categorical_accuracy: 0.8
250
Epoch 51/60
320/320 [==============================] - 0s - loss: 0.2482 - categori
cal_accuracy: 0.9781 - val_loss: 0.3671 - val_categorical_accuracy: 0.9
750
Epoch 52/60
320/320 [==============================] - 0s - loss: 0.1294 - categori
cal_accuracy: 0.9938 - val_loss: 0.3926 - val_categorical_accuracy: 0.9
500
Epoch 53/60
320/320 [==============================] - 0s - loss: 0.1377 - categori
cal_accuracy: 0.9844 - val_loss: 0.3621 - val_categorical_accuracy: 0.9
250
Epoch 54/60
320/320 [==============================] - 0s - loss: 0.1163 - categori
cal_accuracy: 0.9938 - val_loss: 0.3407 - val_categorical_accuracy: 0.9
750
Epoch 55/60
320/320 [==============================] - 0s - loss: 0.1075 - categori
cal_accuracy: 0.9969 - val_loss: 0.3694 - val_categorical_accuracy: 0.9
500
Epoch 56/60
320/320 [==============================] - 0s - loss: 0.1010 - categori
cal_accuracy: 1.0000 - val_loss: 0.3547 - val_categorical_accuracy: 0.9
500
Epoch 57/60
320/320 [==============================] - 0s - loss: 0.1000 - categori
cal_accuracy: 1.0000 - val_loss: 0.3564 - val_categorical_accuracy: 0.9
250
Epoch 58/60
320/320 [==============================] - 0s - loss: 0.0900 - categori
cal_accuracy: 0.9969 - val_loss: 0.3403 - val_categorical_accuracy: 0.9
500
Epoch 59/60
320/320 [==============================] - 0s - loss: 0.0819 - categori
cal_accuracy: 1.0000 - val_loss: 0.3481 - val_categorical_accuracy: 0.9
250
Epoch 60/60
320/320 [==============================] - 0s - loss: 0.0795 - categori
cal_accuracy: 1.0000 - val_loss: 0.3274 - val_categorical_accuracy: 0.9
500
```

Step 5: Plot the progress of loss and accuracy during training.

The mnist demo script shows an example of how to do this.





Step 6: Calculate accuracy on the training, validation and testing data.

Keras function model.evaluate() will calculate the accuracy for you, if you used the 'categorical_accuracy' as indicated above.

```
Training Accuracy: 100.00 %
Validation Accuracy: 95.00 %
Testing Accuracy: 95.00 %
```