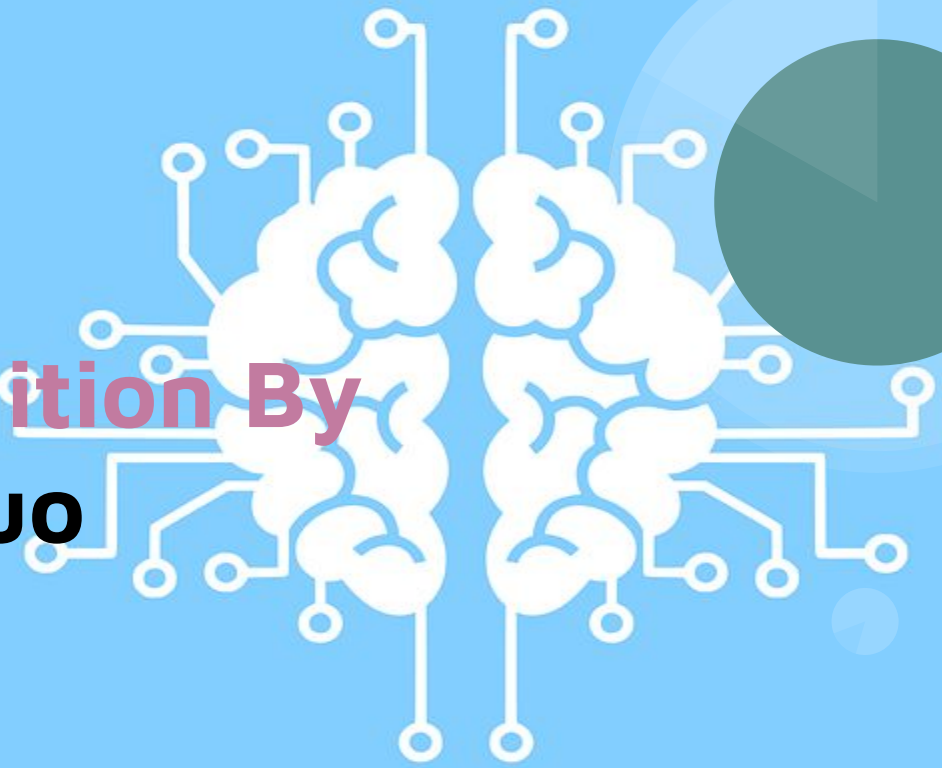
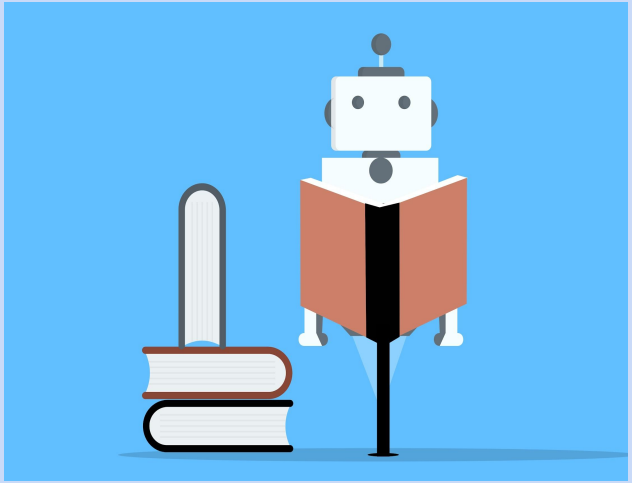
Letter Recognition By

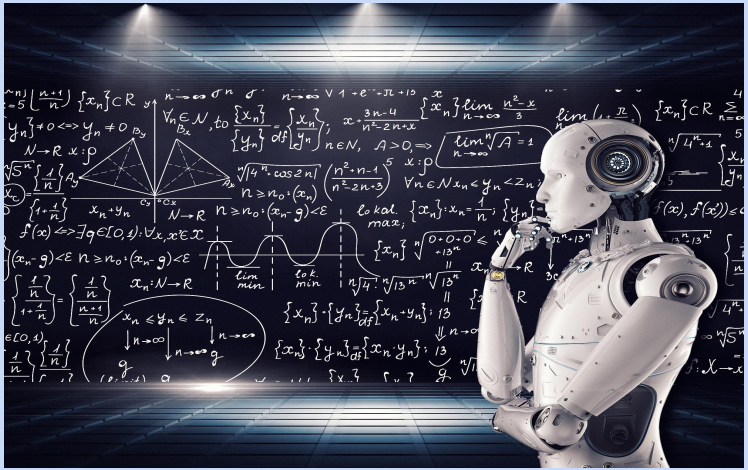
Yang Luo



Introduction



A popular demonstration of the capability of deep learning techniques is object recognition in image data.This deep learning application in python recognizes alphabet through gestures captured real-time on a webcam. The user is allowed to write the alphabet on the screen using an object-of-interest



Character Recognition Using ML

What we will do?

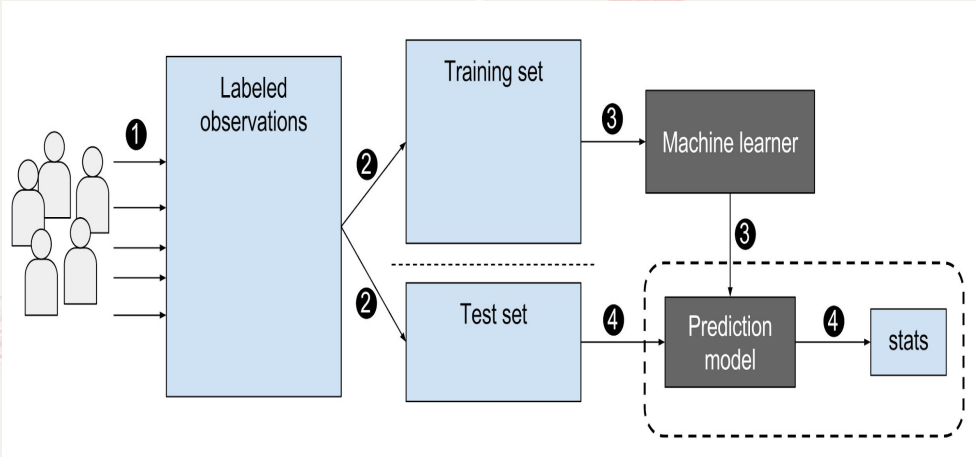
We will take data out which we will be separating them into two parts:

1 Test data

2 Training data

We will train our algorithm trough test data and training data to use for out ANN model. So our model is trained for letter recognition

After that we will use it for predicting purpose and calculate the accuracy of our model



Character Recognition Using ML

What we will do?
The code

We basically use sklearn library to import and split oru train data:

```
ImageDatas = []
files = dataset["image"]
label = dataset["label"]
i = 0
print("=======starting======")
for fileName in files:

image=tf.keras.utils.load_img(os.path.join(directory,fileName),color_
mode='grayscale',target_size=(100,100))
    image=tf.keras.utils.img_to_array(image)
    image=image/255.0
    imageDatas+=[image]
    i = i + 1
    if (i % 500 == 0):
        print(i)

class_names = dataset["label"].unique()
print(class_names)
print("=======done======")
```

```
=======starting======
500
1000
1500
2000
2500
3000
['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'
 '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']
=======done======
```

Character Recognition Using ML

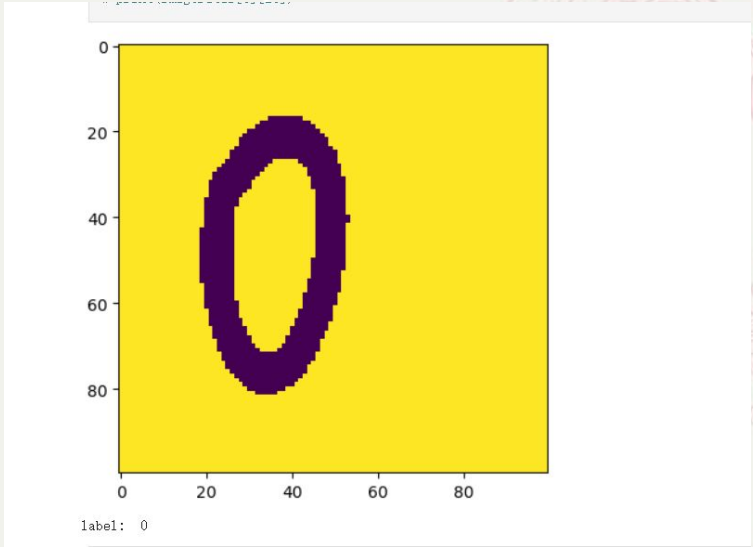
What we will do?

The code...

Then we verify the image data and print label imagedata result.

```
directory = "archive"
showingImage=tf.keras.utils.load_img(os.path.join(directory,files[0]
),color_mode='grayscale',target_size=(100,100))
plt.imshow(imageDatas[0])
plt.show()

print("label: ", label[0])
# print(imageDatas[0][20])
```



Character Recognition Using ML

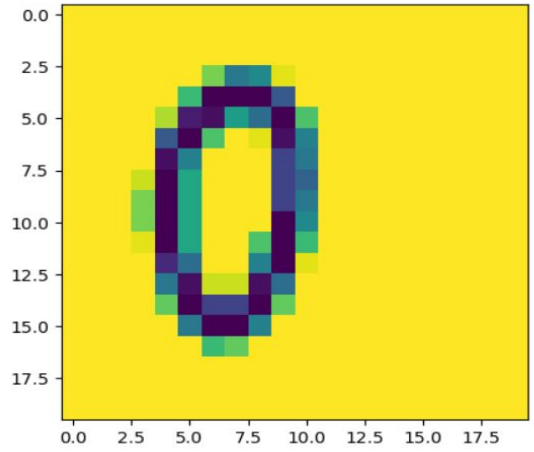
What we will do?
The code...

```
ImageData_compressed = []

print(np.array(imageDatas[0]).shape)
# test = []
# test += [imageDatas[0]]
count = 0
print("=====start compressing=====")
for img in imageDatas:
    img_compressed = []
    for i in range(20):
        row_compressed = []
        for j in range(20):
            sum = 0
            for r in range(5):
                for c in range(5):
                    sum += img[5 * i + r][5 *j + c]
            row_compressed += [sum/25]
            #
print(np.array(row_compressed).shape)
        img_compressed += [row_compressed]
        #
print(np.array(img_compressed).shape)
    ImageData_compressed += [img_compressed]
    count += 1
    if (count % 500 == 0):
        print("compressed: ", count, " imgs" )

print("=====done compressing=====")
print(np.array(ImageData_compressed).shape)
plt.imshow(ImageData_compressed[0])
# print(ImageData_compressed[0])
plt.show()
```

```
(100, 100, 1)
compressed:  500  imgs
compressed:  1000  imgs
compressed:  1500  imgs
compressed:  2000  imgs
compressed:  2500  imgs
compressed:  3000  imgs
==
(3410, 20, 20, 1)
```



Character Recognition Using ML

What we will do?
The code...

Then we process label to find match number

```
def getIndex(letter):
    index = ord(letter)-48
    if (index > 10):
        index -= 7
    if (index > 35):
        index -= 6
    return index

# process label to number
letters = label.unique()
print(letters)

# checking
indexs = []
for l in letters:
    indexs += [getIndex(l)]
```

```
print(indexs)
for i in range(61):
    if (i != indexs[i]):
        print("error in: ", i)
```

```
['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'
 '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']
[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, 36, 3
7, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61]
```

Character Recognition Using ML

What we will do?
The code...

we split data
data_nparr=np.array(ImageData_compressed)
#
data_nparr=np.array(ImageData_compressed_type2)
 # data_nparr=np.array(imageDatas)
label_nparr=np.array(label)
label_index_nparr =
np.vectorize(getIndex)(label_nparr)

train_data, test_data, train_label,
test_label = train_test_split(data_nparr,
label_index_nparr, test_size=0.1,
random_state=42)
print("data splited")

Then tensorflow library to import datasets, layers, models

import tensorflow **as** tf
from tensorflow.keras **import** datasets, layers, models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3),
activation='relu', input_shape=(20, 20, 1)))
 # compressed (20 x 20)
 # model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 1))) # uncompressed (100 x 100)
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'))

```
layer added
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 18, 18, 32)        320

max_pooling2d_6 (MaxPooling  (None, 9, 9, 32)          0
2D)

conv2d_13 (Conv2D)           (None, 7, 7, 64)          18496

max_pooling2d_7 (MaxPooling  (None, 3, 3, 64)          0
2D)

conv2d_14 (Conv2D)           (None, 1, 1, 64)          36928

flatten_3 (Flatten)          (None, 64)                0

dense_6 (Dense)              (None, 64)                4160

dense_7 (Dense)              (None, 62)                4030

=================================================================
Total params: 63,934
Trainable params: 63,934
Non-trainable params: 0
_________________________________________________________________
```

Character Recognition Using ML

What we will do?
The code...

```
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3),
activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64,
activation='relu'))
model.add(layers.Dense(62,
activation='softmax'))
print("layer added")
model.summary()
```

```
layer added
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 18, 18, 32)        320

max_pooling2d_6 (MaxPooling  (None, 9, 9, 32)          0
2D)

conv2d_13 (Conv2D)           (None, 7, 7, 64)          18496

max_pooling2d_7 (MaxPooling  (None, 3, 3, 64)          0
2D)

conv2d_14 (Conv2D)           (None, 1, 1, 64)          36928

flatten_3 (Flatten)          (None, 64)                0

dense_6 (Dense)              (None, 64)                4160

dense_7 (Dense)              (None, 62)                4030

=================================================================
Total params: 63,934
Trainable params: 63,934
Non-trainable params: 0
_________________________________________________________________
```

Character Recognition Using ML

What we will do?
The code...

```
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
              metrics=['accuracy'])

print("======start training======")
history = model.fit(train_data, train_label, epochs=30,
                    validation_data=(test_data, test_label))
print("======done training======")
```

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_data,  test_label, verbose=2)
```