In [3]:
```
import numpy as np
import pandas as pd
import os
```

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [4]:
```
testing_letter = pd.read_csv('/content/drive/MyDrive/emnist-letters-test.cs
training_letter = pd.read_csv('/content/drive/MyDrive/emnist-letters-train.
```

In [5]:
```
print(training_letter.shape)
print(testing_letter.shape)
```

```
(88799, 785)
(14799, 785)
```

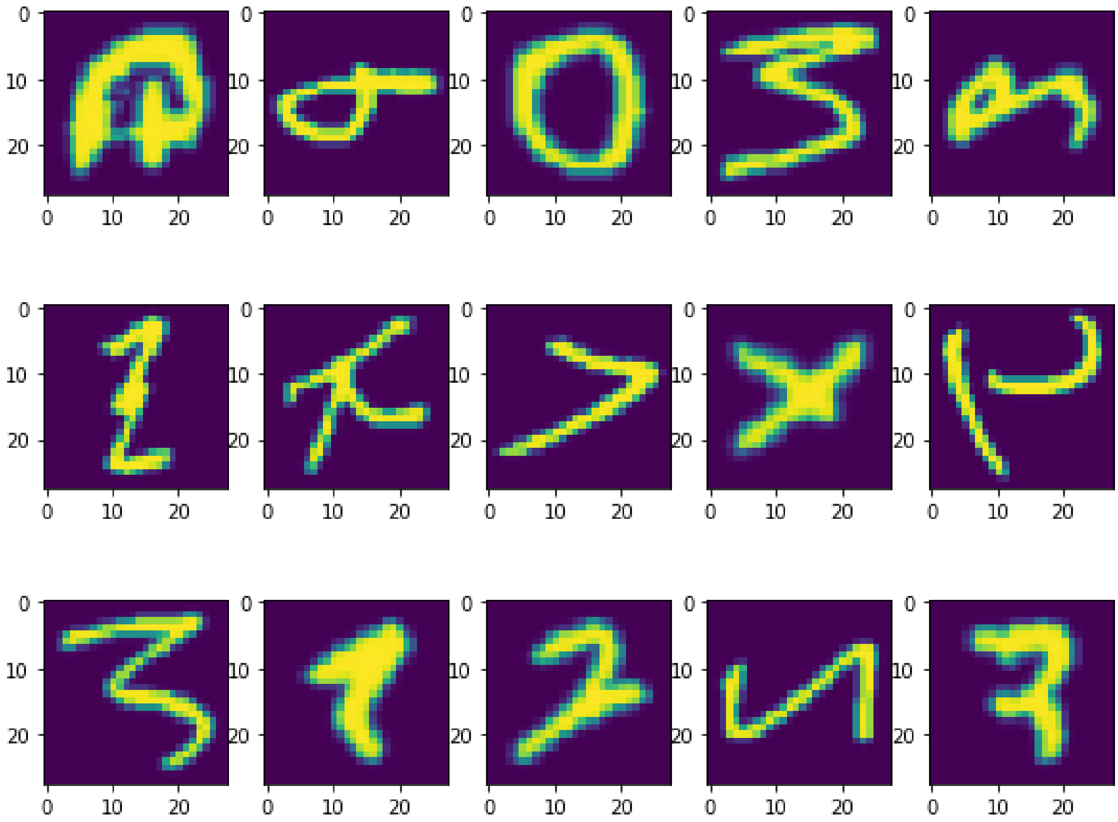
In [6]:
```
y1 = np.array(training_letter.iloc[:,0].values)
x1 = np.array(training_letter.iloc[:,1:].values)

y2 = np.array(testing_letter.iloc[:,0].values)
x2 = np.array(testing_letter.iloc[:,1:].values)
print(y1.shape)
print(x1.shape)
```

```
(88799,)
(88799, 784)
```

In [7]:

```
import matplotlib.pyplot as plt
fig,axes = plt.subplots(3,5,figsize=(10,8))
for i,ax in enumerate(axes.flat):
    ax.imshow(x1[i].reshape([28,28]))
```



In [8]:

```
import tensorflow as tf
```

In [9]:

```
train_images = x1 / 255.0
test_images = x2 / 255.0

train_images_number = train_images.shape[0]
train_images_height = 28
train_images_width = 28
train_images_size = train_images_height*train_images_width

train_images = train_images.reshape(train_images_number, train_images_heigh

test_images_number = test_images.shape[0]
test_images_height = 28
test_images_width = 28
test_images_size = test_images_height*test_images_width

test_images = test_images.reshape(test_images_number, test_images_height, t
```

In [10]:

```
number_of_classes = 37

y1 = tf.keras.utils.to_categorical(y1, number_of_classes)
y2 = tf.keras.utils.to_categorical(y2, number_of_classes)
```

In [11]:

```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau,Moc
```

In [12]:

```
from sklearn.model_selection import train_test_split
```

In [13]:

```
train_x,test_x,train_y,test_y = train_test_split(train_images,y1,test_size=
```

In [14]:

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32,3,input_shape=(28,28,1)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(input_shape=(28,28,1)),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.Dense(128,activation='relu'),
    tf.keras.layers.Dense(number_of_classes,activation='softmax')
])
```

In [15]:

```
model.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=[
```

In [16]:

```
MCP = ModelCheckpoint('Best_points.h5',verbose=1,save_best_only=True,monitc
ES = EarlyStopping(monitor='val_accuracy',min_delta=0,verbose=0,restore_bes
RLP = ReduceLROnPlateau(monitor='val_loss',patience=3,factor=0.2,min_lr=0.0
```

In [17]:
```
history = model.fit(train_x,train_y,epochs=10,validation_data=(test_x,test_
```

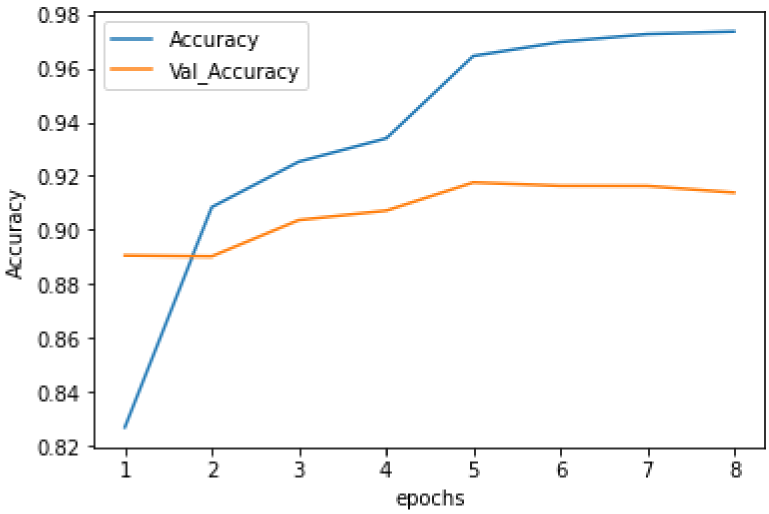
```
Epoch 1/10
2218/2220 [============================>.] - ETA: 0s - loss: 0.5546 - acc
uracy: 0.8268
Epoch 00001: val_accuracy improved from -inf to 0.89048, saving model to
Best_points.h5
2220/2220 [==============================] - 34s 10ms/step - loss: 0.5544
- accuracy: 0.8269 - val_loss: 0.3341 - val_accuracy: 0.8905 - lr: 0.0010
Epoch 2/10
2218/2220 [============================>.] - ETA: 0s - loss: 0.2822 - acc
uracy: 0.9084
Epoch 00002: val_accuracy did not improve from 0.89048
2220/2220 [==============================] - 22s 10ms/step - loss: 0.2823
- accuracy: 0.9084 - val_loss: 0.3510 - val_accuracy: 0.8901 - lr: 0.0010
Epoch 3/10
2219/2220 [============================>.] - ETA: 0s - loss: 0.2237 - acc
uracy: 0.9252
Epoch 00003: val_accuracy improved from 0.89048 to 0.90360, saving model
to Best_points.h5
2220/2220 [==============================] - 22s 10ms/step - loss: 0.2236
- accuracy: 0.9253 - val_loss: 0.3432 - val_accuracy: 0.9036 - lr: 0.0010
Epoch 4/10
2215/2220 [============================>.] - ETA: 0s - loss: 0.2001 - acc
uracy: 0.9339
Epoch 00004: val_accuracy improved from 0.90360 to 0.90709, saving model
to Best_points.h5
2220/2220 [==============================] - 22s 10ms/step - loss: 0.2001
- accuracy: 0.9339 - val_loss: 0.3540 - val_accuracy: 0.9071 - lr: 0.0010
Epoch 5/10
2217/2220 [============================>.] - ETA: 0s - loss: 0.0981 - acc
uracy: 0.9645
Epoch 00005: val_accuracy improved from 0.90709 to 0.91757, saving model
to Best_points.h5
2220/2220 [==============================] - 22s 10ms/step - loss: 0.0981
- accuracy: 0.9645 - val_loss: 0.3047 - val_accuracy: 0.9176 - lr: 2.0000
e-04
Epoch 6/10
2219/2220 [============================>.] - ETA: 0s - loss: 0.0807 - acc
uracy: 0.9697
Epoch 00006: val_accuracy did not improve from 0.91757
2220/2220 [==============================] - 22s 10ms/step - loss: 0.0807
- accuracy: 0.9697 - val_loss: 0.3243 - val_accuracy: 0.9163 - lr: 2.0000
e-04
Epoch 7/10
2219/2220 [============================>.] - ETA: 0s - loss: 0.0730 - acc
uracy: 0.9726
Epoch 00007: val_accuracy did not improve from 0.91757
2220/2220 [==============================] - 22s 10ms/step - loss: 0.0730
- accuracy: 0.9726 - val_loss: 0.3535 - val_accuracy: 0.9163 - lr: 2.0000
e-04
Epoch 8/10
2218/2220 [============================>.] - ETA: 0s - loss: 0.0689 - acc
uracy: 0.9736
Epoch 00008: val_accuracy did not improve from 0.91757
2220/2220 [==============================] - 22s 10ms/step - loss: 0.0689
- accuracy: 0.9736 - val_loss: 0.4110 - val_accuracy: 0.9138 - lr: 2.0000
e-04
```

```
In [18]: import seaborn as sns
```

```
In [20]: q = len(history.history['accuracy'])

         plt.figsize=(10,10)
         sns.lineplot(x = range(1,1+q),y = history.history['accuracy'], label='Accur
         sns.lineplot(x = range(1,1+q),y = history.history['val_accuracy'], label='V
         plt.xlabel('epochs')
         plt.ylabel('Accuracy')
```

```
Out[20]: Text(0, 0.5, 'Accuracy')
```



```
In [ ]:
```