```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
import sys
import seaborn as sns
import numpy as np
np.set_printoptions(threshold=sys.maxsize)
import pandas as pd
import os
import h5py
import PIL
import cv2
import tensorflow as tf
import tensorflow.keras as keras
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import ReduceLROnPlateau
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.models import load_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pylab as plt
from matplotlib import cm
%matplotlib inline
import matplotlib.pylab as pylab
pylab.rcParams["figure.figsize"] = (16,4)
```

```
input_folder = '/content/drive/MyDrive/ru_letters/all_letters_image/'
all_letters_filename = os.listdir(input_folder)
len(all_letters_filename)
```

```
14190
```

```
i = Image.open("/content/drive/MyDrive/ru_letters/all_letters_image/02_168.png")
i
```

```
i_arr = np.array(i)
i_arr
```

```
array([[[150, 147, 151, 255],
        [150, 147, 151, 255],
        [150, 148, 151, 255],
        [150, 147, 150, 255],
        [151, 146, 150, 255],
        [152, 146, 150, 255],
        [150, 145, 149, 255],
        [150, 145, 149, 255],
        [151, 146, 150, 255],
        [150, 145, 149, 255],
        [150, 144, 148, 255],
        [149, 144, 148, 255],
        [149, 143, 147, 255],
        [148, 143, 147, 255],
        [148, 143, 147, 255],
        [149, 143, 147, 255],
        [147, 143, 147, 255],
        [145, 144, 147, 255],
        [145, 144, 146, 255],
        [145, 143, 146, 255],
        [145, 144, 147, 255],
        [147, 143, 146, 255],
        [147, 143, 146, 255],
        [147, 142, 146, 255],
        [148, 143, 147, 255],
        [148, 143, 147, 255],
        [149, 143, 147, 255],
        [148, 142, 146, 255],
        [146, 141, 145, 255],
        [147, 141, 145, 255],
        [146, 141, 145, 255],
        [146, 141, 145, 255]],

       [[150, 145, 149, 255],
        [151, 147, 151, 255],
        [151, 146, 149, 255],
        [150, 145, 149, 255],
        [151, 146, 150, 255],
        [151, 146, 150, 255],
        [150, 145, 149, 255],
        [149, 144, 148, 255],
        [150, 145, 149, 255],
        [150, 145, 149, 255],
        [150, 144, 148, 255],
        [149, 144, 148, 255],
        [149, 143, 147, 255],
        [149, 144, 147, 255],
        [148, 143, 147, 255],
        [149, 144, 148, 255],
        [149, 145, 148, 255],
        [147, 145, 148, 255],
        [146, 143, 146, 255],
        [146, 144, 147, 255],
        [146, 144, 147, 255],
        [146, 145, 147, 255],
        [146, 144, 147, 255],
        [148, 143, 147, 255],
        [149, 144, 148, 255],
```

```
def img_to_array(img_name, input_folder):
    img = image.load_img(input_folder + img_name, target_size=(32,32))
    x = image.img_to_array(img)
    return np.expand_dims(x, axis=0)
def data_to_tensor(img_names, input_folder):
    list_of_tensors = [img_to_array(img_name, input_folder) for img_name in img_names]
    return np.vstack(list_of_tensors)
```

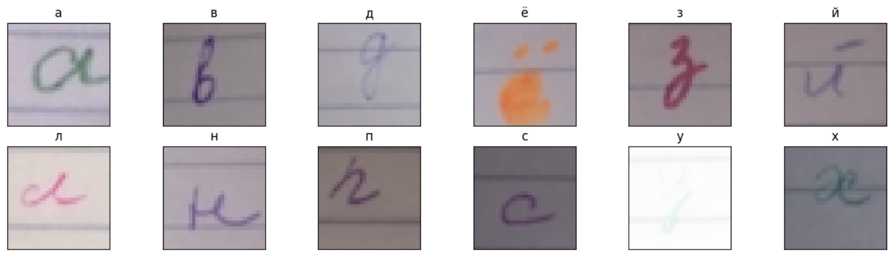
```
data = pd.read_csv("/content/drive/MyDrive/ru_letters/all_letters_info.csv")
image_names = data['file']
letters = data[ 'letter']
backgrounds = data['background'].values
targets = data['label'].values
tensors = data_to_tensor(image_names, input_folder)
tensors[0]
```

```
array([[[196., 186., 203.],
        [196., 188., 203.],
        [194., 187., 202.],
        [194., 186., 202.],
        [196., 188., 204.],
        [196., 188., 203.],
        [194., 186., 202.],
        [195., 187., 203.],
        [193., 186., 201.],
        [196., 188., 203.],
        [194., 187., 202.],
        [194., 185., 200.],
        [196., 186., 201.],
        [195., 186., 201.],
        [196., 187., 202.],
        [196., 187., 202.],
        [196., 188., 200.],
        [196., 188., 200.],
        [195., 187., 199.],
        [191., 184., 198.],
        [191., 183., 199.],
        [193., 186., 201.],
        [195., 186., 199.],
        [190., 182., 194.],
        [189., 183., 194.],
        [193., 186., 199.],
        [192., 184., 200.],
        [193., 185., 200.],
        [191., 183., 199.],
        [191., 182., 198.],
        [193., 185., 200.],
        [193., 186., 201.]],

       [[192., 184., 201.],
        [193., 186., 203.],
        [192., 188., 204.],
        [190., 185., 202.],
        [190., 186., 202.],
        [190., 186., 202.],
        [188., 184., 200.],
        [187., 183., 200.],
        [187., 182., 199.],
        [187., 182., 200.],
        [189., 185., 202.],
        [192., 183., 198.],
        [192., 182., 198.],
        [191., 183., 198.],
        [191., 182., 197.],
        [193., 183., 198.],
        [192., 183., 198.],
        [192., 183., 198.],
        [192., 183., 198.],
        [187., 181., 197.],
        [185., 182., 198.],
        [188., 185., 201.],
        [191., 184., 199.],
        [189., 180., 195.],
        [187., 183., 196.],
```

```
print ('Tensor shape:', tensors.shape)
print ('Target shape', targets.shape)
```

```
Tensor shape: (14190, 32, 32, 3)
Target shape (14190,)
```

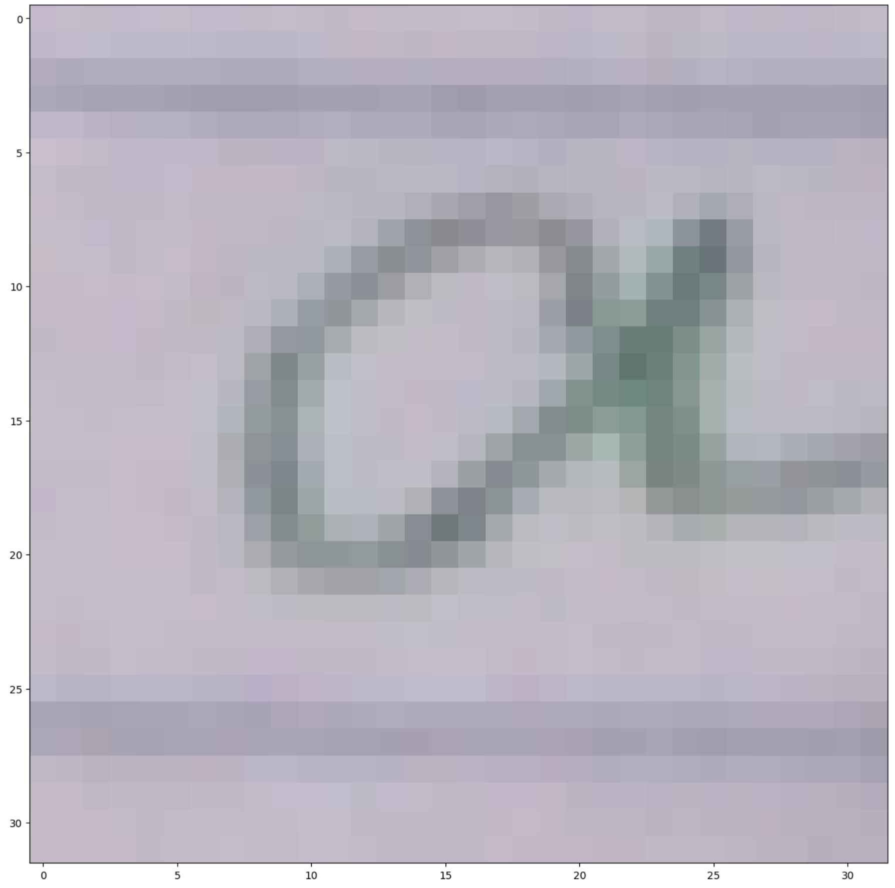
```
def display_images(img_path, ax):
    img = cv2.imread(input_folder + img_path)
    ax.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
fig = plt.figure(figsize=(16, 4))
for i in range(12):
    ax = fig.add_subplot(2, 6, i + 1, xticks=[], yticks=[], title=letters[i*100])
    display_images(image_names[i*100], ax)

```



```
g = sns.countplot(targets)
```

```
X = tensors.astype("float32")/255
```

```
arr = X[0]
arr_ = np.squeeze(arr)
plt.imshow(arr_)
plt.show()
```



```
targets[0]
```

```
1
```

```
y = targets
img_rows=img_cols = 32,
num_classes = 33
y = keras.utils.to_categorical(y-1, num_classes)
```

```
print(X.shape)
print(y.shape)
```

```
(14190, 32, 32, 3)
(14190, 33)
```

```
def captch_ex(file_name):
    img = cv2.imread(file_name)
    img_final = cv2.imread(file_name)
    img2gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, mask = cv2.threshold(img2gray, 100, 255, cv2.THRESH_BINARY)
    image_final = cv2.bitwise_and(img2gray, img2gray, mask=mask)
    ret, new_img = cv2.threshold(image_final, 100, 255, cv2.THRESH_BINARY)
    kernel = cv2.getStructuringElement(cv2.MORPH_CROSS, (3,
                                                         3))
    dilated = cv2.dilate(new_img, kernel, iterations=9)
    contours, hierarchy = cv2.findContours(new_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
    for contour in contours:
        [x, y, w, h] = cv2.boundingRect(contour)
        if w < 35 and h < 35:
            continue
            cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 255), 2)
    cv2.imshow('captcha_result', img)
    cv2.waitKey()


```

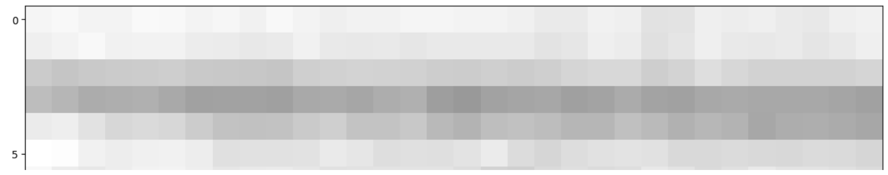
```
file_name = '/content/drive/MyDrive/ru_letters/all_letters_image/03_168.png'
```

```
X_grey = np.dot(X[...,:3], [0.100, 0.50, 0.60])
print ('Grayscaled Tensor shape:', X_grey.shape)
```

```
Grayscaled Tensor shape: (14190, 32, 32)
```

```
plt.imshow(X_grey[0], cmap=plt.get_cmap("gray"))
```

```
<matplotlib.image.AxesImage at 0x7b7df3123ee0>
```



```
X_train_whole, X_test, y_train_whole, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train_whole, y_train_whole, test_size=0.1, random_state=1)
```

```
deep_RU_model = Sequential()
deep_RU_model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same', activation ='relu'))
deep_RU_model.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                 activation ='relu'))
deep_RU_model.add(MaxPooling2D(pool_size=(2,2)))
deep_RU_model.add(Dropout(0.25))
deep_RU_model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
deep_RU_model.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                 activation ='relu'))
deep_RU_model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
deep_RU_model.add(Dropout(0.25))
deep_RU_model.add(Flatten())
deep_RU_model.add(Dense(256, activation = "relu"))
deep_RU_model.add(Dropout(0.5))
deep_RU_model.add(Dense(33, activation = "softmax"))
```

```
optimizer = tf.keras.optimizers.legacy.RMSprop(learning_rate=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
```

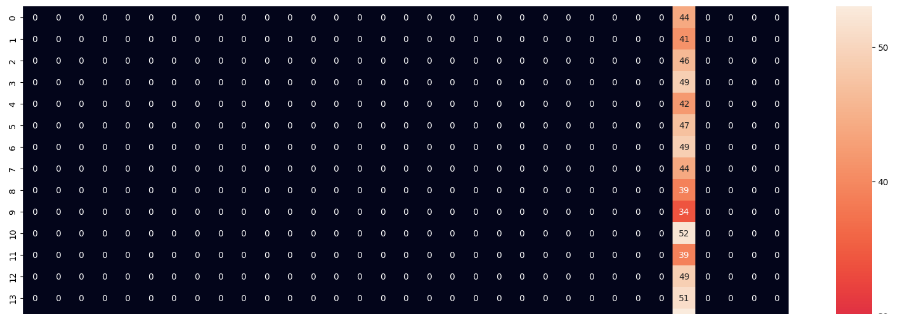
```
deep_RU_model.compile(loss="categorical_crossentropy", optimizer = optimizer,metrics=["accuracy"])
```

```
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=3,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
```

```
es = EarlyStopping(monitor='val_accuracy', mode='max', verbose=1, patience=50)
mc = ModelCheckpoint('best_model.h5', monitor='val_accuracy', mode='max', verbose=1, save_best_only=True)
```

```
y_pred = deep_RU_model.predict(X_test)
y_pred = np.argmax(y_pred, axis=1)
y_test = np.argmax(y_test, axis=1)
confusion_mtx = confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_mtx, annot=True, fmt='d')
```

```
45/45 [==============================] - 3s 59ms/step
<Axes: >
```



1