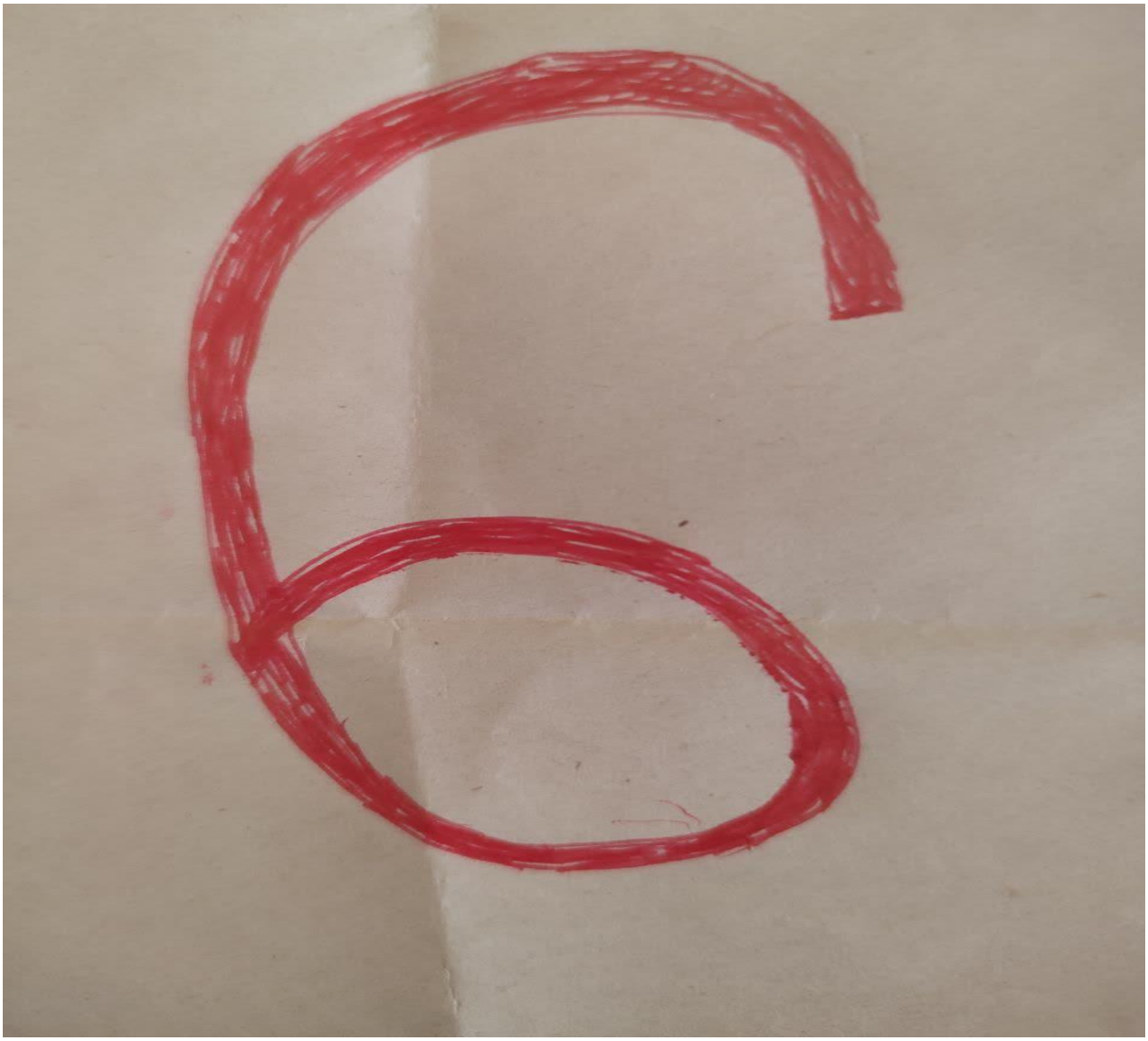
به نام خدا

برای پیش بینی یک عدد دست نویس ابتدا یک عدد روی برگه نوشته و از آن عکس گرفته و در repository در صفحه Github آن را ذخیره می‌کنیم.

عکس عدد ذخیره شده در Github



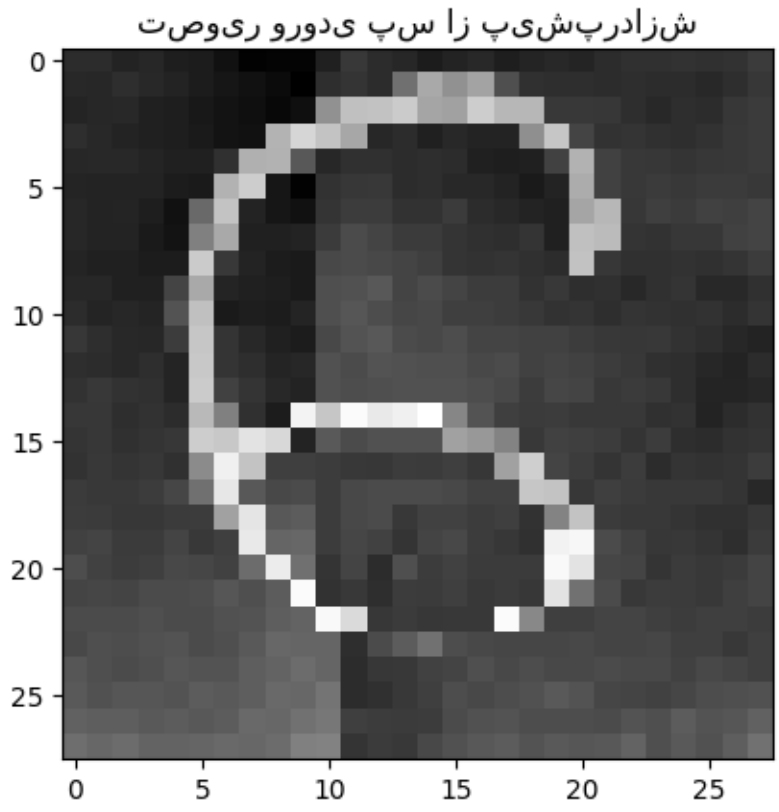
سپس با استفاده از دیتاست MNIST کدهای دستوری برای پیش بینی این عدد را در کولب اجرا می‌گیریم:

```
# Import کتابخانه‌های مورد نیاز
import cv۲
import numpy as np
import matplotlib.pyplot as plt
from keras.models import load_model
from keras.datasets import mnist
from keras.utils import to_categorical
import os

# ۱. بارگیری یا آموزش مدل
def load_or_train_model():
    # بررسی وجود فایل مدل
    if os.path.exists('model_mnist.h٥'):
        print("بارگیری مدل از فایل ذخیره شده...")
        model = load_model('model_mnist.h٥')
    else:
        print("آموزش مدل جدید...")
        # بارگیری داده‌های MNIST
        (X_train, y_train), (X_test, y_test) = mnist.load_data()

        # پیش‌پردازش داده‌ها
        X_train = X_train.reshape(-١, ٢٨, ٢٨, ١).astype('float٣٢') / ٢٥٥,٠
        X_test = X_test.reshape(-١, ٢٨, ٢٨, ١).astype('float٣٢') / ٢٥٥,٠
        y_train = to_categorical(y_train, ١٠)
        y_test = to_categorical(y_test, ١٠)

        # ساخت مدل CNN
        from keras.models import Sequential
        from keras.layers import Conv٢D, MaxPooling٢D, Flatten, Dense,
Dropout

        model = Sequential([
            Conv٢D(٣٢, (٣, ٣), activation='relu', input_shape=(٢٨, ٢٨,
١)),
            MaxPooling٢D((٢, ٢)),
            Conv٢D(٦٤, (٣, ٣), activation='relu'),
            MaxPooling٢D((٢, ٢)),
            Flatten(),
```

```
            Dense(۱۲۸, activation='relu'),
            Dropout(۰,۵),
            Dense(۱۰, activation='softmax')
        ])

        # کامپایل و آموزش مدل
        model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
        model.fit(X_train, y_train, epochs=۵, batch_size=۶۴,
validation_data=(X_test, y_test))

        # ذخیره مدل
        model.save('model_mnist.h۵')
        print("مدل آموزش داده و ذخیره شد.")

    return model

# ۲. تابع پیش‌پردازش تصویر ورودی
def preprocess_image(image_path):
    # خواندن تصویر به صورت grayscale
    img = cv۲.imread(image_path, cv۲.IMREAD_GRAYSCALE)

    if img is None:
        raise ValueError(f"نمی‌توان تصویر را از مسیر {image_path} بارگیری
کرد")

    # تغییر سایز به ۲۸x۲۸ پیکسل
    img = cv۲.resize(img, (۲۸, ۲۸))

    # معکوس کردن رنگ (اگر زمینه تصویر تیره است)
    img = cv۲.bitwise_not(img)

    # نرمال سازی به محدوده [۰, ۱]
    img = img.astype('float۳۲') / ۲۵۵,۰

    # تغییر شکل به (۱, ۲۸, ۲۸, ۱)
    img = np.expand_dims(img, axis=۰)  # بعد batch
    img = np.expand_dims(img, axis=-۱)  # بعد channel

    return img

# ۳. تابع اصلی برای تشخیص عدد
def recognize_digit(image_path):
    # بارگیری یا آموزش مدل
    model = load_or_train_model()
```

```
    try:
        # پیش‌پردازش تصویر
        processed_img = preprocess_image(image_path)

        # بررسی شکل تصویر پردازش شده
        print("شکل تصویر پردازش شده:", processed_img.shape)  # باید (۱, ۲۸, ۲۸, ۱) باشد

        # نمایش تصویر ورودی
        plt.imshow(processed_img[۰, :, :, ۰], cmap='gray')
        plt.title("تصویر ورودی پس از پیش‌پردازش")
        plt.show()

        # پیش‌بینی مدل
        prediction = model.predict(processed_img)
        predicted_number = np.argmax(prediction)
        confidence = np.max(prediction) * ۱۰۰

        print(f"\n نتیجه تشخیص: عدد {predicted_number} با اطمینان {confidence:.۲f}%")

        return predicted_number, confidence

    except Exception as e:
        print("خطا در پردازش تصویر:", str(e))
        return None, None

# ٤. استفاده از تابع
if __name__ == "__main__":
    # مسیر تصویر مورد نظر را اینجا قرار دهید
    image_path = "/content/CNN-MNIST/number.jpg"  # یا مسیر کامل مانند '/content/number.jpg'

    # تشخیص عدد
    digit, confidence = recognize_digit(image_path)

    if digit is not None:
        print(f"\nعدد تشخیص داده شده: {digit} با اطمینان {confidence:.۲f}%")
```

خروجی دستورات بالا :



نتیجه تشخیص: عدد ۶ با اطمینان ۳۱٫۹۸٪