

به نام خدا

موضوع: پیش‌بینی عدد دست‌نویس با استفاده از شبکه‌های عصبی پیچشی
(Convolutional Neural Networks)

کدهای پایتون این پروژه در صفحه **gitab** به آدرس زیر است:

<https://github.com/Esmaeili.git1/CNN-MNIST4.2>

برای پیش‌بینی عدد دست‌نویس با استفاده از شبکه‌های عصبی پیچشی مراحل زیر را انجام می‌دهیم:

فرآخوانی کتابخانه‌های مورد نیاز:

- **NumPy**: پیش‌پردازش داده‌ها و محاسبات
 - **Matplotlib**: نمایش تصاویر و نمودارها
 - **TensorFlow/Keras**
-
- بارگذاری داده‌های MNIST
 - ساخت مدل CNN
 - آموزش و ارزیابی مدل
 - ذخیره/بارگذاری مدل

بارگذاری داده‌ها:

- از مجموعه‌داده **MNIST** استفاده شده که شامل:
 - ۶۰,۰۰۰ تصویر آموزشی و ۱۰,۰۰۰ تصویر تست.

- هر تصویر 28×28 پیکسل و سیاه و سفید (تک کاناله) است.
 - اعداد ۰ تا ۹ (۱۰ کلاس مختلف).
-

پیش‌پردازش داده‌ها:

- تغییر شکل داده‌ها (**Reshaping**): تبدیل تصاویر 28×28 پیکسلی به شکل $(1, 28, 28, 28)$ برای سازگاری با $\text{Conv}\Gamma\text{D}$ لایه‌های.

- نرمال‌سازی (**Normalization**): مقادیر پیکسل‌ها از بازه $[1, 255]$ به $[0, 1]$ تبدیل می‌شوند.
-

ساخت مدل CNN

۱. لایه **Conv\Gamma\text{D}** اول:

- فیلتر 3×3
- ورودی: تصاویر 28×28 پیکسل (سیاه و سفید)
- فعال‌ساز: ReLU
- وظیفه: تشخیص لبه‌ها و الگوهای ساده

۲. لایه **MaxPooling\Gamma\text{D}** اول:

- پنجره 2×2
- وظیفه: کاهش ابعاد و حفظ مهم‌ترین ویژگی‌ها

۳. لایه **Conv\Gamma\text{D}** دوم:

- فیلتر 3×3
- فعال‌ساز: ReLU
- وظیفه: تشخیص الگوهای پیچیده‌تر

۴. لایه **MaxPooling\Gamma\text{D}** دوم:

- پنجره 2×2
- وظیفه: کاهش بیشتر ابعاد

۵. لایه **Flatten**:

- تبدیل ماتریس ۳بعدی به بردار ۱بعدی
- آماده‌سازی برای لایه‌های متراکم

۶. لایه متراکم (Dense):

- نورون ۱۲۸
 - فعال‌ساز: ReLU
 - وظیفه: یادگیری ارتباطات غیرخطی
۷. لایه خروجی:
- نورون (مطابق ارقام ۹-۰)
 - فعال‌ساز: Softmax
 - وظیفه: محاسبه احتمال تعلق به هر کلاس

کامپایل مدل:

- بهینه‌ساز Adam :**(Optimizer)**
 - تابع هزینه categorical crossentropy :**(loss)**
 - متریک accuracy :**(metric)**
-

آموزش مدل:

- مدل ۵ بار روی کل داده‌های آموزشی یادگیری می‌کند.
 - **epochs=۵**
 - در هر مرحله ۶۴ نمونه پردازش می‌شود.
 - **batch_size=۶۴**
 - ۲۰٪ داده آموزشی برای اعتبارسنجی جدا می‌شود.
 - **validation_split=۰.۲**
-

پیش‌بینی دو مدل تصادفی

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
```

```

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# ۱. بارگذاری و پیشپردازش داده‌ها
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()

# نرمال‌سازی و تغییر شکل داده‌ها
train_images = train_images.reshape(-1, 28, 28, 1).astype('float32') / 255
test_images = test_images.reshape(-1, 28, 28, 1).astype('float32') / 255

# ۲. تبدیل برچسب‌ها به فرم one-hot
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)

# ۳. ساخت مدل CNN
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# ۴. آموزش مدل
print("آموزش مدل...")
history = model.fit(train_images, train_labels,
                     epochs=5,
                     batch_size=128,
                     validation_split=0.2)

# ۵. ذخیره مدل
model.save('mnist_cnn_model.h5')
print("مدل با موفقیت ذخیره شد")

# ۶. پیش‌بینی روی دو نمونه تصادفی
def show_predictions(num_samples=2):
    indices = np.random.choice(len(test_images), num_samples,
                               replace=False)
    plt.figure(figsize=(10, 4))

```

```

        for i, idx in enumerate(indices):
            plt.subplot(1, num_samples, i+1)
            img = test_images[idx].reshape(28, 28)
            pred = model.predict(img[np.newaxis, ...])
            true_label = np.argmax(test_labels[idx])
            pred_label = np.argmax(pred)
            plt.imshow(img, cmap='gray')
            plt.title(f'پیش‌بینی: {pred_label}\nواقعی: {true_label}')
            plt.axis('off')
        plt.tight_layout()
        plt.show()

show_predictions()

```

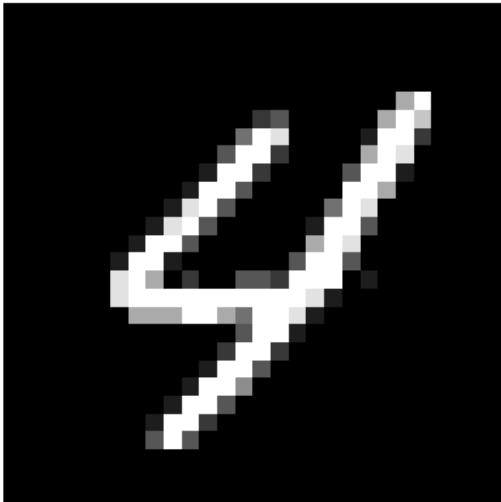
خروجی:

```

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434          1s  0us/step
/usr/local/lib/python3.11/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not
pass an `input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the model
instead.
    super().__init__(activity_regularizer=activity_regularizer, **kwargs)
در حال آموزش مدل...
Epoch 1/0
100/100          0.6s 18ms/step - accuracy: 0.8823 - 
loss: 0.3949 - val_accuracy: 0.9770 - val_loss: 0.013
Epoch 2/0
100/100          0.5s 59ms/step - accuracy: 0.9826 - 
loss: 0.0071 - val_accuracy: 0.9846 - val_loss: 0.0014
Epoch 3/0
100/100          0.4s 51ms/step - accuracy: 0.9884 - 
loss: 0.0272 - val_accuracy: 0.9871 - val_loss: 0.0016
Epoch 4/0
100/100          0.4s 58ms/step - accuracy: 0.9917 - 
loss: 0.0200 - val_accuracy: 0.9873 - val_loss: 0.0034
Epoch 5/0
100/100          0.4s 59ms/step - accuracy: 0.9942 - 
loss: 0.0183 - val_accuracy: 0.9893 - val_loss: 0.0074
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
مدل با موفقیت ذخیره شد.
1/1          1.92ms/step
1/1          2.1ms/step

```

4: یعنی بسیار
4: یعنی قاچاق



3: یعنی بسیار
3: یعنی قاچاق



رسم ماتریس آشتفتگی:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from sklearn.metrics import confusion_matrix

# ۱. بارگذاری و پیشپردازش داده‌ها
_, _ = mnist.load_data()
test_images = test_images.reshape(-1, 28, 28, 1).astype('float32') / 255
test_labels_true = np.argmax(tf.keras.utils.to_categorical(test_labels, 10), axis=1)

# ۲. بارگذاری مدل (یا آموزش جدید)
try:
    model = load_model('mnist_cnn_model.h5')
    print("مدل از فایل بارگذاری شد")
except:
    print("...مدل یافت نشد! در حال آموزش مدل جدید")
    (train_images, train_labels) = mnist.load_data() [::]
    train_images = train_images.reshape(-1, 28, 28, 1).astype('float32') / 255
    train_labels = tf.keras.utils.to_categorical(train_labels, 10)
```

```

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=64)
model.save('mnist_cnn_model.h5')

# ۳. پیش‌بینی روی داده‌های تست
test_pred = np.argmax(model.predict(test_images), axis=1)

# ۴. محاسبه ماتریس آشتفتگی
cm = confusion_matrix(test_labels_true, test_pred)

# ۵. رسم ماتریس آشتفتگی
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=range(10),
            yticklabels=range(10))
plt.xlabel('پیش‌بینی مدل')
plt.ylabel('مقادیر واقعی')
plt.title('ماتریس آشتفتگی (MNIST CNN Model)')
plt.show()

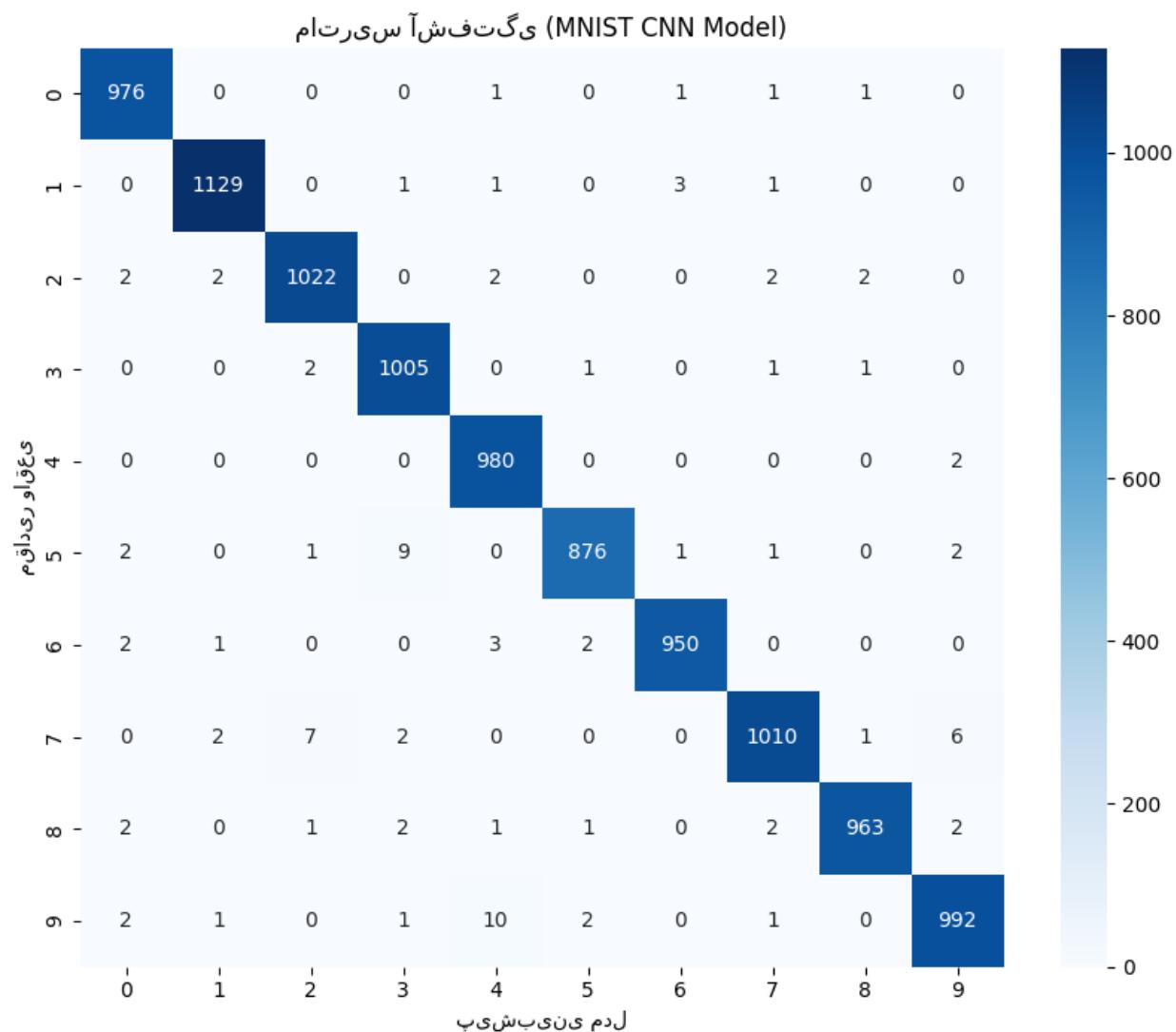
```

خروجی:

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.

مدل از فایل بارگذاری شد.

۳۱۳/۳۱۳ ————— ۴s ۱۱ms/step



تفسیر ماتریس آشفتگی (Confusion Matrix) برای مدل تشخیص اعداد MNIST:

۱. ساختار ماتریس:

محور افقی: (x) پیش‌بینی‌های مدل (۰ تا ۹)

محور عمودی: (y) مقادیر واقعی (۰ تا ۹)

هر سلول: تعداد/درصد نمونه‌هایی که مدل آنها را به یک کلاس خاص نسبت داده است.

۲. مقادیر مطلوب:

مقادیر روی قطر اصلی

- نشان دهنده پیش‌بینی‌های صحیح هستند.
- مثال: سلول (۳، ۳) تعداد دفعاتی که عدد ۳ به درستی تشخیص داده شده را نشان می‌دهد.

۳. مقادیر مشکل‌ساز:

- مقادیر خارج از قطر اصلی
- نشان دهنده خطاهای مدل هستند.
- مثال: سلول (۸، ۵) تعداد دفعاتی که عدد ۵ به اشتباه به عنوان ۸ پیش‌بینی شده را نشان می‌دهد.

۴. نمونه‌های کاربردی از تفسیر:

- اگر سلول (۱، ۷) مقدار بالایی دارد:
 - مدل در تشخیص تفاوت بین ۷ و ۱ مشکل دارد.
 - احتمالاً به دلیل شباهت ظاهری (مثلاً عدد ۷ با خط مورب کوتاه).
- اگر سطر مربوط به عدد ۹ مقادیر پراکنده‌ای دارد:
 - مدل در تشخیص عدد ۹ ضعیف عمل می‌کند.
 - ممکن است به دلیل تنوع در نحوه نوشتگی عدد ۹ باشد.

پیش‌بینی یک عدد دستنویس:

تصویر یک عدد دستنویس در آدرس ریپوی:

<https://github.com/Esmaeili4.3/CNN-MNIST1.git>

ذخیره شده را روی تصویر پیش‌پردازش می‌کنیم:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from PIL import Image

# دانلود ریپوی عمومی شما
!git clone https://github.com/Esmaeili4.3/CNN-MNIST1.git
```

```

# ۲۰. بارگذاری مدل
try:
    model = keras.models.load_model('/content/CNN-
MNIST/mnist_cnn_model.h5')
    print("☑ مدل با موفقیت بارگذاری شد")
except Exception as e:
    print("☒ خطای بارگذاری مدل", e)
    print("لیست فایل‌های موجود در ریپو")
    !ls -la /content/CNN-MNIST/


# ۲۱. تابع پیش‌پردازش تصویر
def preprocess_image(image_path):
    try:
        img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
        if img is None:
            raise ValueError("تصویر بارگذاری نشد! مسیر را بررسی کنید")

        img = 255 - img # تبدیل به فرمت MNIST (زمینه سیاه)
        img = cv2.resize(img, (28, 28))
        img = img.astype('float32') / 255
        return img.reshape(1, 28, 28, 1)
    except Exception as e:
        print("☒ خطای پیش‌پردازش", e)
        return None


# ۴۰. مسیر تصویر
image_path = '/content/CNN-MNIST/0.jpg'

# ۵۰. نمایش و پیش‌بینی
try:
    # نمایش تصویر
    print(f"\n☒ نمایش تصویر از مسیر {image_path}")
    img = Image.open(image_path)
    plt.imshow(img, cmap='gray')
    plt.axis('off')
    plt.title('تصویر ورودی')
    plt.show()

    # پیش‌بینی
    processed_img = preprocess_image(image_path)
    if processed_img is not None:
        prediction = model.predict(processed_img)
        predicted_num = np.argmax(prediction)
        confidence = np.max(prediction) * 100

```

```

print(f"\n{12}":نتیجه پیشینی)
print(f"- عدد تشخیص داده شده: {predicted_num}")
print(f"- میزان اطمینان:confidence:.2f}%)")

#نمایش احتمال‌های تمام کلاس‌ها
print("\n{13} ۹۰۰":احتمال‌های تمام اعداد)
for i, prob in enumerate(prediction[1]):
    print(f" عدد {i}: {prob*100:.2f}%)"

except Exception as e:
    print("\n{14} خطأ:", e)
print("برای عیبیابی، محتوای ریپو را بررسی کنید\n{15}!")
!ls -la /content/CNN-MNIST/

```

خروجی مدل تصویر اصلی:

```

Cloning into 'CNN-MNIST'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (8/8), 22,82 KiB | 4,19 MiB/s, done.
Resolving deltas: 100% (1/1), done.
{16} خطأ در بارگذاری مدل [Errno 2] Unable to synchronously open file (unable
to open file: name = '/content/CNN-MNIST/mnist_cnn_model.h5', errno = 2,
error message = 'No such file or directory', flags = 0, o_flags = 0)

```

لیست فایل‌های موجود در ریپو:

```

total 6.
drwxr-xr-x 3 root root 4096 Jul 22 12:00 .
drwxr-xr-x 1 root root 4096 Jul 22 12:00 ..
-rw-r--r-- 1 root root 29662 Jul 22 12:00 0.jpg
drwxr-xr-x 8 root root 4096 Jul 22 12:00 .git
-rw-r--r-- 1 root root 4688 Jul 22 12:00 .gitignore
-rw-r--r-- 1 root root 1060 Jul 22 12:00 LICENSE
-rw-r--r-- 1 root root 12 Jul 22 12:00 README.md

```

نمایش تصویر از مسیر /content/CNN-MNIST/0.jpg

۵ دوره ریوصت



۱/۱ ————— ۰.۶۶ms/step

- نتیجه پیش‌بینی** ۱۲
۳۴
- عدد تشخیص داده شده: ۵
- میزان اطمینان: ۵۹, ۵۴%

احتمال‌های تمام اعداد:

- ۰, ۰۱ : ۰٪
- ۰, ۴۳ : ۱٪
- ۰, ۱۹ : ۲٪
- ۳۵, ۹۶ : ۳٪
- ۰, ۰۴ : ۴٪
- ۵۹, ۵۴ : ۵٪
- ۰, ۹۱ : ۶٪
- ۱, ۶۲ : ۷٪
- ۰, ۲۷ : ۸٪
- ۱, ۰۴ : ۹٪

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras

# تابع پیش‌پردازش تصویر
def preprocess_image(image_path):
```

```

خواندن تصویر به صورت خاکستری #
img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

معکوس کردن رنگها (تبديل به زمینه سیاه) #
img = 255 - img

پیکسل ۲۸×۲۸ تغییر اندازه به ۲۸ #
img = cv2.resize(img, (28, 28))

نرمال سازی #
img = img.astype('float32') / 255

تغییر شکل برای ورودی مدل #
return img.reshape(1, 28, 28, 1)

# مسیر تصویر شما - تغییر دهید
image_path = '/content/CNN-MNIST/0.jpg' # مسیر تصویر خود را وارد کنید

# پیش پردازش تصویر
processed_img = preprocess_image(image_path)

# نمایش تصویر پردازش شده
print("تصویر پس از پردازش (آماده برای مدل)") #:تصویر پس از پردازش (آماده برای مدل")
plt.imshow(processed_img.reshape(28, 28), cmap='gray')
plt.axis('off')
plt.show()

# پیش بینی با مدل
prediction = model.predict(processed_img)
predicted_number = np.argmax(prediction)
confidence = np.max(prediction) * 100

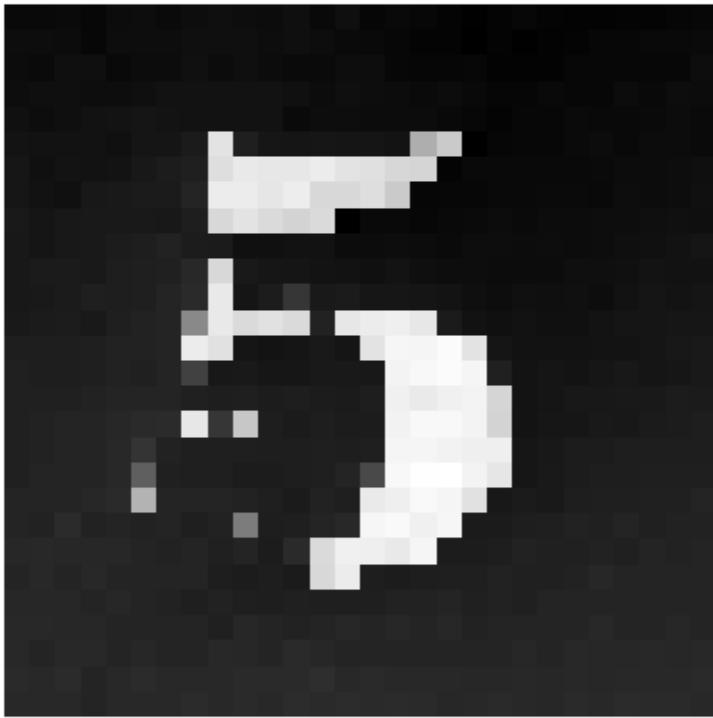
# نمایش نتیجه
print(f"\n{predicted_number} مدل پیش بینی می کند این عدد {confidence:.2f}%") #:میزان اطمینان برای تمام اعداد
аст با اطمینان

# نمایش احتمال های تمام کلاس ها
print("\n---") #:میزان اطمینان برای تمام اعداد
for i, prob in enumerate(prediction[0]):
    print(f"کلاس {i}: {prob*100:.2f}%") #:کلاس ۰: ۹۷.۷۷٪

```

خروجی:

تصویر پس از پردازش (آماده برای مدل):



۱/۱ ۰s ۵۶ms/step

۵۹,۵۴٪ مدل پیش‌بینی می‌کند این عدد ۵ است با اطمینان

میزان اطمینان برای تمام اعداد ۹۰٪:
۰,۰۱٪ عدد ۰
۰,۴۳٪ عدد ۱
۰,۱۹٪ عدد ۲
۳۵,۹۶٪ عدد ۳
۰,۰۴٪ عدد ۴
۵۹,۵۴٪ عدد ۵
۰,۹۱٪ عدد ۶
۱,۶۲٪ عدد ۷
۰,۲۷٪ عدد ۸
۱,۰۴٪ عدد ۹

تفسیر نتایج پیش‌بینی مدل:

۱. پیش‌بینی اصلی:

- ۵ عدد تشخیص داده شده:
- ۵۹,۵۴٪ میزان اطمینان:

- این نشان می‌دهد مدل با اعتماد متوسطی عدد ۵ را پیش‌بینی کرده است (حداقل ۸۰٪ معمولاً اطمینان بالایی محسوب می‌شود).

۲. توزیع احتمالات:

- رقم دوم محتمل : عدد ۳ با ۳۵.۹۶٪.
- این نشان می‌دهد تصویر ورودی ممکن است ویژگی‌های مشترکی با عدد ۳ داشته باشد (مثلاً شیاهت در انحناها).
- سایر ارقام : احتمال ناچیز (محدوده ۱٪) برای سایر اعداد.