پیش بینی یک عدد دست نویس با مدل CNN

کدهای پایتون مربوط به این پروژه در سایت github به آدرس زیر است:

https://github.com/E-Khoshbakht98/CNN2/tree/main

برای پیش بینی یک عدد دست نویس از دیتاست mnist استفاده می‌کنیم. برای این کار مطابق مراحل زیر اقدام می‌کنیم:

فراخوانی کتابخانه‌های مورد نیاز:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.datasets import mnist
import cv2
from google.colab.patches import cv2_imshow
import pandas as pd
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

بارگذاری داده‌های mnist و پیش پردازش آنها:

```
(train_images, train_labels), (test_images, test_labels) =
mnist.load_data()
# پیش‌پردازش داده ها
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images.astype('float32') / 255
 # تبدیل برچسب‌ها به فرم one-hot
train_labels = keras.utils.to_categorical(train_labels)
test_labels = keras.utils.to_categorical(test_labels)
```

داده‌های MNIST شامل ۶۰۰۰۰ تصویر آموزشی و ۱۰۰۰۰ تصویر تست است. تصاویر ۲۸*۲۸پیکسل و سیاه‌وسفید هستند. در مرحله پیش پردازش، مقادیر پیکسل‌ها به بازه[0,1] استاندارد می‌شوند و داده‌ها را برای سازگاری با ورودی مدل تغییر شکل می‌دهیم و برچسب‌ها را به فرم one-hot encoding تبدیل می‌کنیم.

برازش مدل CNN :

```
# ساخت مدل CNN
model = keras.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28,
1)),
```

```
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

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

# آموزش مدل
history = model.fit(train_images, train_labels,
                    epochs=5,
                    batch_size=64,
                    validation_split=0.2)
```

مدل از ۳ لایه کانولوشنی با فعال‌ساز ReLU استفاده می‌کند. پس از هر دو لایه کانولوشنی یک لایهMaxPooling برای کاهش ابعاد و استخراج ویژگی‌های اصلی قرار دارد. در انتها دو لایه تمام‌متصل (Dense) وجود دارد.

در کامپایل مدل، از بهینه ساز adam استفاده می‌شود و تابع هزینه آن، آنتروپی دسته‌ای است که مناسب برای مسائل چند کلاسه می باشد.

مدل با ۵ دوره و اندازه دسته ۱۲۸ آموزش می بیند.

خروجی:

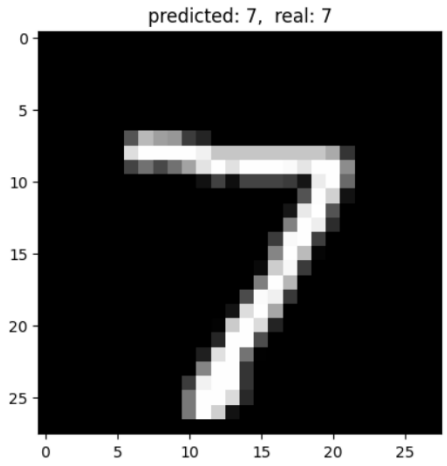
```
Epoch 1/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 32s 39ms/step - accuracy: 0.8601 - loss:
0.4794 - val_accuracy: 0.9762 - val_loss: 0.0780
Epoch 2/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 27s 36ms/step - accuracy: 0.9811 - loss:
0.0616 - val_accuracy: 0.9852 - val_loss: 0.0515
Epoch 3/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 44s 40ms/step - accuracy: 0.9866 - loss:
0.0409 - val_accuracy: 0.9875 - val_loss: 0.0449
Epoch 4/5
750/750 ━━━━━━━━━━━━━━━━━━━━ 26s 34ms/step - accuracy: 0.9905 - loss:
0.0302 - val_accuracy: 0.9886 - val_loss: 0.0391
Epoch 5/5
  750/750 ━━━━━━━━━━━━━━━━━━━━ 27s 36ms/step - accuracy: 0.9931 - loss:
                        0.0216 - val_accuracy: 0.9882 - val_loss: 0.0391
```

ارزیابی مدل:

```
# ارزیابی مدل روی داده تست
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(f'دقت مدل روی داده تست: {test_acc}')

# ذخیره مدل
model.save('mnist_cnn_model.h5')

# نمایش نمونه‌ای از پیش‌بینی‌ها
predictions = model.predict(test_images)

# نمایش یک نمونه تصویر و پیش‌بینی آن
def plot_sample_prediction(index):
    plt.imshow(test_images[index].reshape(28, 28), cmap='gray')
    pred = np.argmax(predictions[index])
    true = np.argmax(test_labels[index])
    plt.title(f'predicted: {pred},  real: {true}')
    plt.show()

plot_sample_prediction(0)  # نمایش اولین تصویر در مجموعه تست
```

خروجی:

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9863 - loss: 0.0413
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')`.
دقت مدل روی داده تست: ۰٫۹۸۹۸۹۹۹۹۲۹۴۲۸۱۰۱
313/313 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step
```



برای ارزیابی مدل ابتدا مقدار دقت مدل روی داده های (تصویرهای) تست را به دست می آوریم که مقدار آن حدودا ۹۹ درصد است. بنابراین مدل عملکرد بسیار خوبی دارد و می تواند اعداد دست نویس را با دقت بالای ۹۸ درصد تشخیص دهد.

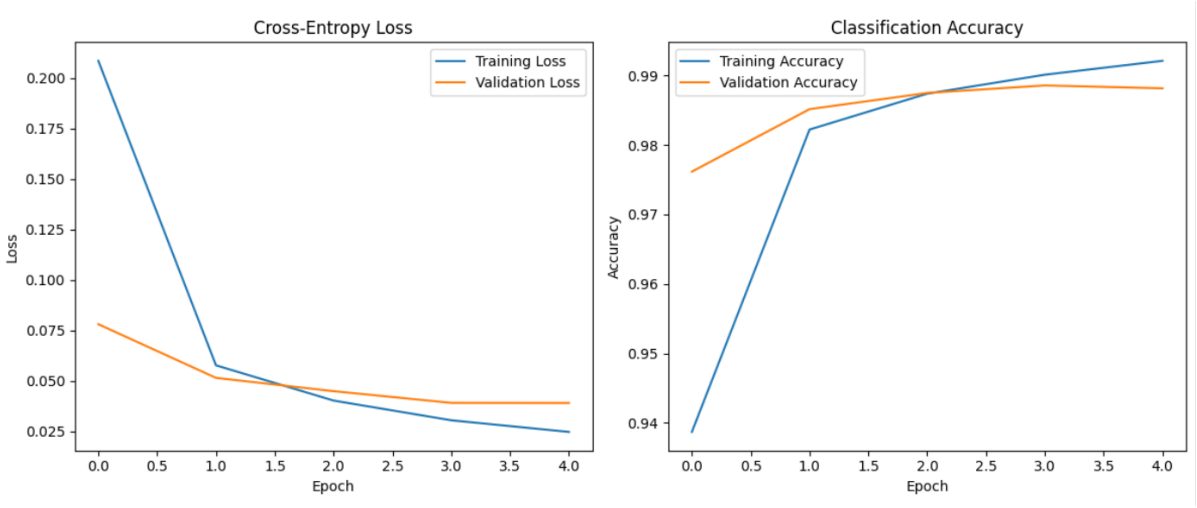
سپس اولین تصویر مجموعه تست را به عنوان نمونه نمایش می دهیم تا به طور شهودی پیش بینی مدل را با مقدار واقعی مقایسه کنیم که در اینجا تصویر عدد ۷ را نمایش می دهد که توسط مدل هم پیش بینی شده است.

همچنین در خروجی نشان داده شده است که ارزیابی مدل روی ۳۱۳ دسته انجام شده است و زمان پردازش حدود ۶ ثانیه بوده است.

رسم نمودار دقت و خطا:

```
# تابع رسم نمودارها
def plot_training_history(history):
    plt.figure(figsize=(12, 5))

    # نمودار Loss
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Cross-Entropy Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

    # نمودار Accuracy
    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation
Accuracy')
    plt.title('Classification Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.tight_layout()
    plt.show()

# فراخوانی تابع برای رسم نمودارها
plot_training_history(history)
```

خروجی:



تفسیر نمودار اول) Cross-Entropy Loss :خطای آنتروپی متقاطع

ویژگی‌های مشاهده شده:

- **خط آموزش (Training Loss)**: از حدود ۰.۲ شروع شده و به تدریج به زیر ۰.۰۲۵ کاهش یافته است.
- **خط اعتبارسنجی (Validation Loss)**: از حدود ۰.۱۵۰ شروع شده و به حدود ۰.۰۵۰ رسیده است.
- **روند کلی**: هر دو منحنی به صورت پایدار در حال کاهش هستند.

تحلیل:

۱. **همگرایی مناسب**: هر دو منحنی به سمت پایین حرکت می‌کنند که نشان‌دهنده یادگیری صحیح مدل است

۲. **فاصله بین منحنی‌ها**: فاصله نسبتاً کم (حدود ۰.۰۲۵ در پایان آموزش) نشان می‌دهد مدل دچار بیش‌برازش (overfitting) نشده است.

۳. **پایداری**: کاهش خطا بدون نوسانات شدید، نشان‌دهنده انتخاب مناسب نرخ یادگیری است.

تفسیر نمودار دوم) Classification Accuracy :دقت طبقه‌بندی

ویژگی‌های مشاهده شده:

- **خط آموزش (Training Accuracy)**: از حدود ۰.۹۵ شروع شده و به بیش از ۰.۹۹ رسیده است
- **خط اعتبارسنجی (Validation Accuracy)**: از حدود ۰.۹۶ شروع شده و به حدود ۰.۹۸۵ رسیده است
- **روند کلی**: هر دو منحنی به صورت یکنواخت در حال افزایش هستند

تحلیل:

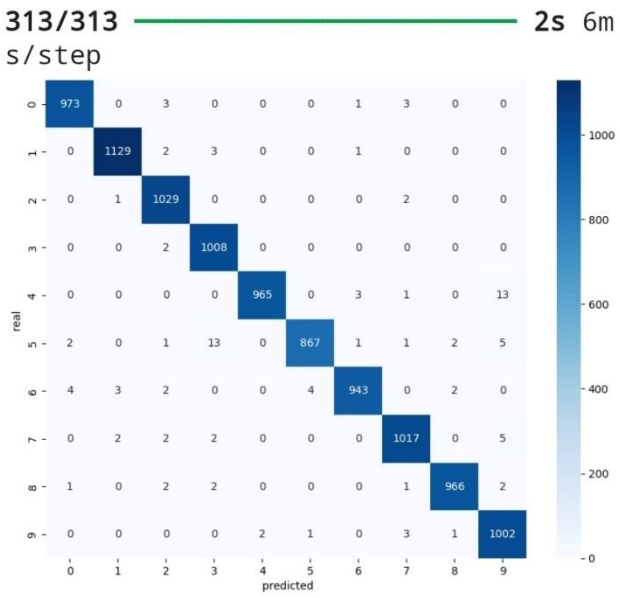
۱. **دقت نهایی عالی**: دقت اعتبارسنجی نزدیک به ۹۸.۵٪ نشان‌دهنده عملکرد بسیار خوب مدل است

۲. **هماهنگی بین آموزش و اعتبارسنجی** :فاصله کم بین دو منحنی (حدود ۰.۵٪) تأیید می‌کند مدل به خوبی تعمیم یافته است

۳. **سرعت یادگیری** :مدل در همان دوره‌های اولیه به دقت بالا دست یافته که نشان‌دهنده معماری مناسب است.

رسم ماتریس آشفتگی:

```
# بارگیری داده های تست
(_, _), (test_images, test_labels) = mnist.load_data()
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float32')
/ 255

# پیش‌بینی روی داده های تست
y_pred = model.predict(test_images)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = test_labels

# محاسبه ماتریس درهم‌ریختگی
cm = confusion_matrix(y_true, y_pred_classes)

# رسم ماتریس
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=range(10), yticklabels=range(10))
plt.xlabel(' predicted')
plt.ylabel('real ')
plt.show()
```



تفسیر خروجی: ساختار ماتریس درهم‌ریختگی به صورت زیر است:

- **ستون‌ها** :پیش‌بینی‌های مدل (اعداد ۰ تا ۹)
- **سطرها** :مقادیر واقعی (اعداد ۰ تا ۹)
- **مقادیر روی قطر اصلی** :پیش‌بینی‌های صحیح
- **مقادیر خارج از قطر اصلی** :خطاهای طبقه‌بندی

به طور کلی اعداد به درستی پیش بینی شده اند و عدد ۱ بهترین عملکرد را داشته است. اما خطاهای قابل توجه عبارت اند از:

عدد ۵: ۱۳ مورد از عدد ۵ به اشتباه به عنوان عدد ۳ طبقه‌بندی شده‌اند.

عدد ۹: ۵ مورد به اشتباه به عنوان عدد ۷ طبقه‌بندی شده‌اند.

عدد ۶: ۵ مورد به اشتباه به عنوان عدد ۰ طبقه‌بندی شده‌اند.

پیش بینی یک عدد دست نویس:

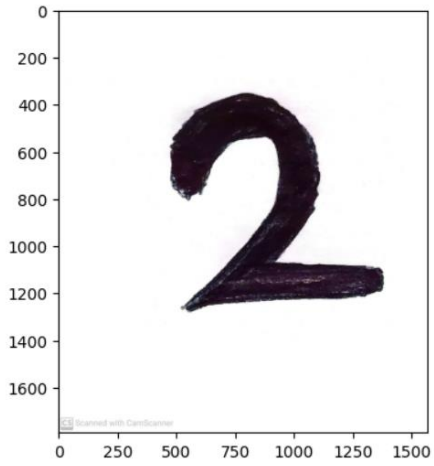
در این مرحله تصویر یک عدد دست نویس شخصی را که در یک repository در github ذخیره شده است، فراخوانی کرده و به مدل می دهیم تا عدد نوشته شده را پیش بینی کند. البته قبل از آن مرحله پیش پردازش را روی عکس ورودی اعمال می کنیم تا به فرم تصویرهای mnist باشد.

فراخوانی:

```
!git clone https://github.com/E-Khoshbakht98/CNN2.git

model = keras.models.load_model('mnist_cnn_model.h5')

def preprocess_image(image_path):

    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img = 255 - img
    img = cv2.resize(img, (28, 28))
    img = img.astype('float32') / 255
    img = img.reshape(1, 28, 28, 1)

    return img

original_img = cv2.imread('number2.jpg', cv2.IMREAD_GRAYSCALE)

from PIL import Image
import matplotlib.pyplot as plt

img = Image.open('/content/CNN2/number2.jpg')
```

```
plt.imshow(img)
plt.show()
```

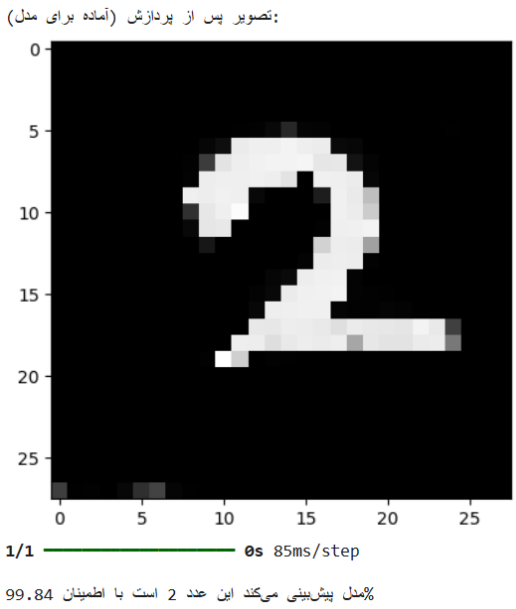
خروجی: (تصویر اصلی)



پیش پردازش:

```
processed_img = preprocess_image('/content/CNN2/number2.jpg')
print("\n(آماده برای مدل) تصویر پس از پردازش:")
plt.imshow(processed_img.reshape(28, 28), cmap='gray')
plt.show()


prediction = model.predict(processed_img)
predicted_number = np.argmax(prediction)

print(f"\n مدل پیش‌بینی می‌کند این عدد {predicted_number} است با اطمینان
{np.max(prediction)*100:.2f}%")
```

خروجی:



ابعاد تصویر اصلی به ۲۸ *۲۸ تغییر سایز داده می‌شود و بعد از اعمال تمام مراحل پیش پردازش، به مدل داده می شود و در تصویر بالا مشاهده می کنیم که مدل با دقت ۹۹/۸۴ درصد پیش بینی کرده است که عدد تصویر ۲ است. بنابراین مدل عملکرد خوبی داشته است.

تست حساسیت به نویز تصویر:

```
def test_noise_sensitivity(image, model, true_label):
    noise_levels = [0.01, 0.05, 0.1, 0.2]
    results = []

    for level in noise_levels:
        # اضافه کردن نویز گوسی
        noisy_img = image + np.random.normal(0, level, image.shape)
        noisy_img = np.clip(noisy_img, 0, 1)

        pred = model.predict(noisy_img[np.newaxis, ...])
        results.append({
            'noise_level': level,
            'prediction': np.argmax(pred),
            'confidence': np.max(pred),
            'correct': np.argmax(pred) == true_label
        })

    return pd.DataFrame(results)

# مثال استفاده:
results = test_noise_sensitivity(processed_img[0], model, 2)
print(results)
```

خروجی:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
   noise_level  prediction  confidence  correct
0         0.01           2    0.998394     True
1         0.05           2    0.998368     True
2         0.10           2    0.996616     True
3         0.20           2    0.997107     True
```

از جدول بالا مشاهده می شود که مدل در برابر نویز مقاوم است زیرا حتی با نویز ۲۰٪ (میزان قابل توجهی) پیش‌بینی صحیح انجام داده است و سطح اطمینان مدل تقریباً ثابت مانده(~99.7%) هم چنین پیش‌بینی در تمام سطوح نویز عدد ۲ بوده که نشان‌دهنده ثبات مدل است. علاوه بر این تغییرات درconfidence نیز بسیار ناچیز است.

تست حساسیت به چرخش تصویر:

```
def test_rotation_sensitivity(image, model, true_label):
    angles = range(-30, 31, 5)
    results = []

    for angle in angles:
        # چرخش تصویر
        rows, cols = image.shape[:2]
        M = cv2.getRotationMatrix2D((cols/2, rows/2), angle, 1)
        rotated = cv2.warpAffine(image, M, (cols, rows))

        pred = model.predict(rotated[np.newaxis, ..., np.newaxis])
        results.append({
            'angle': angle,
            'prediction': np.argmax(pred),
            'confidence': np.max(pred),
            'correct': np.argmax(pred) == true_label
        })

    return pd.DataFrame(results)

# مثال استفاده:
rotation_results = test_rotation_sensitivity(processed_img[0], model,
2)
print(rotation_results)
```

خروجی:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
    angle  prediction  confidence  correct
0     -30           2    0.973523     True
1     -25           2    0.980365     True
2     -20           2    0.971594     True
3     -15           2    0.992603     True
```

4	-10	2	0.994484	True
5	-5	2	0.993597	True
6	0	2	0.998379	True
7	5	2	0.992092	True
8	10	2	0.971332	True
9	15	2	0.945414	True
10	20	2	0.724209	True
11	25	7	0.647800	False
12	30	7	0.966644	False

از جدول بالا مشاهده می شود که مدل محدوده عملکرد قوی دارد زیرا مدل تا ±۲۰ درجه چرخش، عدد "۲" را **به درستی** تشخیص داده است و سطح اطمینان در این محدوده عموماً بالای ۹۷٪ بوده است. در زاویه ۲۰- درجه، اطمینان به ۷۲.۴٪ کاهش یافته اما همچنان پیش‌بینی صحیح است.

نقاط شکست مدل، در چرخش ۲۵ و ۳۰ درجه است. در چرخش ۲۵ درجه، مدل به اشتباه عدد را "۷" تشخیص داده با اطمینان ۶۴.۸% که این اولین خطای مدل در تست چرخش است. در چرخش ۳۰ درجه، مجدداً عدد را "۷" تشخیص داده اما این بار با اطمینان بسیار بالا(96.7%) که نشان می‌دهد با چرخش بیشتر، مدل کاملاً اشتباه می‌کند.

سطح اطمینان، در چرخش‌های کوچک (±۱۵ درجه)، بالای ۹۴٪ باقی می‌ماند. در ۲۰ درجه، به ۷۲.۴٪ سقوط می‌کند اما پیش‌بینی صحیح است و در چرخش‌های بیشتر، مدل کاملاً گمراه می‌شود اما با اطمینان بالا پیش‌بینی نادرست می‌کند.

میانگین زمان پیش‌بینی برای هر نمونه حدود 30-37میلی‌ثانیه است که این زمان برای کاربردهای عملی کاملاً مناسب خواهد بود.

نقاط قوت مدل:

- مقاومت خوب در برابر چرخش‌های کوچک (±۱۵ درجه)
- عملکرد سریع با زمان پردازش مناسب
- سطح اطمینان بالا در چرخش‌های محدود

نقاط ضعف

- مقاومت در برابر چرخش‌های بزرگ
- کاهش اطمینان کاذب
- بهبود تشخیص در چرخش‌های میانی

تست حساسیت به روشنایی تصویر:

خروجی:

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step
```

	brightness_factor	prediction	confidence	correct
0	0.5	2	0.916471	True
1	0.7	2	0.984076	True
2	1.0	2	0.998379	True
3	1.3	2	0.999249	True
4	1.5	2	0.999240	True
5	2.0	2	0.999257	True

از جدول بالا مشاهده می شود که مدل در تمام سطوح روشنایی آزمایش شده (از ۰.۵ تا ۲.۰) عدد "۲" را **به درستی** تشخیص داده است و سطح اطمینان مدل در تمام حالات بالای ۹۱٪ بوده است.

سطح اطمینان در روشنایی پایین (۰/۵) به ۹۱.۶٪ کاهش یافته (کمترین سطح در این تست) اما همچنان پیش‌بینی صحیح انجام شده است. در روشنایی نرمال (۱) سطح اطمینان ۹۹.۸٪ (بالاترین سطح در شرایط استاندارد) است و در روشنایی بالا (۱.۳ به بالا) اطمینان مدل به بیش از ۹۹.۹٪ رسیده است که نشان می‌دهد مدل در تصاویر روشن‌تر حتی عملکرد بهتری دارد.

میانگین زمان پیش‌بینی برای هر نمونه حدود **26-28میلی‌ثانیه** است که این زمان پردازش برای کاربردهای عملی بسیار مناسب است.

نقاط قوت مدل:

۱. مقاومت عالی در برابر تغییرات روشنایی:
   - از تصاویر تاریک (۰.۵) تا بسیار روشن (۲.۰) عملکرد پایدار دارد
   - این ویژگی برای کاربردهای واقعی که شرایط نور متغیر دارند بسیار ارزشمند است
۲. سطح اطمینان بالا:
   - حتی در بدترین حالت (تاریک‌ترین تصویر) اطمینان بالای ۹۱٪ دارد
   - در شرایط نرمال و روشن، اطمینان به ≈۱۰۰٪ می‌رسد