**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 داده‌ها

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 60000 تصویر آموزشی و 10000 تصویر تست است. تصاویر 28\*28پیکسل 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 3 لایه کانولوشنی با فعال‌ساز ReLU استفاده می‌کند. پس از هر دو لایه کانولوشنی یک لایه MaxPooling 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 5 دوره و اندازه دسته 128 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

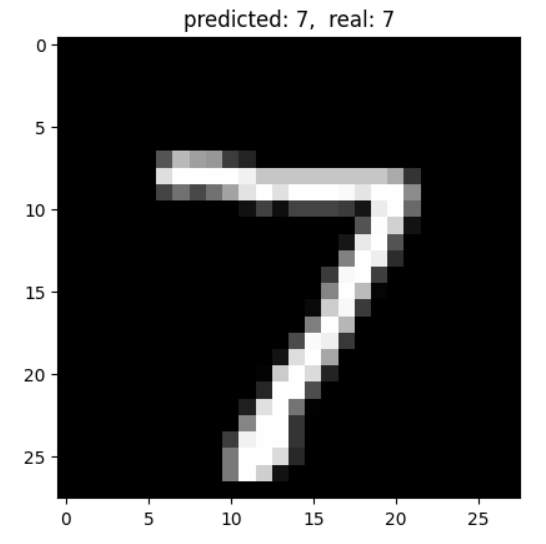
خروجی:

**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: 0.9898999929428101

**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 99 درصد است. بنابراین مدل عملکرد بسیار خوبی دارد و می تواند اعداد دست نویس را با دقت بالای 98 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 7 را نمایش می دهد که توسط مدل هم پیش بینی شده است.

همچنین در خروجی نشان داده شده است که ارزیابی مدل روی 313 دسته انجام شده است و زمان پردازش حدود 6 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')

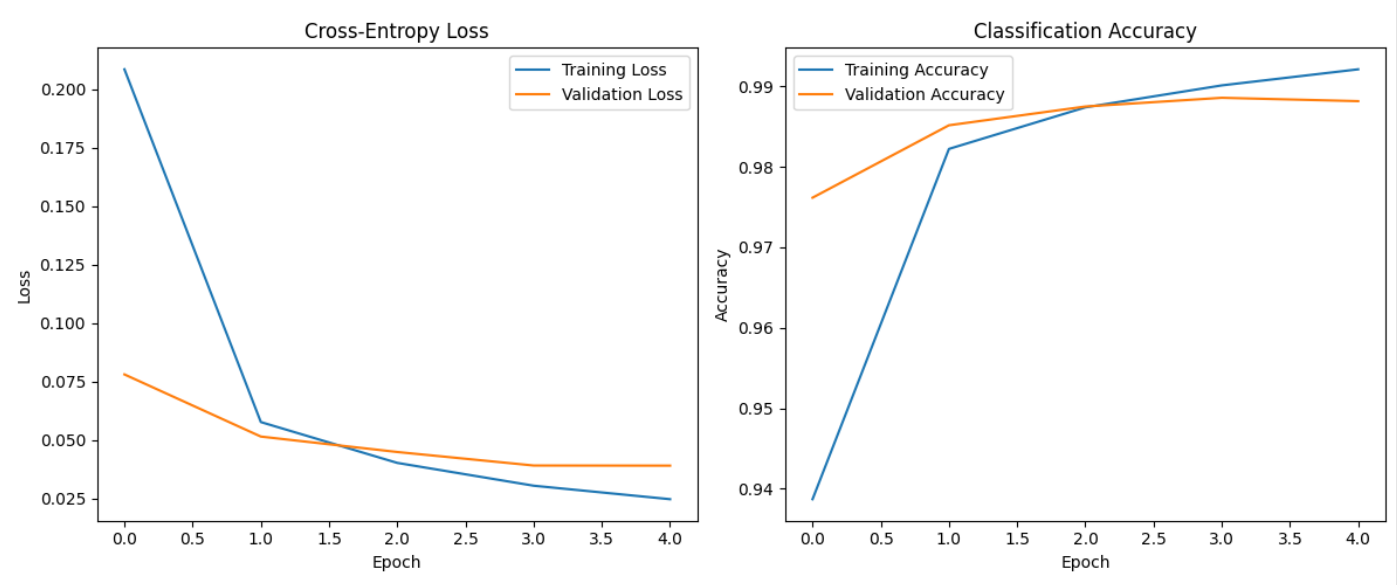
    plt.legend()

    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 0.2 شروع شده و به تدریج به زیر 0.025 کاهش یافته است.
* **خط اعتبارسنجی (Validation Loss)**: از حدود 0.150 شروع شده و به حدود 0.050 رسیده است.
* **روند کلی**: هر دو منحنی به صورت پایدار در حال کاهش هستند.

تحلیل:

1. **همگرایی مناسب**: هر دو منحنی به سمت پایین حرکت می‌کنند که نشان‌دهنده یادگیری صحیح مدل است
2. **فاصله بین منحنی‌ها**: فاصله نسبتاً کم (حدود 0.025 در پایان آموزش) نشان می‌دهد مدل دچار بیش‌برازش (overfitting) نشده است.
3. **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 0.95 شروع شده و به بیش از 0.99 رسیده است
* **خط اعتبارسنجی (Validation Accuracy)**: از حدود 0.96 شروع شده و به حدود 0.985 رسیده است
* **روند کلی**: هر دو منحنی به صورت یکنواخت در حال افزایش هستند

تحلیل:

1. **دقت نهایی عالی**: دقت اعتبارسنجی نزدیک به 98.5% نشان‌دهنده عملکرد بسیار خوب مدل است
2. **هماهنگی بین آموزش و اعتبارسنجی**: فاصله کم بین دو منحنی (حدود 0.5%) تأیید می‌کند مدل به خوبی تعمیم یافته است
3. **سرعت یادگیری**: مدل در همان دوره‌های اولیه به دقت بالا دست یافته که نشان‌دهنده معماری مناسب است.

رسم ماتریس آشفتگی:

# بارگیری داده‌های تست

(\_, \_), (test\_images, test\_labels) = mnist.load\_data()

test\_images = test\_images.reshape((10000, 28, 28, 1)).astype('float32') / 255

# پیش‌بینی روی داده‌های 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))

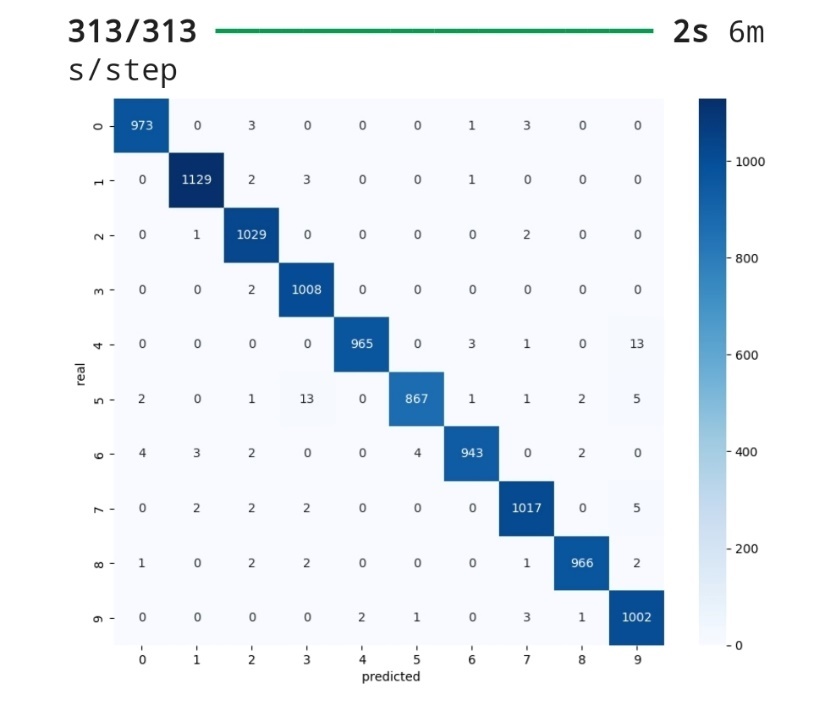
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',

            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 0 تا 9)
* **سطرها**: مقادیر واقعی (اعداد 0 تا 9)
* **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 1 بهترین عملکرد را داشته است. اما خطاهای قابل توجه عبارت اند از:

**عدد 5:**  13 مورد از عدد 5 به اشتباه به عنوان عدد 3 طبقه‌بندی شده‌اند.

**عدد 9**: 5 مورد به اشتباه به عنوان عدد 7 طبقه‌بندی شده‌اند.

**عدد 6**: 5 مورد به اشتباه به عنوان عدد 0 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)

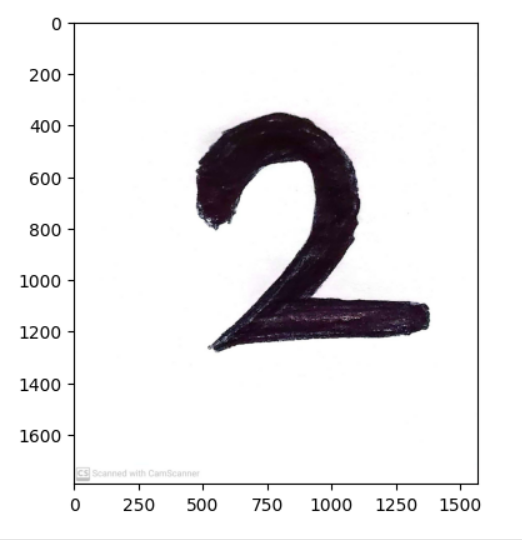
from PIL import Image

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("\nتصویر پس از پردازش (آماده برای مدل):")

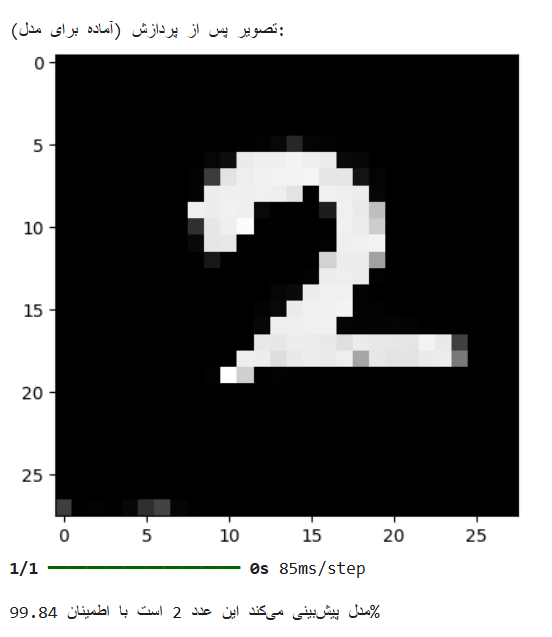
plt.imshow(processed\_img.reshape(28, 28), cmap='gray')

plt.show()

prediction = model.predict(processed\_img)

predicted\_number = np.argmax(prediction)

print(f"\nمدل پیش‌بینی می‌کند این عدد {predicted\_number} است با اطمینان {np.max(prediction)\*100:.2f}%")

خروجی:

ابعاد تصویر اصلی به 28 \*28 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 84/99 درصد پیش بینی کرده است که عدد تصویر 2 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 20% (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 2 بوده که نشان‌دهنده ثبات مدل است. علاوه بر این تغییرات در confidence 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 ±20 درجه چرخش، عدد "2" را **به درستی** تشخیص داده است و سطح اطمینان در این محدوده عموماً بالای 97% بوده است. در زاویه 20- درجه، اطمینان به 72.4% کاهش یافته اما همچنان پیش‌بینی صحیح است.

نقاط شکست مدل، در چرخش 25 و 30 درجه است. در چرخش 25 درجه، مدل به اشتباه عدد را "7" تشخیص داده با اطمینان 64.8% که این اولین خطای مدل در تست چرخش است. در چرخش 30 درجه، مجدداً عدد را "7" 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 (±15 درجه)، بالای 94% باقی می‌ماند. در 20 درجه، به 72.4% 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**-**37 میلی‌ثانیه است که این زمان برای کاربردهای عملی کاملاً مناسب خواهد بود.

نقاط قوت مدل:

* مقاومت خوب در برابر چرخش‌های کوچک (±15 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 0.5 تا 2.0) عدد "2" را **به درستی** تشخیص داده است و سطح اطمینان مدل در تمام حالات بالای 91% بوده است.

سطح اطمینان در روشنایی پایین (5/0) به 91.6% کاهش یافته (کمترین سطح در این تست) اما همچنان پیش‌بینی صحیح انجام شده است. در روشنایی نرمال (1) سطح اطمینان 99.8% (بالاترین سطح در شرایط استاندارد) است و در روشنایی بالا (1.3 به بالا) اطمینان مدل به بیش از 99.9% رسیده است که نشان می‌دهد مدل در تصاویر روشن‌تر حتی عملکرد بهتری دارد.

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

نقاط قوت مدل:

1. مقاومت عالی در برابر تغییرات روشنایی:
   * از تصاویر تاریک (0.5) تا بسیار روشن (2.0) عملکرد پایدار دارد
   * این ویژگی برای کاربردهای واقعی که شرایط نور متغیر دارند بسیار ارزشمند است
2. سطح اطمینان بالا:
   * حتی در بدترین حالت (تاریک‌ترین تصویر) اطمینان بالای 91% دارد
   * در شرایط نرمال و روشن، اطمینان به ≈100% می‌رسد