به نام خالق رنگین کمان

ستاره باباجانی – گزارش تمرین سری 6

سوال 1:

الف)

|  |  |  |
| --- | --- | --- |
| **layer** | **Output shape** | **Number of parameters** |
| Input | (512, 512, 3) | 0 |
| Conv2D(32, (9, 9), strides=2, padding='same') | (256, 256, 32) | 7,808(9\*9\*3\*32 + 32) |
| MaxPooling2D((4, 4), strides=4) | (64, 64, 32) | 0 |
| Conv2D(64, (5, 5), strides=1) | (60, 60, 64) | 51,264 (5\*5\*32\*64 + 64) |
| AveragePooling2D((2, 2), strides=2) | (30, 30, 64) | 0 |
| Conv2D(128, (3, 3), strides=1, padding='valid') | (28, 28, 128) | 73,856 (3\*3\*64\*128 + 128) |
| Conv2D(128, (3, 3), strides=1, padding='same') | (28, 28, 128) | 147,584 (3\*3\*128\*128 + 128) |
| MaxPooling2D((2, 2), strides=2) | (14, 14, 128) | 0 |
| Conv2D(512, (3, 3), strides=1, padding='valid') | (12, 12, 512) | 590,336 (3\*3\*128\*512 + 512) |
| GlobalAveragePooling2D() | (512) | 0 |
| Dense(1024) | (1024) | 525,312 (512\*1024 + 1024) |
| Dense(10) | (10) | 10,250 (1024\*10 + 10) |

ب)

|  |  |  |
| --- | --- | --- |
| **Additions** | **Multiplications** | **Layer** |
| 256\*256\*32\*(3\*81 - 1) | 256\*256\*32\*3\*9\*9 | Conv2D(32, (9, 9), strides=2, padding='same') |
| 64\*64\*32\*15 | 0 | MaxPooling2D((4, 4), strides=4) |
| 60\*60\*64\*(32\*25 - 1) | 60\*60\*32\*64\*5\*5 | Conv2D(64, (5, 5), strides=1) |
| 30\*30\*64\*3 | 0 | AveragePooling2D((2, 2), strides=2) |
| 28\*28\*128\*(64\*9 - 1) | 28\*28\*64\*128\*3\*3 | Conv2D(128, (3, 3), strides=1, padding='valid') |
| 28\*28\*128\*(128\*9 - 1) | 28\*28\*128\*128\*3\*3 | Conv2D(128, (3, 3), strides=1, padding='same') |
| 14\*14\*128\*3 | 0 | MaxPooling2D((2, 2), strides=2) |
| 12\*12\*512\*(128\*9 - 1) | 12\*12\*128\*512\*3\*3 | Conv2D(512, (3, 3), strides=1, padding='valid') |
| 512\*(12\*12 - 1) | 0 | GlobalAveragePooling2D() |
| 512\*1024 + 1024 | 512\*1024 | Dense(1024) |
| 1024\*10 + 10 | 1024\*10 | Dense(10) |

ج) تعداد پارامترهای سه لایه آخر به شرح زیر خواهد شد:

**Flatten():** 0 parameters

**Dense(1024)**: 73,728×1024+1024=75498496 parameters

**Dense(10)**: 1024×10+10=10250 parameters

* Total count of parameters with Flatten: 76379594
* Total count of parameters with Flatten: 1406410

که نسبت آنها 54.31 میشود.

سوال 2: ابتدا مشتق L نسبت به X را باید بدست آوریم که برابر با 2X – 10 میشود. پس مقدار گرادیان با در نظر گرفتن مقدار 20 برای X، 30 میشود.

همانطور که میدانیم فرمول آپدیت X طبق GD به شرح زیر است:

𝑋𝑛𝑒𝑤 = 𝑋𝑜𝑙𝑑 – η \* 𝑑𝐿/𝑑𝑋

حال به بررسی هرکدام از مقادیر X جدید داده شده میپردازیم:

1. X = 14 => 14 = 20 – η \*30 => η = 0.2
2. X = 19.4 => 19.4 = 20 – η \*30 => η = 0.02
3. X = 19.94 => 19.94 = 20 – η \*30 => η = 0.002

همانطور که میدانیم هرچه نرخ یادگیری بیشتر باشد، مقدار ضرر سریعتر کم میشود. پس اولین گزینه مربوط به نمودار c، دومی مربوط به b و سومی مربوط به a میباشد.

سوال 3: اگر برای همه‌ی بلوک‌ها پدینگ یکسان درنظر بگیریم، خواهیم داشت:

Conv 1×1 ⇒ (12, 12, 64)

Conv 1×1 and Conv 3×3 ⇒ (12, 12, 32)

Conv 1×1 and Conv 5×5 ⇒ (12, 12, 128)

max-pool 3×3 and Conv 1×1 ⇒ (12, 12, 64)

با concat کردن آنها، ابعاد خروجی (288, 12, 12) میشود.

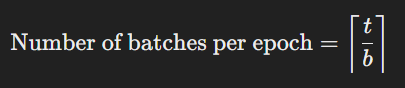
اگر فیلتر بلوک کانولوشن انتخابی را تغییر دهیم، خروجی متفاوت نخواهد بود زیرا بعد

آن بلوک، ما بلوک کانولوشن دیگری داریم که باعث میشود خروجی همان شود.

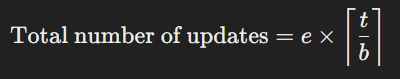
سوال 4: الف) برای تعیین اینکه وزن‌های شبکه چند بار در طول آموزش به‌روزرسانی می‌شوند، باید تعداد کل نمونه‌های آموزشی (t)، تعداد دوره‌ها (e) و اندازه دسته (b) را در نظر بگیریم.

تعداد دفعاتی که وزن شبکه به روز می شود را می توان به صورت زیر محاسبه کرد:

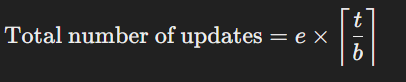
* تعداد دسته در هر دوره: این تعداد کل نمونه های آموزشی (t) تقسیم بر اندازه دسته (b) است:



* تعداد کل به روز رسانی ها: این تعداد دسته در هر دوره ضرب در تعداد دوره های (e) است:



بنابراین، فرمول محاسبه تعداد کل به روز رسانی وزن شبکه در طول آموزش به صورت زیر است:



ب) نمودار داده شده مربوط به mini-batch GD می‌باشد زیرا که در این روش برخلاف batch GD کل مجموعه داده به batch های کوچکتر تقسیم میشود و برای هرکدام از آن batch ها، آپدیت جدا داریم.(علت نویزی شدن loss ها هم همین است.)

ج) برای نمودار A، به دلیل تفاوت زیاد ضرر در validation و training(ضرر در validation خیلی بیشتر است)، overfitting رخ داده است که برای حل آن از داده‌افزایی و کاهش تعداد ویژگی‌های ورودی استفاده میکنیم تا پیچیدگی مدل را کمتر کنیم(با افزایش لایه‌های شبکه، مدل پیچیده‌تر میشود و نتیجه بدتر میشود). برای نمودار B نیز، ضرر validation و training شبیه به هم است و مدل تعمیم پذیری نسبتا خوبی را از خود نشان داده است، پس با افزودن لایه‌های شبکه، مدل میتواند پیچیدگی‌های بیشتری را یاد بگیرد و خروجی بهتری داشته باشد.(در اینجا استفاده از داده‌افزایی و کاهش تعداد ویژگی‌های ورودی ممکن است فایده نداشته باشد.)

سوال 5: الف) برای هر پیکسل غیر صفر باید همسایگان 8-connectivity آن را در نظر بگیریم. سپس، هر همسایه باید با پیکسل مرکزی مقایسه شود تا بررسی شود که بزرگتر یا مساوی آن است، که اگر بود 1 میشود و در غیراینصورت 0 میشود. سپس یک کد باینری از جهت عقربه های ساعت میسازیم:

(1, 1): 00000000

(1, 2): 00000001

(1, 3): 00000111

(2, 1): 01100100

(2, 2): 11000011

(2, 3): 10000001

(3, 1): 00000000

(3, 2): 11110001

(3, 3): 11000000

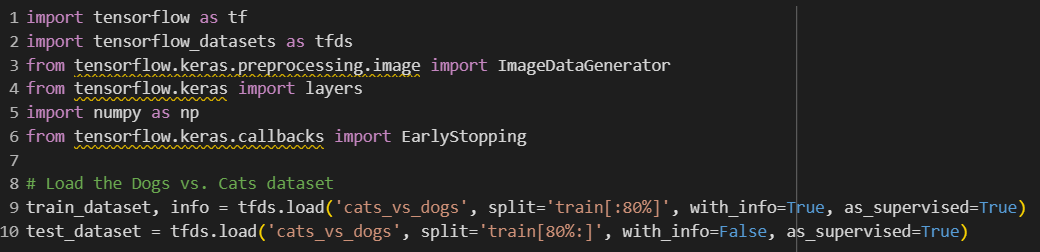
ب) اگر متغیر c بزرگتر از 0 باشد، چه در حالت ضرب و چه جمع، تغییری در کد ایجاد نمیشود.

ج)

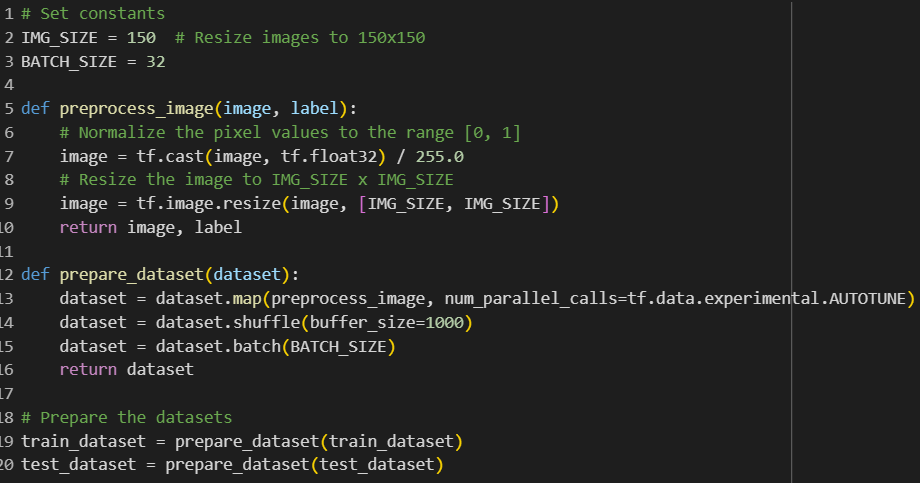
* هیستوگرام A با تصویر سوم مطابقت دارد زیرا تصویر از تعداد زیادی سیاه و سفید تشکیل شده است و در خود هیستوگرام نیز توزیع نسبتا یکنواخت‌تری وجود دارد.
* هیستوگرام B با تصویر اول مطابقت دارد زیرا اگر بخواهیم تصویر را به LBP تبدیل کنیم، از بسیاری از پیکسل های سفید (سیاه های موجود در تصویر) و بسیاری از پیکسل های خاکستری تشکیل می شود.
* هیستوگرام C با تصویر دوم مطابقت دارد زیرا تصویر LBP از تعداد زیادی پیکسل سفید و کمی سیاه یا خاکستری تشکیل شده است.

سوال 6: الف) مراحل ساخت و آموزش CNN به شرح زیر است:

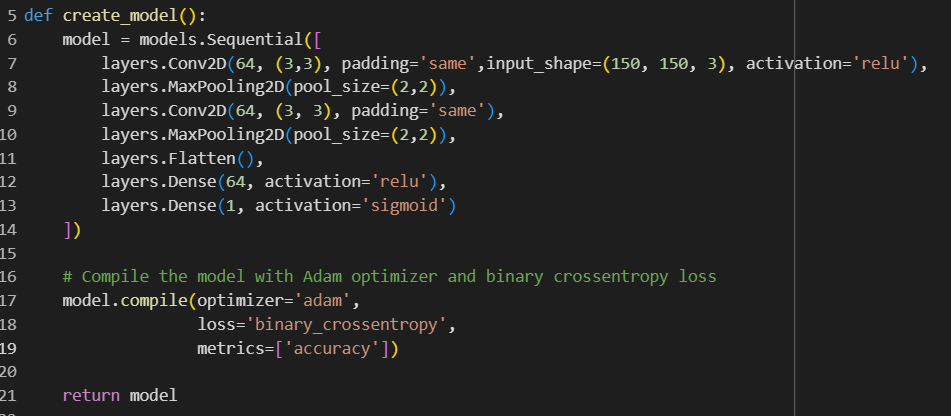
* صدا زدن کتابخانه‌های مورد نیاز و load کردن دیتا:

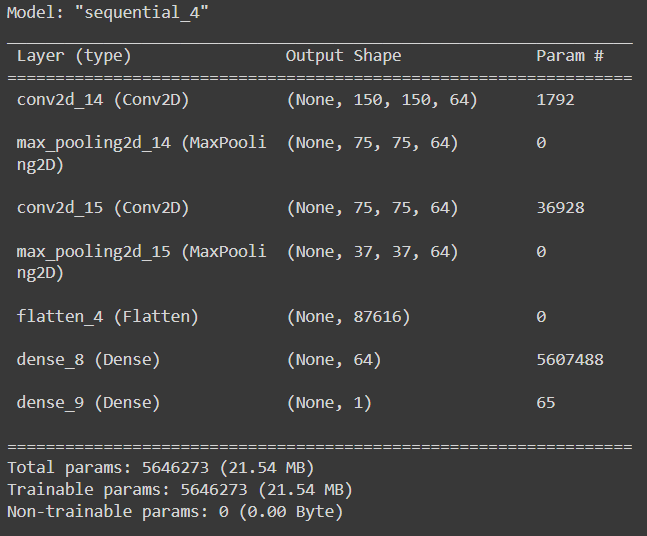


* Preprocessing:

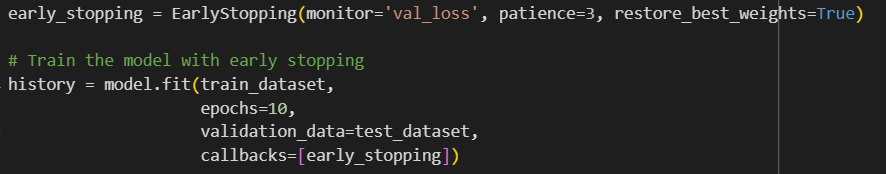


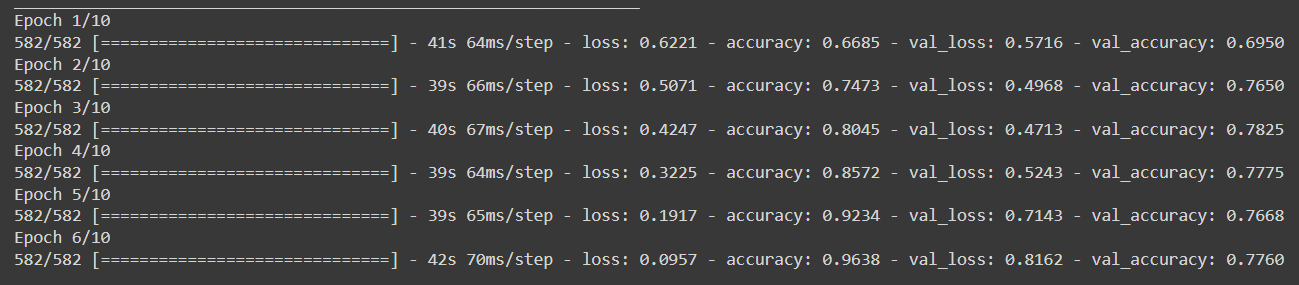
* تعریف مدل:





* آموزش مدل: همانطور که مشاهده میشود از تکنیک early-stopping برای جلوگیری از overfitting استفاده شده است:

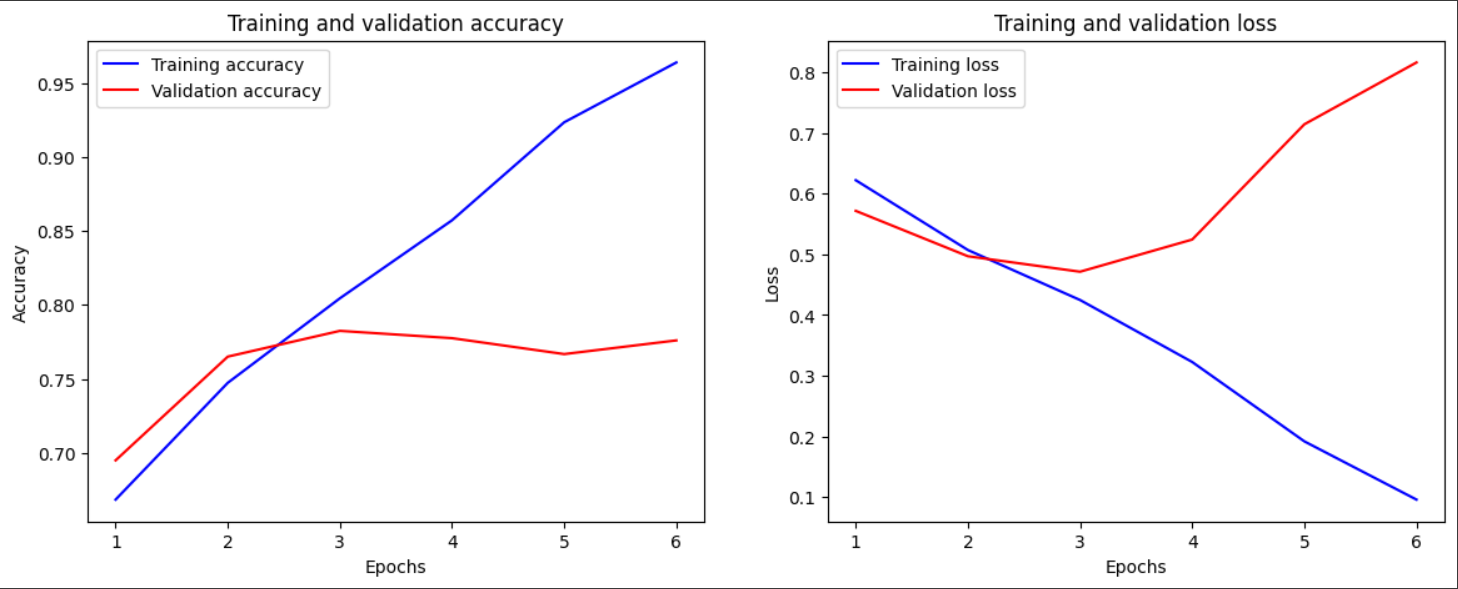




که در اینجا مقدار ضرر و accuracy روی داده های تست به شرح زیر است:



* رسم نمودار و نشان دادن خروجی‌ها:



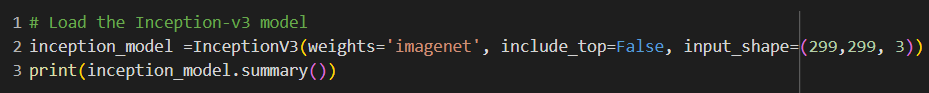
ب) حال در این بخش از یک مدل pre-trained استفاده میکنیم. مراحل preprocessing عین بخش الف است.

* ابتدا چند نمونه نشان میدهیم:

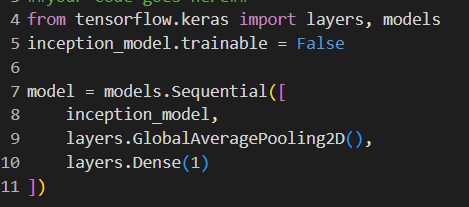


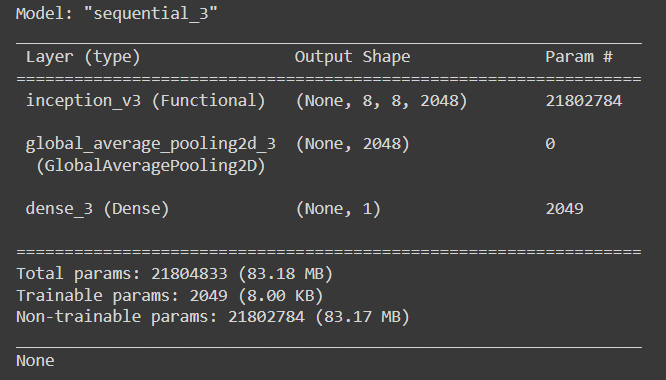


* مدل از پیش آموزش داده شده را صدا میکنیم:

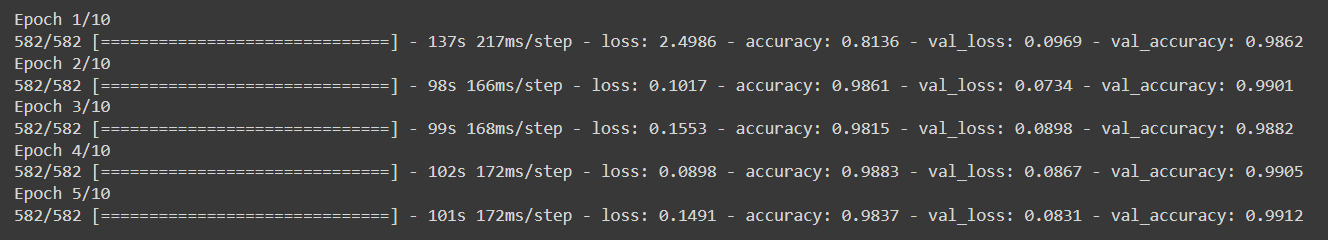


* پارامترهای آن را فریز کرده و سپس یک GAP و یک لایه Dense اضافه میکنیم:





* آموزش مدل:

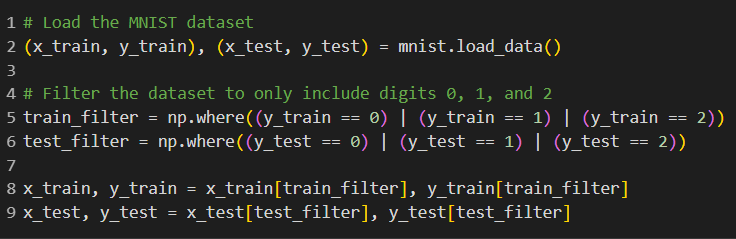


* دقت روی داده تست:

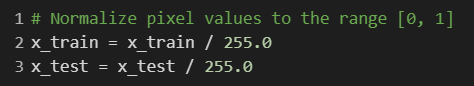


سوال 7: کد خواسته شده به شرح زیر است:

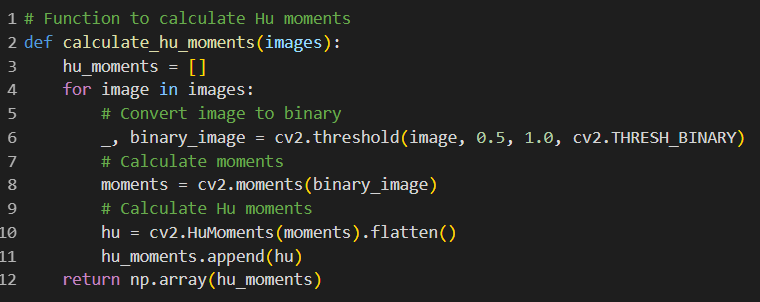
1. دانلود مجموعه داده و محدود کردن آن به ارقام 0 و 1 و 2:



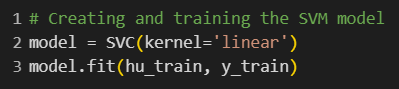
1. نرمال سازی مقادیر: همان طور که در کد ذکر شده، چون عکس های MNIST خودشان 28\*28 هستند، نیاز به resize کردن نیست.



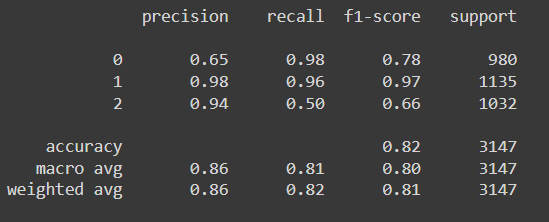
1. محاسبه Hu Moments: همان طور که میدانیم Hu Moments مجموعه‌ای از هفت عدد هستند که نسبت به تبدیل‌های تصویر مانند ترجمه، مقیاس‌بندی و چرخش ثابت هستند. آنها از لحظه های تصویر گرفته شده اند، که ویژگی های شکل اشیاء را در یک تصویر ثبت می کنند. Hu Moments به ویژه برای تجزیه و تحلیل شکل و وظایف تشخیص الگو مفید هستند زیرا آنها نمایش فشرده و کارآمدی از شکل را ارائه می دهند.



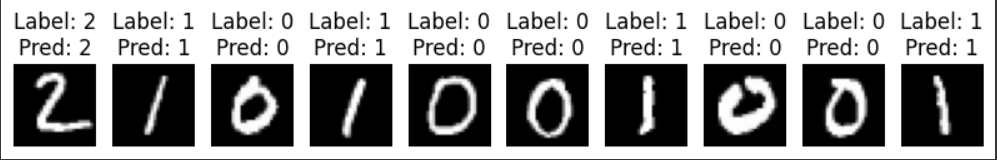
1. انتخاب مدل و آموزش بر روی دیتا: همان طور که میدانیم SVC یک مدل قوی و مورد استفاده در مسائل طبقه بندی است.



1. خروجی: مقدار دقت بر روی داده تست به شرح زیر است:



حال به نمایش برخی از عکس ها با لیبل واقعی و پیش بینی شده آنها میپردازیم:



همچنین برای درک بهتر، چند نمونه داده اشتباه برچسب گذاری شده نیز نمایش میدهیم:

