به نام خدا

تمرین ششم – بینایی کامپیوتر **نیایش خانی ۹۹۵۲۱۲۳۵**

سوال ۱)

الف)

سوال ۱. الف)

layer 1) output shape : (512, 512, 3) , Parameters = 0

layer 2) output height, width = $\frac{512}{2}$ = 256

output Depth = 32 → output shape = (256, 256, 32)

parameters = 32 × (9 × 9 × 3 + 1) = 7808

layer 3) output height, width = $\frac{256}{4}$ = 64 → output shape = (64, 64, 32)

parameter = 0

layer 4) output height, width = 64 − 5 + 1 = 60

output Depth = 64 → output shape = (60, 60, 64)

parameter = 64 (5 × 5 × 32 + 1) = 51264

layer 5) output height, width = $\frac{60}{2}$ = 30 → output shape = (30, 30, 64)

parameter = 0

layer 6) output height, width = 30 − 3 + 1 = 28

output depth = 128 → output shape = (28, 28, 128)

parameter = (3 × 3 × 64 × 128) + 128 = 73856

layer 7) output height, width = 28 , depth = 128 → output shape = (28, 28, 128)

Parameter = (3 × 3 × 128 × 128) × 128 = 147584

layer 8) output height, width = $\frac{28}{2}$ = 14 → output shape = (14, 14, 128)

Parameter = 0

layer 9) output height, width = 14 - 3 + 1 = 12
output depth = 512 → output shape = (12, 12, 512)
parameter = (3 × 3 × 128 × 512) + 512 = 590,336

layer 10) output shape = 512 parameter = 0

layer 11) output shape = 1024 parameter = (512 × 1024) + 1024 = 525312

layer 12) output shape = 10 parameter = (1024 × 10) + 10 = 10250

ب)

$layer\ 1)$ $mul = 0$, $add = 0$

$layer\ 2)$ $mul = 256 \times 256 \times 32 \times 3 \times 9 \times 9 = 509{,}607{,}936$

$add = 256 \times 256 \times 32\ (3 \times 9 \times 9 - 1) = 507{,}510{,}784$

$layer\ 3)$ $mul = 0$

$add = 64 \times 64 \times 32 \times (4 \times 4 - 1) = 1966080$

$layer\ 4)$ $mul = 60 \times 60 \times 64 \times 32 \times 5 \times 5 = 184320000$

$add = 60 \times 60 \times 64\ (32 \times 5 \times 5 - 1) = 184089600$

$layer\ 5)$ $mul = 0$

$add = 30 \times 30 \times 64\ (2 \times 2 - 1) = 172800$

$layer\ 6)$ $mul = 28 \times 28 \times 128 \times 64 \times 3 \times 3 = 57802752$

$add = 28 \times 28 \times 128\ (64 \times 3 \times 3 - 1) = 57702400$

$layer\ 7)$ $mul = 28 \times 28 \times 128 \times 128 \times 3 \times 3 = 115605504$

$add = 28 \times 28 \times 128\ (128 \times 3 \times 3 - 1) = 115505152$

$layer\ 8)$ $mul = 0$

$$add = 14 \times 14 \times 128\,(4 \times 4 - 1) = 376320$$

$layer\ 9)\quad mul = 12 \times 12 \times 512 \times 128 \times 3 \times 3 = 84934656$

$$add = 12 \times 12 \times 512\,(128 \times 3 \times 3 - 1) = 84860928$$

$layer\ 10)\quad mul = 0$

$$add = 512\,(12 \times 12 - 1) = 73216$$

$layer\ 11)\quad mul = 512 \times 1024 = 524288$

$$add = 512 \times 1024 + 1024 = 525312$$

$layer\ 12)\quad mul = 1024 \times 10 = 10240$

$$add = 1024 \times 10 + 10 = 10250$$

ج) با استفاده از flatten برای سه لایه آخر به این پارامترها می رسیم :

$layer\ 10)\ output\ shape = 73728 \qquad parameters = 0$

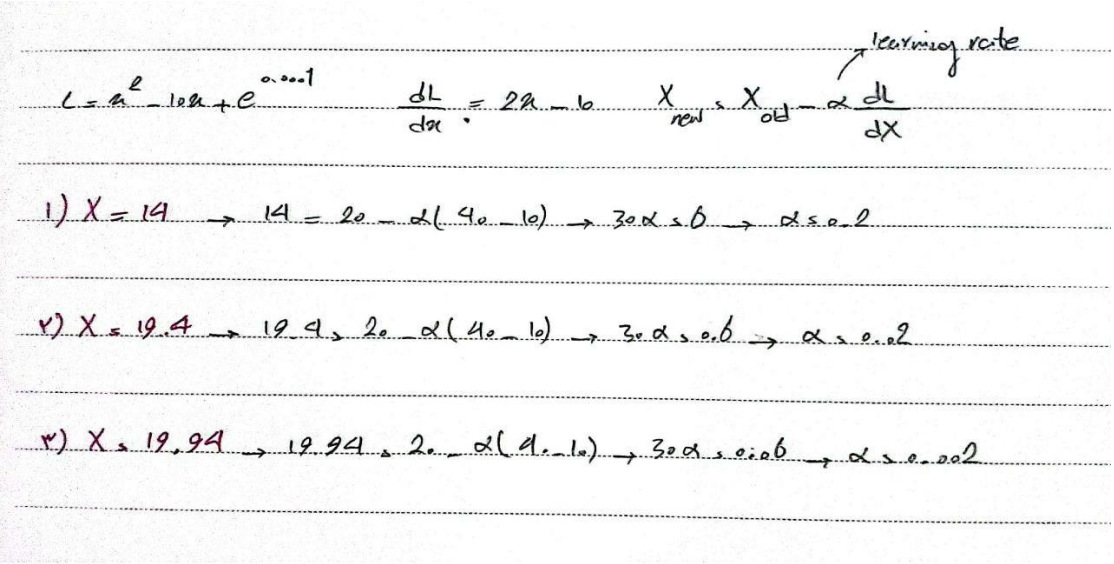
$layer\ 11)\ output\ shape = 1024 \qquad parameters = 73728 \times 1024 \times +1024 = 75498496$

$layer\ 12)\ output\ shape = 10 \qquad parameters = 1024 \times 10 + 10 = 10250$

تعداد کل پارامترها با استفاده از GAP، 1406410 و با استفاده از Flatten ، 76379594 است بنابراین 54.31 برابر شده است.

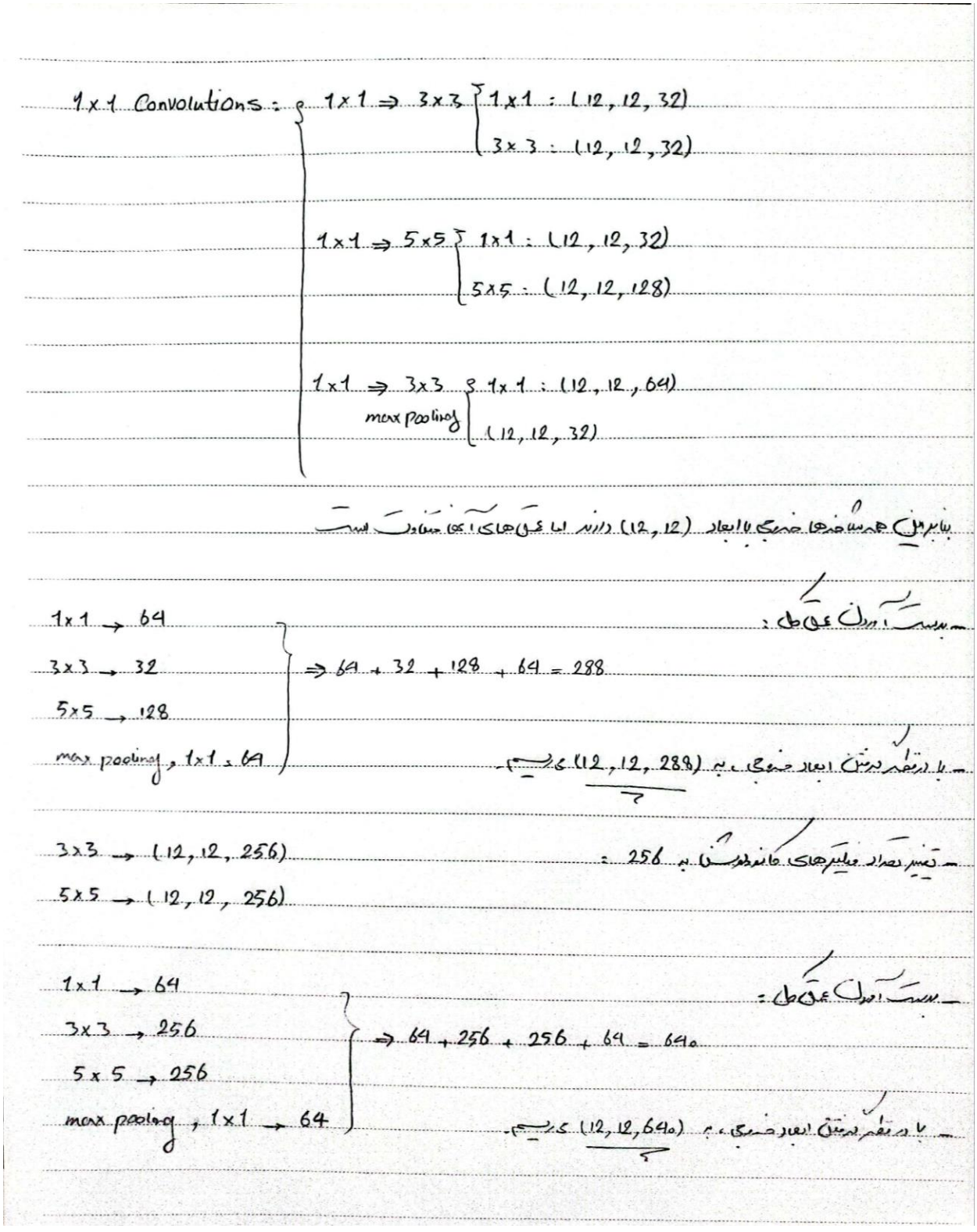
$$\frac{76379594}{1406410} = 54.31$$

سوال ۲)



هرچقدر که مقدار α بیشتر باشد، مقدار x در هر مرحله بسیار سریع تغییر می کند و زود به کمینه loss می رسیم. همانطور که مشاهده می شود بیشترین مقدار α مربوط به X = 14 است پس نمودار نارنجی مربوط به این x است .α = 0.02
مقدار بعدی است بنابراین نمودار سبز برای X = 19.4 و کمترین مقدار نیز مربوط به X = 19.94 است که نمودار آبی به آن تعلق دارد.

سوال ۳)


1x1 Convolutions :
1x1 ⇒ 3x3
1x1 : (12, 12, 32)
3x3 : (12, 12, 32)
1x1 ⇒ 5x5
1x1 : (12, 12, 32)
5x5 : (12, 12, 128)
1x1 ⇒ 3x3
max pooling
1x1 : (12, 12, 64)
(12, 12, 32)
بنابراین همه‌ی خروجی‌ها ابعاد (12, 12) دارند پس عمق‌های آن‌ها جمع می‌شود
بعد آرایه‌ی اول عمق:
1x1 → 64
3x3 → 32
5x5 → 128
max pooling, 1x1 → 64
⇒ 64 + 32 + 128 + 64 = 288
با در نظر گرفتن ابعاد خروجی، به (12, 12, 288) می‌رسیم.
تعیین تعداد فیلترهای کانولوشن به 256 :
3x3 → (12, 12, 256)
5x5 → (12, 12, 256)
بعد آرایه‌ی دوم عمق:
1x1 → 64
3x3 → 256
5x5 → 256
max pooling, 1x1 → 64
⇒ 64 + 256 + 256 + 64 = 640
با در نظر گرفتن ابعاد خروجی، به (12, 12, 640) می‌رسیم.

سوال ۴)

الف - تعداد به‌روزرسانی‌ها در هر دوره برابر است با :

$$\text{Number of updates per epoch} = \left\lceil \frac{t}{b} \right\rceil$$

(در اینجا، ما از تابع سقف استفاده می‌کنیم تا احتمال اینکه آخرین دسته ممکن است یک دسته کامل نباشد، اگر t کاملاً بر b تقسیم‌پذیر نباشد، را در نظر بگیریم.)

با توجه به این فرمول، تعداد کل به روز رسانی ها عبارت است از:

$$\text{Total number of updates} = e \times \left\lceil \frac{t}{b} \right\rceil$$

ب - Batch GD گرادیان را با استفاده از کل مجموعه داده محاسبه می کند و معمولا منجر به یک منحنی loss هموارتر می شود زیرا گرادیان ها را در تمام نقاط داده به طور میانگین می دهد.

از طرفی mini batch GD گرادیان را با استفاده از زیرمجموعه ای از مجموعه دیتا محاسبه می کند و به دلیل واریانس محاسبه شده توسط دسته های کوچکتر، منحنی loss با نویز بیشتری ایجاد می کند.

این نمودار نیز یک منحنی loss نویزی را نشان می دهد که نشان دهنده نوسانات در مقادیر تلفات در تمام دوره ها است. بنابراین این نویز برای mini-batch GD است، زیرا دسته های کوچکتر، تغییرپذیری را در به روز رسانی گرادیان ایجاد می کنند.

د) در منحنی A ، training loss کاهش می یابد و سپس با مقدار بسیار کم هموار می شود. Validation loss نیز کاهش می یابد اما بالاتر از training loss در طول فرآیند train باقی می ماند. این نشان می دهد که مدل overfit شده است.

در منحنی B، training loss و validation loss در طول دوره ها به طور پیوسته کاهش می یابند و این نشان می‌دهد که مدل ضعیف است و مدل به اندازه کافی الگوهای اساسی را یاد نمی‌گیرد.

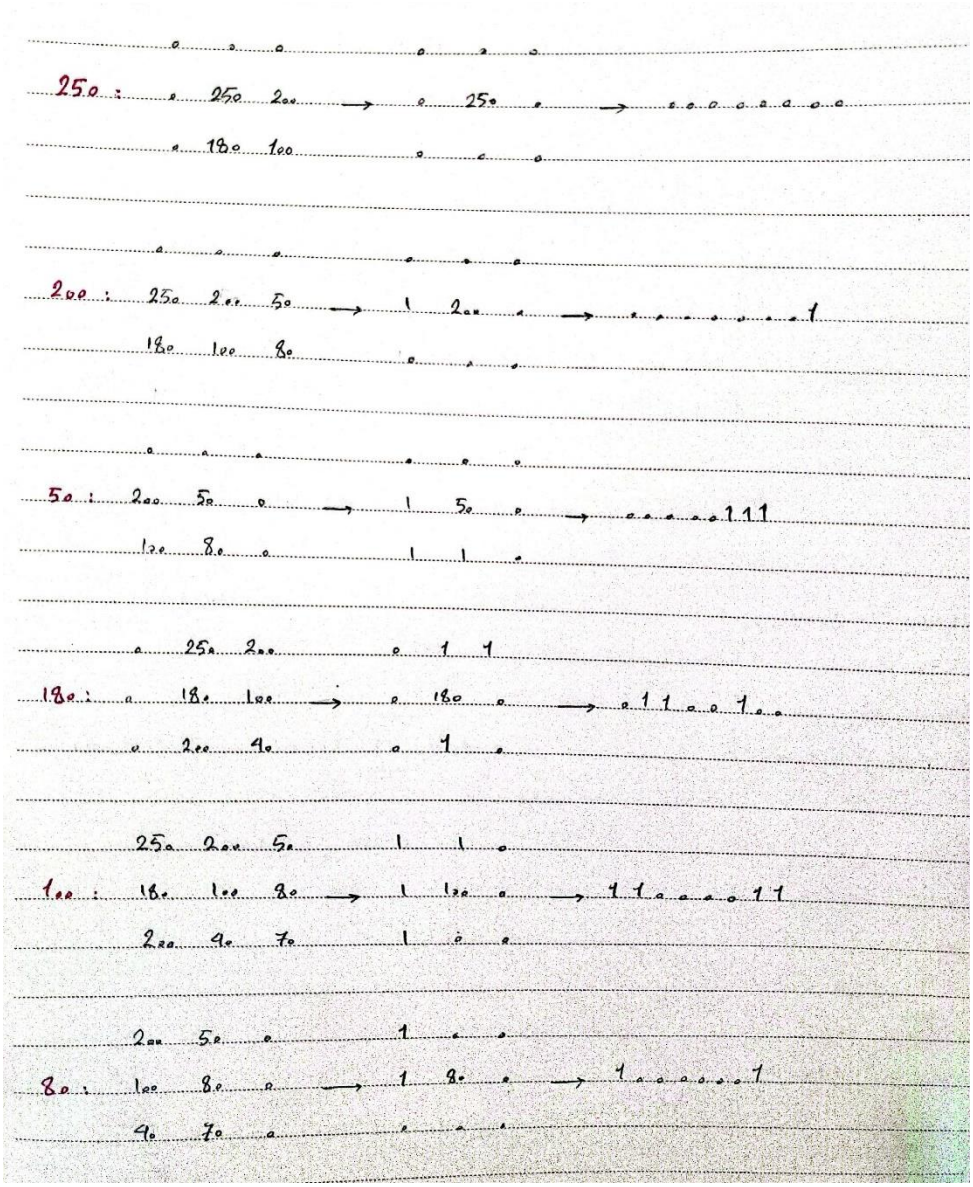
۱. افزودن داده ها : در منحنی افزودن داده های بیشتر می تواند به تعمیم بهتر مدل کمک کند و شکاف بین آموزش و از دست دادن اعتبار سنجی را کاهش دهد و برای overfitting مناسب است.
در منحنی B ، افزودن داده‌های بیشتر می‌تواند نمونه‌های بیشتری را در اختیار مدل قرار دهد تا یادگیری بهتر شود.

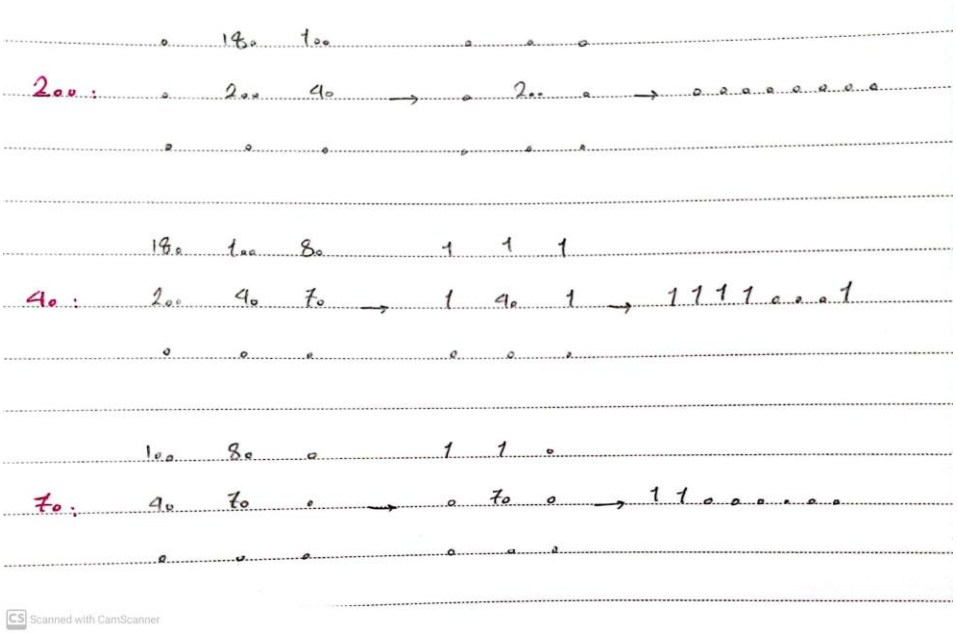
۲. افزایش لایه های شبکه : این کار برای A توصیه نمی شود زیرا می تواند با پیچیده تر کردن مدل، overfitting را تشدید کند.
در B ، افزایش تعداد لایه های شبکه می تواند به مدل کمک کند تا الگوهای پیچیده تری را بیاموزد و یادگیری را بهتر کند.

۳. کاهش تعداد ویژگی های ورودی : برای منحنی A ، کاهش تعداد ویژگی های ورودی می تواند به کاهش پیچیدگی مدل و کاهش بیش از حد برازش با تمرکز بر مرتبط ترین ویژگی ها کمک کند.

اما در منحنی B ، کاهش ویژگی های ورودی توصیه نمی شود زیرا می تواند اطلاعات بالقوه مهمی را که مدل نیاز به یادگیری موثر آنها دارد حذف کند.

سوال ۵) الف - برای بدست آوردن کد، همسایگی ذکر شده را برای هر پیکسل در نظر می گیریم. اگر همسایه بزرگتر از پیکسل بود، ۱ و در غیر این صورت ۰ می گذاریم. کد حاصل شروع از پیکسل (۰و۰) و قرار دادن ۰ و ۱ ها به صورت ساعتگرد پشت یکدیگر است.





ب) با اضافه کردن و یا جمع کردن عدد ثابت بزرگتر از صفر، تغییری در کد ایجاد نمی شود.

ج) هیستوگرام A تقریبا یکنواخت است و رنگ های خاکستری زیادی را در بر دارد بنابراین می تواند برای تصویر سوم باشد.

هیستوگرام B نیز از بسیاری از پیکسل های سفید (سیاه های موجود در تصویر) و بسیاری از پیکسل های خاکستری تشکیل شده است که با تصویر اول تطابق بیشتری دارد.

هیستوگرام C نیز که بیشتر از پیکسل های سفید و سیاه تشکیل شده است، برای تصویر دوم است زیرا تصویر LBP از تعداد زیادی پیکسل سفید و کمی سیاه یا خاکستری تشکیل شده است که خطی را نشان می دهد که کدام رنگ تغییر کرده است.

سوال ۶)

الف - ابتدا دیتا های train و تست را آماده می کنیم. سپس با استفاده از فانکشن preprocess_image اطمینان حاصل می کنیم که تابع به درستی هم تصویر و هم برچسب را به عنوان آرگومان می گیرد.

بوسیله تابع map، preprocess_image را برای هر عنصر در مجموعه داده اعمال می کنیم ، تابع shuffle داده های آموزشی را به هم می زند و داده ها را به دسته های ۳۲ تایی گروه بندی می کند. برای بهتر عمل کردن این مدل از prefetch استفاده کردم که عملکرد را با تهیه دسته بعدی در حالی که دسته فعلی در حال پردازش است، بهبود می بخشد.

معماری مدل:

یک مدل ترتیبی با لایه‌های Conv2D، MaxPooling2D، Flatten و Dense داریم. لایه Dense با تابع فعال سازی سیگموئید برای طبقه بندی باینری استفاده می شود.

سپس مدل را کامپایل می کنیم و از متود های رایج adam، binary_crossentropy و accuracy استفاده می کنیم.

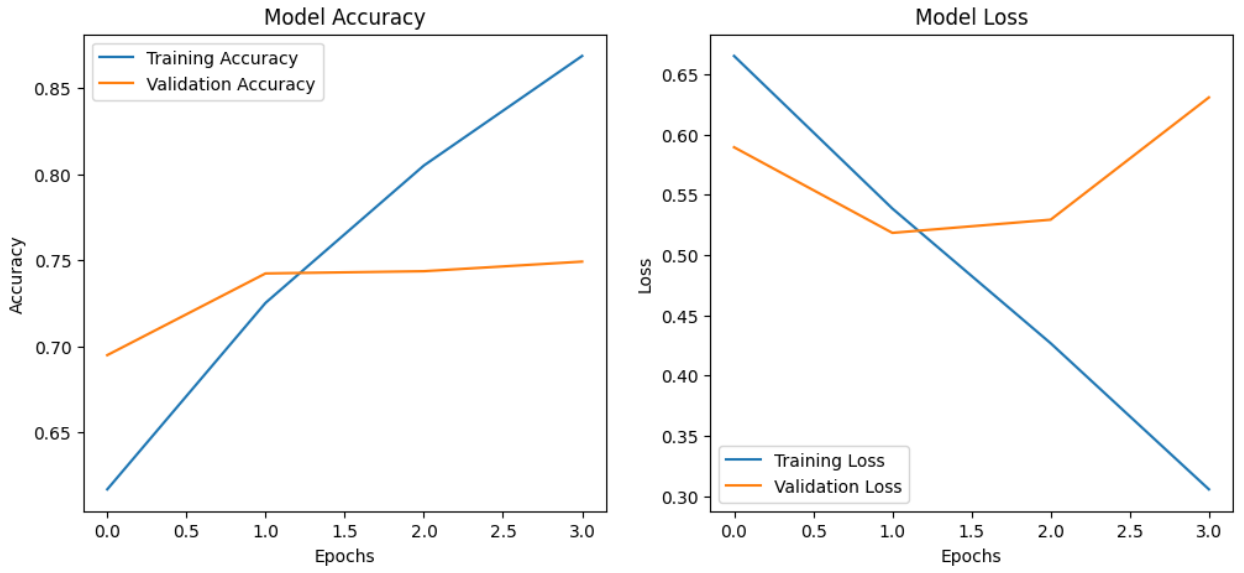
برای جلوگیری از overfitting از متود توقف EarlyStopping با patience=2 استفاده شده است.

حال به سراغ train کردن مدل می رویم و epoch را 5 قرار می دهیم.

```
Epoch 1/5
582/582 [==============================] - 1371s 2s/step - loss: 0.6652 - accuracy: 0.6167 - val_loss: 0.5894 - val_accuracy: 0.6948
Epoch 2/5
582/582 [==============================] - 1429s 2s/step - loss: 0.5386 - accuracy: 0.7251 - val_loss: 0.5184 - val_accuracy: 0.7423
Epoch 3/5
582/582 [==============================] - 1371s 2s/step - loss: 0.4268 - accuracy: 0.8050 - val_loss: 0.5293 - val_accuracy: 0.7436
Epoch 4/5
582/582 [==============================] - 1349s 2s/step - loss: 0.3055 - accuracy: 0.8687 - val_loss: 0.6308 - val_accuracy: 0.7491
```

سپس عملکرد مدل بر روی مجموعه داده validation ارزیابی می شود و منحنی‌های دقت و loss برای train و validation رسم و ذخیره می‌شوند.

```
146/146 [==============================] - 96s 653ms/step - loss: 0.5184 - accuracy: 0.7423
Validation accuracy: 0.7422614097595215
```



ب – ابتدا مانند قبل، با استفاده از تابع preprocess_image مقادیر پیکسل را عادی می کنیم و اندازه تصاویر را به ۲۹۹*۲۹۹ تغییر می دهیم که اندازه ورودی مورد نیاز InceptionV3 (قرار داده شده در کد) است.

برای نمایش تصاویر از تابع show_images استفاده می کنیم که چند تصویر از مجموعه داده های آموزشی را به همراه لیبل های آنها به تصویر می کشد.

معماری مدل:

مدل پایه InceptionV3 با وزن های freeze شده مطابق کد استتفاده شده است.

```
Model: "inception_v3"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_3 (InputLayer)        [(None, 299, 299, 3)]     0         []

 conv2d_188 (Conv2D)         (None, 149, 149, 32)      864       ['input_3[0][0]']

 batch_normalization_188 (B  (None, 149, 149, 32)      96        ['conv2d_188[0][0]']
 atchNormalization)

 activation_188 (Activation  (None, 149, 149, 32)      0         ['batch_normalization_188[0][0
 )                                                               ]']

 conv2d_189 (Conv2D)         (None, 147, 147, 32)      9216      ['activation_188[0][0]']

 batch_normalization_189 (B  (None, 147, 147, 32)      96        ['conv2d_189[0][0]']
 atchNormalization)

 activation_189 (Activation  (None, 147, 147, 32)      0         ['batch_normalization_189[0][0
 )                                                               ]']

 conv2d_190 (Conv2D)         (None, 147, 147, 64)      18432     ['activation_189[0][0]']

 batch_normalization_190 (B  (None, 147, 147, 64)      192       ['conv2d_190[0][0]']
...
Trainable params: 21768352 (83.04 MB)
Non-trainable params: 34432 (134.50 KB)
```

سپس لایه GlobalAveragePooling2D برای تبدیل خروجی ۴ بعدی به ۲ بعدی با میانگین هر نقشه ویژگی و در آخر یک لایه Dense با یک تابع فعال سازی سیگموئید برای طبقه بندی باینری اضافه شده است.

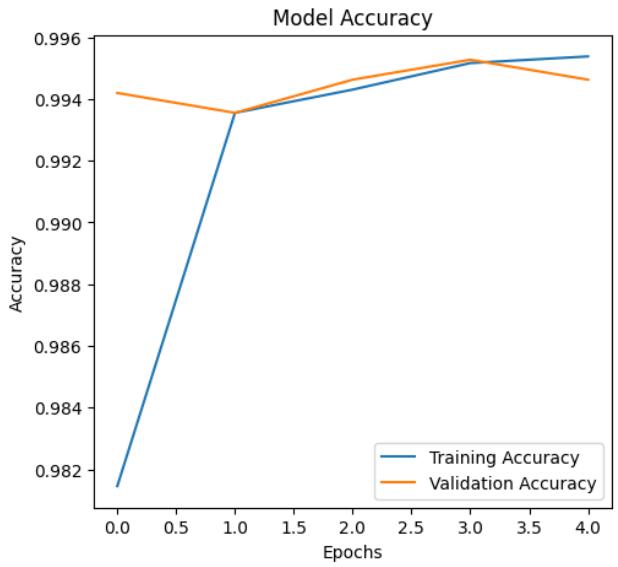
سپس مدل را مانند بخش قبل کامپایل کرده و برای train آن epoch را ۵ قرار می دهیم.

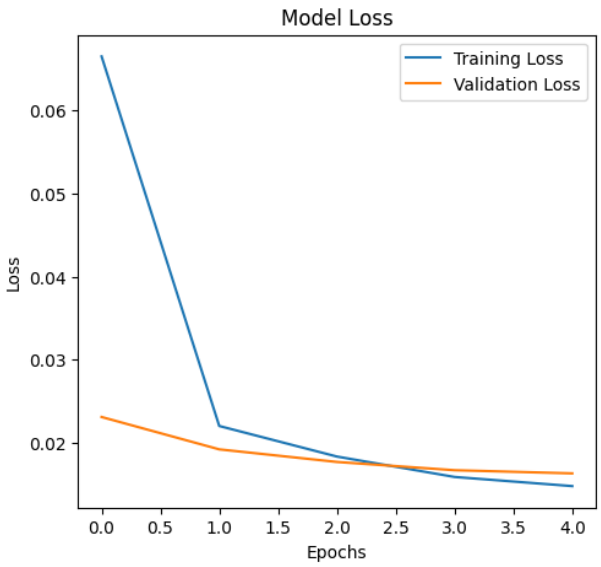
```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 8, 8, 2048)        21802784

 global_average_pooling2d_4  (None, 2048)              0
  (GlobalAveragePooling2D)

 dense_5 (Dense)             (None, 1)                 2049

=================================================================
Total params: 21804833 (83.18 MB)
Trainable params: 2049 (8.00 KB)
Non-trainable params: 21802784 (83.17 MB)
_________________________________________________________________
```

```
Epoch 1/5
582/582 [==============================] - 116s 174ms/step - loss: 0.0665 - accuracy: 0.9815 - val_loss: 0.0231 - val_accuracy: 0.9942
Epoch 2/5
582/582 [==============================] - 97s 163ms/step - loss: 0.0220 - accuracy: 0.9936 - val_loss: 0.0192 - val_accuracy: 0.9936
Epoch 3/5
582/582 [==============================] - 100s 169ms/step - loss: 0.0183 - accuracy: 0.9943 - val_loss: 0.0177 - val_accuracy: 0.9946
Epoch 4/5
582/582 [==============================] - 97s 163ms/step - loss: 0.0159 - accuracy: 0.9952 - val_loss: 0.0167 - val_accuracy: 0.9953
Epoch 5/5
582/582 [==============================] - 101s 169ms/step - loss: 0.0148 - accuracy: 0.9954 - val_loss: 0.0163 - val_accuracy: 0.9946
```

```
146/146 [==============================] - 18s 119ms/step - loss: 0.0163 - accuracy: 0.9946
Test accuracy: 0.994625985622406
```

در نهایت عملکرد مدل بر روی مجموعه تست ارزیابی می شود و دقت چاپ شده و دقت آموزش و اعتبارسنجی و منحنی‌های loss رسم می شود.





سوال ۷) ابتدا دیتاست MNIST را با استفاده ازmnist.load_data لود می کنیم و فقط تصاویر و برچسب های مربوط به ارقام ۰، ۱ و ۲ را استخراج می کنیم. سپس مقادیر پیکسل تصاویر را در محدوده [۰, ۱] نرمال می کنیم و تصاویر را به گونه ای تغییر شکل می دهیم که یک کانال داشته باشند (مقیاس خاکستری).

به عنوان یک کدل یادگیری ماشین برای دسته بندی از شبکه عصبی کانولوشن (CNN) استفاده کردم. مراحل :

- ما یک مدل ترتیبی را با استفاده از `Sequential()` از TensorFlow Keras تعریف می کنیم.
- سپس لایه های Conv2D و MaxPooling2D را اضافه می کنیم.
- از Flatten استفاده می کنیم تا برای ورودی به لایه های کاملا متصل آماده شود.

- سپس لایه و Dropout را اضافه می کنیم تا از overfitting جلوگیری کنیم.

- در نهایت لایه خروجی را با ۳ نورون (برای ۳ کلاس: ۰، ۱ و ۲) و تابع فعال سازی softmax به احتمالات کلاس خروجی اضافه می کنیم.

سپس مدل را مانند بخش های قبل کامپایل می کنیم.

بعد به سراغ آموزش مدل می رویم. ما مدل را بر روی داده های آموزشی ('X_train' و 'y_train') برای ۵ دوره با اندازه دسته ای ۱۲۸ آموزش می دهیم. همچنین داده های validation ('X_val' و 'y_val') را برای نظارت بر عملکرد مدل در طول آموزش مشخص می کنیم.

در آخر دقت را اندازه گیری می کنیم.

```
99/99 [==============================] - 1s 8ms/step - loss: 0.0051 - accuracy: 0.9975
Test accuracy: 0.9974579215049744
```

برای نمایش برخی از تصاویر از مجموعه آزمایشی، فانکشن display_predictions پیاده سازی شده است.

خروجی :