پاسخ تمرین چهارم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

سوال اول)

۱.الف طراحی شبکه هم گشتی با استفاده از Keras Tuner:

بخش اول)

Keras Tuner یک ابزار است که برای جستجوی بهینه‌ترین پارامترهای مدل‌های عصبی از جمله نوع لایه‌ها، اندازه لایه‌ها، نرخ یادگیری، و سایر پارامترهای مربوط به شبکه‌های عصبی استفاده می‌شود. این ابزار به شما کمک می‌کند تا به سرعت و بهینه‌ترین معماری و پارامترهای مدل خود را از بین گزینه‌های مختلف انتخاب کنید.

ویژگی‌های کلیدی Keras Tuner عبارتند از:

1. **پشتیبانی از چندین الگوریتم جستجو** Keras Tuner :از الگوریتم‌های جستجوی متنوعی مانندRandom Search ، Hyperband، Bayesian Optimizationو Grid Search پشتیبانی می‌کند. این الگوریتم‌ها به شما کمک می‌کنند تا پارامترهای بهینه برای مدل خود را بر اساس استراتژی‌های مختلف جستجویی پیدا کنید.
2. **سادگی استفاده :**واسط کاربری ساده و کتابخانه‌های Python قابل استفاده از Keras Tuner را برای تعریف فضای پارامتر و اجرای جستجوی بهینه‌سازی فراهم می‌کند.
3. **انعطاف پذیری در انتخاب پارامترها :**شما می‌توانید از Keras Tuner برای جستجوی بهینه‌ترین معماری شبکه‌های عصبی، هایپرپارامترها، و سایر تنظیمات شبکه‌ها استفاده کنید.
4. **یکپارچگی با TensorFlow و Keras** : Keras Tuner با TensorFlow و Keras یکپارچه شده است و از آنها به عنوان ابزار اصلی برای تعریف و ساخت مدل‌های عصبی استفاده می‌کند.

استفاده از Keras Tuner به شما کمک می‌کند تا فرآیند تنظیم هایپرپارامترهای مدل‌های عصبی خود را بهبود بخشیده و به سرعت به معماری‌ها و تنظیماتی برسید که عملکرد بهتری در وظایف مورد نظر شما دارند.

بخش دوم)

برای استفاده از Keras Tuner برای بهینه‌سازی شبکه‌های عصبی جهت دسته‌بندی داده‌ها، شما می‌توانید مراحل زیر را دنبال کنید:

1. **تعریف فضای هایپرپارامترها:**
   - ابتدا باید فضای هایپرپارامترها را تعریف کنید. این شامل نوع لایه‌ها، تعداد لایه‌ها، اندازه لایه‌ها، نرخ یادگیری، توابع فعال‌سازی و دیگر پارامترهای مربوط به شبکه عصبی است.
2. **تعریف تابع برای ارزیابی مدل:**
   - باید یک تابعی تعریف کنید که مدل را ایجاد کرده و آن را با داده‌های آموزشی شما آموزش دهد و سپس از داده‌های ارزیابی استفاده کند تا عملکرد مدل را ارزیابی کند.
3. **استفاده از Keras Tuner برای جستجوی بهینه:**
   - بعد از تعریف فضای هایپرپارامترها و تابع ارزیابی، از کلاس‌های Keras Tuner برای جستجوی بهینه استفاده می‌کنید. می‌توانید یک یا چند الگوریتم جستجو (مانند **RandomSearch**یا **Hyperband**) را انتخاب کرده و آن را برای بهینه‌سازی هایپرپارامترهای شبکه‌ی عصبی‌تان استفاده کنید.
4. **تعیین بهترین مدل و آموزش با داده‌های کلیه:**
   - پس از جستجو و یافتن بهترین مدل و تنظیمات، می‌توانید مدل را با داده‌های آموزشی و ارزیابی کلیه آموزش دهید تا در نهایت مدل بهترین پارامترها را داشته باشد.

بخش سوم)

Keras Tuner انواع مختلفی از Tuner ها را برای جستجوی بهینه‌ی هایپرپارامترها ارائه می‌دهد. برخی از Tuner های اصلی به شرح زیر هستند:

1. **RandomSearch:**

- این تنظیم تصادفی ترکیب‌های هایپرپارامتر را امتحان می‌کند. این یک روش جستجوی سریع و پیاده‌سازی آسان است. این مناسب برای شروع و بررسی اولیه‌ی مدل‌هاست.

2. **Hyperband:**
- این یک الگوریتم حذف تنظیماتی است که عملکرد نسبتاً بدتر دارند و به سرعت به نقطه بهینه‌ی موجود می‌رسد. این کارایی خوبی برای دیتاست‌های بزرگ دارد.

3. **Bayesian Optimization:**
- این الگوریتم از تکنیک‌های بهینه‌سازی بیزین استفاده می‌کند تا به طور هوشمندانه ترکیب‌های هایپرپارامترها را بررسی کند. این الگوریتم معمولاً مناسب برای جستجوی دقیق‌تر و بهینه‌سازی مدل‌هاست.

با توجه به سؤال شما درباره دیتاست MNIST برای طبقه‌بندی، زمانی که دیتاست کوچک است و ابعاد پیچیدگی کمی دارد، ممکن است RandomSearch مناسب باشد زیرا این دیتاست رویه‌های پیچیده و هایپرپارامترهای زیادی ندارد. اما اگر تعداد هایپرپارامترها زیاد است و نیاز به جستجوی دقیق تری دارید، الگوریتم Bayesian Optimization می‌تواند گزینه‌ی مناسبی باشد.
انتخاب Tuner مناسب برای دیتاست خاص به ابعاد مدل و موارد مشابه بستگی دارد.

۱.ب استفاده از Keras Tuner برروی دیتاست mnist:

بخش اول)

این دیتاست از 70000 عکس تشکیل شده که شامل اعداد نوشته شده توسط دانش آموزان دبیرستانی و کارمندان سازمان سرشماری آمریکا هست. هر عکس با شماره ای که توی عکس هست لیبل گذاری شده. این دیتاست انقدر مطالعه شده که بعضی اوقات بهش میگن " hello world" ماشین لرنینگ!

دیتاست MNIST یکی از معروف‌ترین و پایه‌ای‌ترین دیتاست‌های بینایی ماشین است که برای تشخیص دست‌نوشته‌های رقمی (digits) به کار می‌رود. این دیتاست شامل 70,000 تصویر دست‌نوشته از ارقام 0 تا 9 است که در 2 دسته آموزش (60.000 عدد) و آزمون(10.000 عدد) قرار دارند. هر تصویر در این دیتاست، دارای ابعاد 28 × 28 پیکسل است.

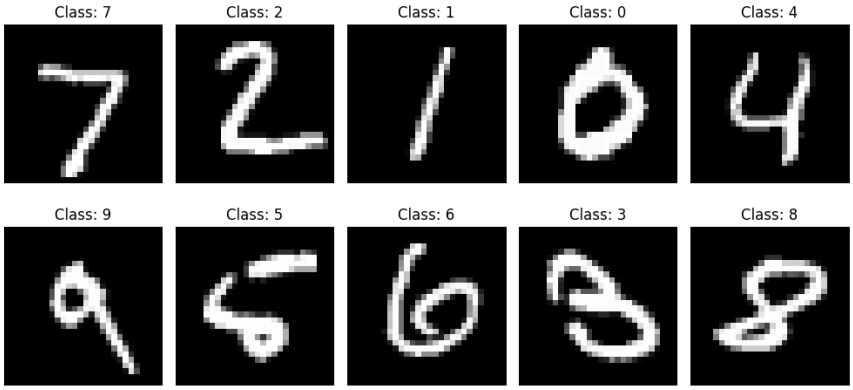
MNISTهمچنین در یادگیری عمیق (Deep Learning) به عنوان یک دیتاست معروف شناخته میشود. سال‌هاست که این دیتاست برای تست الگوریتم‌های جدید ارائه می‌شود و نقطه شروع خوبی برای کسانی است که با ساختار داده‌های تصویری و یادگیری عمیق آشنا شده‌اند.
برای استفاده از این دیتاست، معمولاً از پایتورچ (PyTorch) یا تنسورفلو (TensorFlow) به عنوان کتابخانه محاسباتی استفاده می‌شود.
همچنین، برای اولین بار در سال ۱۹۸۹ معرفی شده و در حال حاضر طیف گسترده‌ای از الگوریتم‌های یادگیری ماشین بر روی این دیتاست تست شده است.

یکی از کاربردهای این دیتاست، تشخیص رقم در سامانه‌های پردازش تصویر است. همچنین، این دیتاست به عنوان یک مثال بسیار ساده و مقرون به صرفه برای آموزش الگوریتم‌های شبکه‌های عصبی در یادگیری عمیق مورد استفاده قرار می‌گیرد.

دیتاست MNISTیک دیتاست اعداد دستنویس است .دیتاست MNIST که گاهی دیتابیس امنیست مخفف Modified National Standards and Technology Database گفته می‌شود، زیرمجموعه اصلاح شده‌ای از دیتاست NIST است. یان لکان (Yann LeCun) و کورینا کورتس (Corinna Cortes) و کریستوفر برجس (Christopher Burges) با انتخاب و ترکیب بخشی از NIST این دیتاست را در سال ۱۹۹۸ رسما معرفی کردند.

هر عکس انتخاب شده از NIST طی دو مرحله یک بار در باکس‌ها ۲۰ در ۲۰ پیکسل مرکزیت داده شد و سپس همین باکس‌ها دوباره در باکس‌های ۲۸ در ۲۸ جای گرفتند. همچنین در مرحله دوم، عدد میان هر عکس با محاسبه مرکز ثقل در میانه عکس قرار گرفت.

در سال‌های بعد پژوهش‌های دیگری برای محک الگوریتم‌های گوناگون یادگیری ماشین و بینایی کامپیوتر روی این دیتاست انجام و منتشر شد. ریز این پژوهش‌های در صفحه رسمی دیتاست امنیست (+) آورده شده است. همچنین آقای ژوزف ردمون (Joseph Redmon) یک نسخه CSV از MNIST را در این صفحه (+) منتشر کردند که کار با آن برای آموزش شبکه‌های عصبی ساده‌تر است.



بخش دوم)

برای استفاده از یک شبکه عصبی بهینه‌سازی شده با Keras Tuner برای دسته‌بندی تصاویر دیتاستMNIST ، شما باید ابتدا مدلی را با استفاده از Keras Tuner بهینه‌سازی کرده و سپس آن را برای آموزش و ارزیابی با دیتاست MNIST استفاده کنید. مراحل زیر نشان می‌دهند چگونگی این کار:

1. **تعریف فضای هایپرپارامترها با استفاده از:Keras Tuner**
   - ابتدا باید فضای هایپرپارامترها را برای شبکه عصبی خود با استفاده از Keras Tuner تعریف کنید. این شامل تعیین تعداد لایه‌ها، اندازه لایه‌ها، نوع بهینه‌ساز، نرخ یادگیری و سایر پارامترهای مدل است.
2. **ساخت مدل با استفاده از پارامترهای بهینه‌سازی شده:**
   - سپس باید مدل را بر اساس بهینه‌سازی‌های دریافت شده از Keras Tuner ایجاد کنید. این شامل استفاده از تنظیمات بهینه‌سازی‌شده برای لایه‌ها، تعداد نورون‌ها، توابع فعال‌سازی و سایر تنظیمات مربوط به مدل است.
3. **آموزش مدل با داده‌های:MNIST**
   - حالا که مدل را ساختید، آن را با داده‌های MNIST آموزش دهید. برای این کار، از توابع آموزشی و ارزیابی TensorFlowیا Keras استفاده کنید.

بخش سوم)

1. **Dropout:**
   - Dropoutیک روش رگولاریزیشن است که در جلوگیری از بیش‌برازش (overfitting) موثر است. در هنگام آموزش شبکه، Dropoutتصادفی برخی از واحدهای نورونی را با احتمال خاصی خاموش می‌کند، به این ترتیب از یادگیری وابستگی بیش از حد به نمونه‌های آموزشی جلوگیری می‌کند و امکان یادگیری ویژگی‌های کلی و کمتر وابسته به داده‌های خاص را فراهم می‌کند. این باعث می‌شود که شبکه‌ی عصبی عمومی‌تری را یاد بگیرد و به طور معمول منجر به عملکرد بهتر در داده‌های تست واقعی شود.
2. **Pooling:**
   - Poolingلایه‌ای در شبکه‌های عصبی است که با کاهش ابعاد فضایی وزن‌ها، تعداد پارامترها و محاسبات در شبکه، بهبود در کارایی و کاهش پیچیدگی مدل را فراهم می‌کند .Pooling معمولاً با انجام عملیاتی مانند Max Pooling یا Average Poolingانجام می‌شود که به ترتیب بیشینه یا میانگین مقادیر در نواحی مشخصی از ورودی را استخراج می‌کنند. این عمل باعث کاهش ابعاد داده‌ها و افزایش توانایی شبکه در تشخیص ویژگی‌های مهم و عملکرد بهتر در مسائل دسته‌بندی می‌شود.

۱.پ پیاده سازی مدل:

بخش اول)

کد مربوطه در فایل Q1.ipynb آورده شده است.

```
tuner.search(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

# Get the best model
best_model = tuner.get_best_models(num_models=1)[0]
best_hyperparameters = tuner.get_best_hyperparameters(num_trials=1)[0]

# Summary of the best model
best_model.summary()
```

```
Trial 1 Complete [00h 00m 00s]

Best val_accuracy So Far: None
Total elapsed time: 00h 00m 00s

Search: Running Trial #2

Value             |Best Value So Far |Hyperparameter
3                 |4                 |num_conv_layers
160               |96                |conv_0_filters
3                 |1                 |num_dense_layers
224               |256               |dense_0_neurons
0.001             |0.001             |learning_rate
160               |32                |conv_1_filters
128               |32                |conv_2_filters
256               |32                |conv_3_filters

Epoch 1/5
1090/1875 [================>.............] - ETA: 2:15 - loss: 0.3006 - accuracy: 0.9044
```

این نتایج برای هایپرپارامترهای رندوم در ایپاک اول تریال دوم است.

```
Trial 3 Complete [00h 14m 43s]
val_accuracy: 0.9912999868392944

Best val_accuracy So Far: 0.9912999868392944
Total elapsed time: 00h 43m 08s

Search: Running Trial #4

Value             |Best Value So Far |Hyperparameter
2                 |2                 |num_conv_layers
224               |96                |conv_0_filters
1                 |5                 |num_dense_layers
32                |160               |dense_0_neurons
0.001             |0.001             |learning_rate
192               |128               |conv_1_filters
192               |32                |conv_2_filters
128               |64                |conv_3_filters
96                |160               |dense_1_neurons
128               |96                |dense_2_neurons
128               |32                |dense_3_neurons
192               |32                |dense_4_neurons

Epoch 1/5
1875/1875 [==============================] - 462s 246ms/step - loss: 0.1250 - accuracy: 0.9614 - val_loss: 0.0462 - val_accuracy: 0.9854
Epoch 2/5
1875/1875 [==============================] - 461s 246ms/step - loss: 0.0424 - accuracy: 0.9866 - val_loss: 0.0387 - val_accuracy: 0.9862
Epoch 3/5
1875/1875 [==============================] - 461s 246ms/step - loss: 0.0284 - accuracy: 0.9910 - val_loss: 0.0333 - val_accuracy: 0.9890
Epoch 4/5
 491/1875 [======>.......................] - ETA: 5:25 - loss: 0.0172 - accuracy: 0.9939
```

این نتایج برای هایپرپارامترهای رندوم در تریال چهارم است.

```
Trial 5 Complete [01h 01m 29s]
val_accuracy: 0.9915000200271606

Best val_accuracy So Far: 0.9915000200271606
Total elapsed time: 02h 24m 00s
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 256)       2560

 max_pooling2d (MaxPooling2  (None, 13, 13, 256)       0
 D)

 conv2d_1 (Conv2D)           (None, 11, 11, 256)       590080

 max_pooling2d_1 (MaxPoolin  (None, 5, 5, 256)         0
 g2D)

 flatten (Flatten)           (None, 6400)              0

 dense (Dense)               (None, 64)                409664

 dense_1 (Dense)             (None, 224)               14560

 dense_2 (Dense)             (None, 96)                21600

 dense_3 (Dense)             (None, 192)               18624

 dense_4 (Dense)             (None, 64)                12352

 dense_5 (Dense)             (None, 10)                650

=================================================================
Total params: 1070090 (4.08 MB)
Trainable params: 1070090 (4.08 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

در این تصویر مدل انتخاب شده که بهترین نتایج را با دقت 90 درصد داشته نشان میدهید.

این مدل دو لایه کانولوشنی و دو مکس پولینگ و 5 لایه فولی کانکتد دارد. تعداد پارامترها و خروجی ها مشخص شده است.

فیلتر ها همگی 3در3 هستند.

```
Number of Filters for Each Convolutional Layer:
================================================
Convolutional Layer 1: 256 filters
Convolutional Layer 2: 256 filters

Number of Neurons for Each Dense Layer:
=======================================
Dense Layer 1: 64 neurons
Dense Layer 2: 224 neurons
Dense Layer 3: 96 neurons
Dense Layer 4: 192 neurons
Dense Layer 5: 64 neurons
Learning Rate: 0.001
```

پاسخ تمرین چهارم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

بخش دوم)

انتخاب اندازه فیلترها (همچنین به عنوان کرنل یا هسته شناخته می‌شوند) در لایه‌های Convolutional یک مسئله مهم در طراحی شبکه‌های عصبی عمیق است. اندازه فیلترها تاثیر زیادی بر عملکرد و عملکرد شبکه دارد و انتخاب درست آن می‌تواند به دقت و کارایی شبکه کمک کند. اینجا چند نکته مهم درباره انتخاب اندازه فیلترها در لایه‌های Convolutional وجود دارد:

1. **ابعاد فیلترها:**
   - ابعاد یک فیلتر شامل ارتفاع و عرض است که به طور معمول در مثال‌های Convolutional به صورت (ارتفاع، عرض) یا (بلندای فیلتر، عرض فیلتر) نمایش داده می‌شود (مانند (3، 3) یا (5، 5)) .این ابعاد تعیین می‌کنند که هر فیلتر چه تعداد از ویژگی‌های ورودی را پوشش می‌دهد.
2. **تأثیر اندازه فیلترها بر ویژگی‌های استخراج شده:**
   - استفاده از فیلترهای کوچک‌تر می‌تواند به شبکه کمک کند تا جزئیات دقیق‌تری از تصویر را استخراج کند، در حالی که فیلترهای بزرگ‌تر ممکن است ویژگی‌های بزرگتر و انتزاعی‌تری را شناسایی کنند. انتخاب اندازه فیلترها بسته به وظیفه و داده‌های مورد استفاده متفاوت است.
3. **پوشش (Stride) و پردازش‌های دیگر:**
   - انتخاب stride و padding نیز تأثیر زیادی بر انتخاب اندازه فیلترها دارد Stride .مشخص می‌کند چقدر از تصویر ورودی با یک حرکت از فیلتر پوشش داده می‌شود. اگر Stride بزرگ باشد، اندازه فیلتر ممکن است کوچکتر باشد زیرا تصویر خروجی کوچکتر خواهد بود.
4. **عمق شبکه:**
   - در لایه‌های اول شبکه، معمولاً از فیلترهای کوچک‌تر شروع می‌کنند و به مرور به فیلترهای بزرگ‌تر در لایه‌های عمیق‌تر می‌پردازند. این انتخاب عمدتاً بر اساس پردازش‌های اولیه و استخراج ویژگی‌های سطح پایین تا سطح بالا در شبکه است.
5. **تعداد فیلترها:**
   - تعداد فیلترهای موجود در هر لایه نیز اهمیت دارد. استفاده از تعداد فیلترهای بیشتر می‌تواند به شبکه کمک کند تا ویژگی‌های مختلف‌تر و پیچیده‌تر را استخراج کند

معمولا هرچه سایز فیلتر کوچکتر باشد فیچرهای سطح پایین تر مثل لبه را تشخیص میدهد و فیلترهای بزرگتر میتوانند ویژگی های مثل وجود دایره یا نوع خاصی از اشکال را در تصاویر پیدا کنند. معمولا برای جلوگیری از محاسبات زیاد به جای استفاده از یک لایه کانولوشنی با فیترهای بزرگ از چند لایه کانولوشنی با فیلترهای کوچکتر استفاده میشود. در اینجا مقدار 3 را درنظر گرفتیم.

بخش سوم)

1. **Pooling:**
   - لایه‌های Pooling مانند Max Pooling یا Average Pooling که بعد از لایه‌های Convolutional قرار می‌گیرند، به کاهش ابعاد فضایی داده‌ها و افزایش توانایی شبکه در تشخیص ویژگی‌های مهم کمک می‌کنند. این عمل باعث کاهش ابعاد ویژگی‌ها می‌شود و در نتیجه تعداد پارامترها کاهش می‌یابد. این می‌تواند از بیش‌برازش جلوگیری کرده و باعث بهبود عملکرد شبکه در داده‌های آزمون می‌شود.
2. **Dropout:**
   - Dropoutیک روش رگولاریزیشن است که با تصادفی کردن خاموش کردن برخی از واحدهای نورونی در طول آموزش، از وابستگی بیش از حد به نمونه‌های آموزشی جلوگیری می‌کند. این باعث می‌شود که شبکه عصبی انعطاف پذیرتری داشته باشد و ویژگی‌های کلی‌تری از داده‌ها را یاد بگیرد. استفاده از Dropout می‌تواند بهبود دقت و کاهش اثر بیش‌برازش در شبکه کمک کند.

منبع :

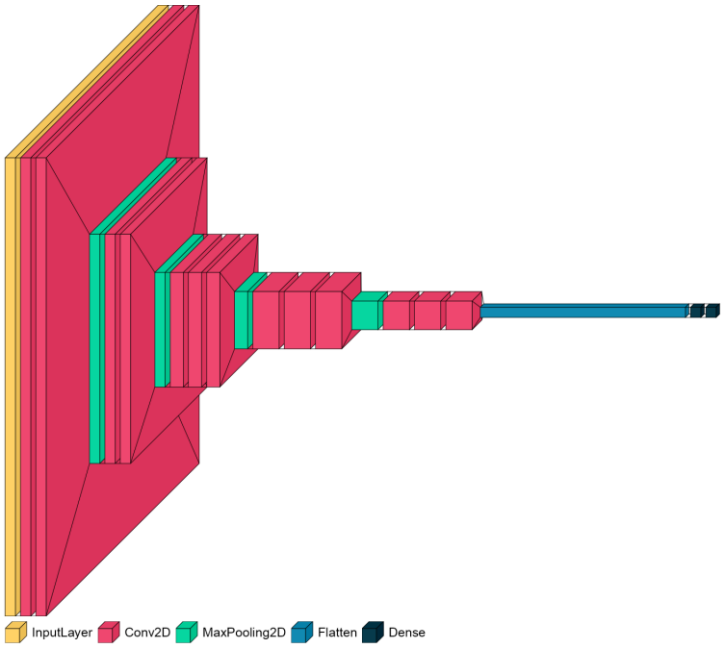
https://keras.io/keras_tuner/
https://www.tensorflow.org/tutorials/keras/keras_tuner
https://hamruyesh.com/what-is-a-mnist-database/

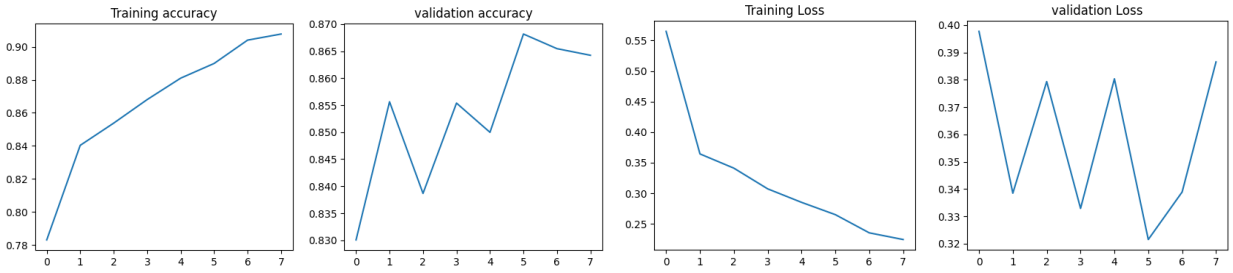
https://class.vision/blog/%D8%AF%DB%8C%D8%AA%D8%A7%D8%B3%D8%AA-mnist/

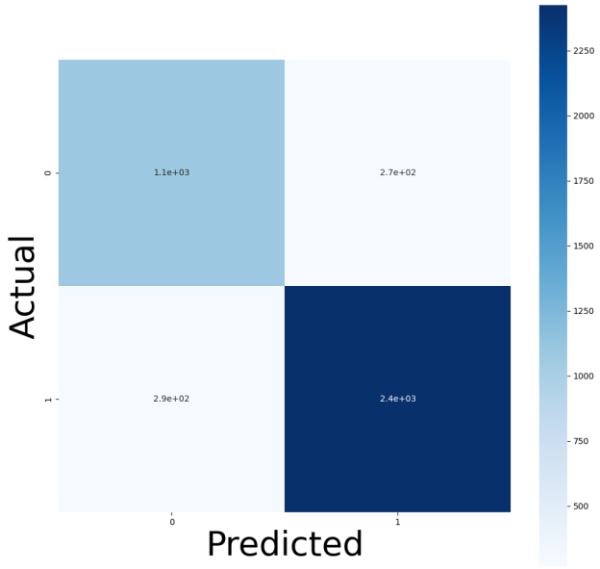
سوال دوم:

کدهای فایل medical.ipynb کامل شده است.

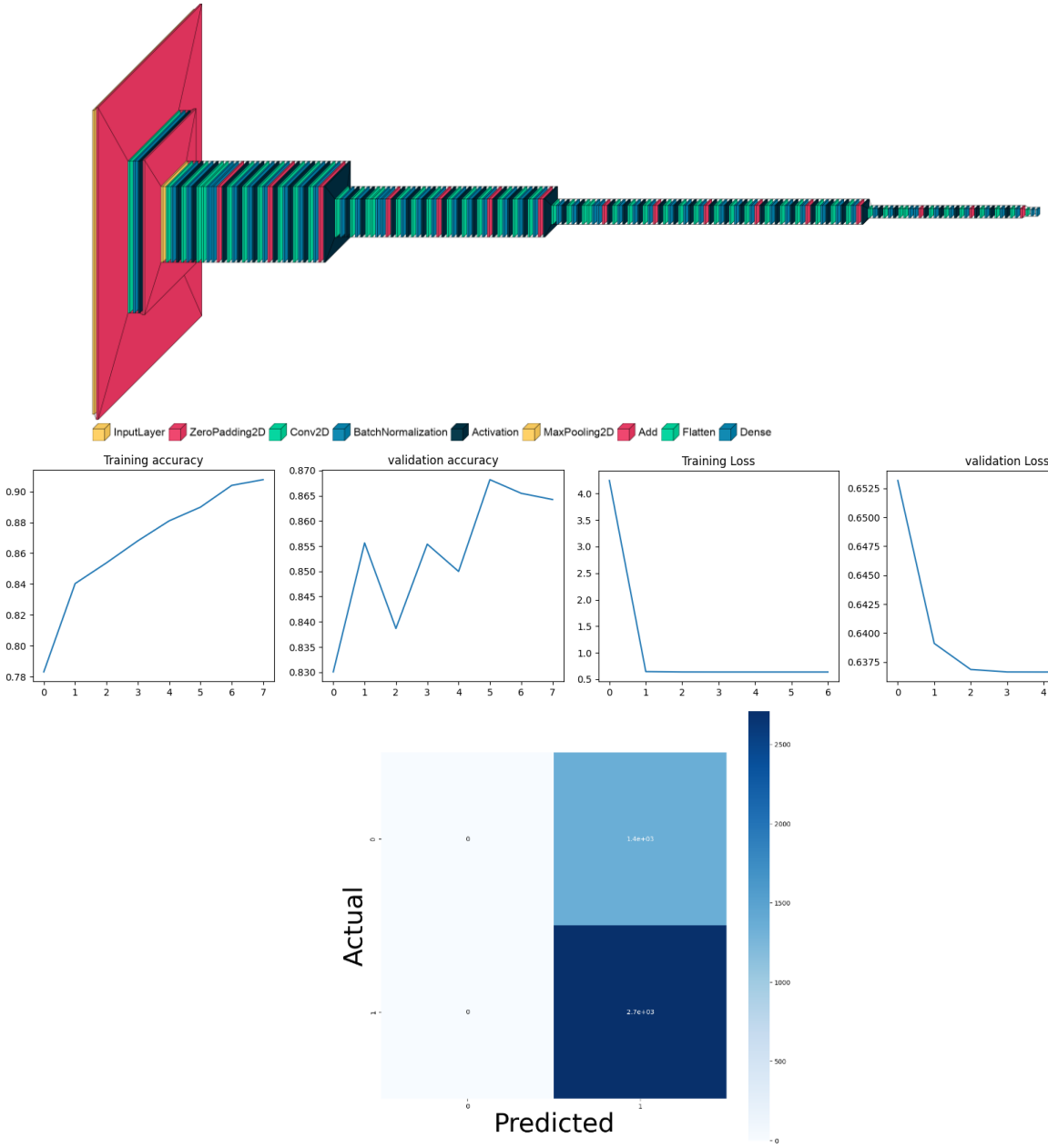
نتایج vgg به صورت زیر شد:







نتایج resnet به صورت زیر شد:



نتایج کلی:

model	number of train data	number of test data	train accuracy	test accuracy	time	parametes
VGG	16261	4066	0.8893057107925415	0.8615346550941467	510.1060461997986	17926338
ResNet	16261	4066	0.7109034061431885	0.695032000541687	895.9931626319885	36433154

سوال سوم:

مشکل اصلی این کد نوع استفاده از دیتا و عدم پیش پردازش روی آن است.

date	time (hour)	year	cons (m3/unit)	month	day	holiday	T air©
2023-02-03	12	2023	0.525037	3	6	0	18.2
2023-02-03	13	2023	0.487244	3	6	0	18.0
2023-02-03	14	2023	0.472253	3	6	0	17.0
2023-02-03	15	2023	0.469390	3	6	0	15.8
2023-02-03	16	2023	0.481152	3	6	0	12.0

فیچرهایی که قرار است خروجی یا میزان مصرف از آنها پیشبینی شود، جنس متفاوتی دارند و از همین رو کاملا scale متفاوتی دارند. مثلا چند دهم افزایش یا کاهش دما میتواند روی خروجی تاثر بگذارد، درصورتی که تغییرات ساعت یک واحد کامل هستند. برای حل مشکل باید داده ها نرمال سازی شوند تا بتوان بهینه سازی تابع هزینه را درست انجام داد همانطر که میتوان دید تابع هزینه در طول آموزش خیلی تغییر نکرده است چون مدام در حال نوسان بوده و سرعت کاهش آن بسیار کم بوده است..

روش های نرمال سازی معمولا به این صورت است که میانگین دیتا ها را به 0 و واریانس را به 1 تبدیل کند.

استفاده از کد زیر میتواند مناسب باشد:

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

یکی دیگر از مشکلات این است که ما از 6 فیچر برای پیشبینی استفاده کردیم در صورتی که تعدادی از آنها خیلی مهم نیستند. مثلا دمای هوا بیشترین تاثیر را روی میزان مصرف گاز دارد ممکن است فکر کنید این که در چه فصلی از سال هستیم هم موثر است که خب چون دمای آن ثبت شده واقعا احتیاجی نیست به ماه و سال به عنوان فیچر جدا فکر کنیم و میتوانیم فقط دما را به عنوان ورودی در نظر بگیریم.

سوال چهارم:

۴.الف:

شبکه‌های همگشتی (Convolutional Neural Networks - CNNs) و شبکه‌های بازگشتی (Recurrent Neural Networks - RNNs)دو نوع از معماری‌های شبکه‌های عصبی عمیق هستند، هر کدام ویژگی‌ها و کاربردهای خاص خود را دارند.

شبکه‌های همگشتی:(CNNs)

- **ویژگی‌ها:**
  - مناسب برای تصویر و شناسایی الگوها در داده‌های سه‌بعدی مانند تصاویر.
  - از لایه‌های کانولوشنی و لایه‌های پولینگ (Pooling) برای استخراج ویژگی‌ها از تصاویر استفاده می‌کند.
  - قابلیت یادگیری ویژگی‌های سطح بالا را دارد.
  - مناسب تسک هایی که طول خروجی و ورودی در آنها ثابت است.
  - ویژگی های محلی را خیلی خوب پیدا میکنند.
- **کاربردها:**
  - دسته‌بندی تصاویر: تشخیص الگوها و اشیاء در تصاویر.
  - تشخیص اشیاء و شیوهای مختلف در پردازش تصویر.

- بازشناسی صورت و تصویر، ترجمه ماشینی و خودروهای خودران.

شبکه‌های بازگشتی:(RNNs)

- **ویژگی‌ها:**
  - مناسب برای داده‌های دنباله‌ای و زمانی، مانند متون و داده‌های زمانی.
  - قابلیت حفظ اطلاعات قبلی و اعمال تاثیر زمانی.
  - از لایه‌های بازگشتی برای مدل‌سازی وابستگی‌های زمانی استفاده می‌کند.
  - مناسب تسک هایی که در آنها طول ورودی و خروجی متغیر است.
  - ویژگی های گذشته را حفظ میکند و برای پیشبینی آینده در کنار ویژگی محلی استفاده میکند.
- **کاربردها:**
  - ترجمه ماشینی، تولید متن و شبیه‌سازی گفتار.
  - پیش‌بینی داده‌های زمانی مانند سری‌های زمانی، تحلیل متن و استفاده در داده‌های مرتبط با زمان.
  - تولید توالی‌های مختلف مانند موسیقی، مدل‌سازی زبان طبیعی و تحلیل متون.
  - مناسب سیگنال های دنباله ای

کاربرد مسائل:

- **CNNs**عمدتاً برای داده‌های ساختارمند با ابعاد بالا مانند تصاویر و ویدیوها مناسب هستند. این شبکه‌ها برای استخراج ویژگی‌های مکانی از داده‌ها کارآمد هستند.
- **RNNs**به خوبی برای دنباله‌های داده و اطلاعات وابسته به زمان مانند متون، موسیقی، سری زمانی و وظایف مشابه دیگر مناسبند. آن‌ها می‌توانند با وابستگی‌های زمانی کار کنند و از اطلاعات گذشته برای پیش‌بینی‌ها استفاده کنند.

۴.ب :

۱ .تعداد پارامترها:

- **CNNs**:
  - پارامترها در CNNs شامل وزن‌های مربوط به فیلترهای کانولوشن، بایاس‌ها، وزن‌های لایه‌های کاملاً متصل (Fully Connected) و ... می‌شود.
  - تعداد پارامترها در CNNs معمولاً کمتر از RNNs است زیرا CNNs بیشتر وزن‌های مشترک را به اشتراک می‌گذارند و با استفاده از فیلترها و عملکرد پولینگ ویژگی‌های مکانی مهم را فشرده‌تر می‌کنند.
- **RNNs**:
  - در RNNs پارامترها شامل وزن‌هایی است که هر واحد (یا سلول) در هر لایه با لایه قبلی دارد.

- به دلیل وجود وابستگی‌های زمانی و اعمال تأثیر وزن‌های هر لایه بر وضعیت لایه‌های قبلی، تعداد پارامترها در RNNs ممکن است بسیار بیشتر از CNNs باشد.

تعداد پارامترهای یک شبکه‌ی عصبی بازگشتی (RNN) به طور کلی متغیر است و بستگی به معماری خاص شبکه، تعداد لایه‌ها، نوع سلول بازگشتی و تعداد واحدها دارد. اما فرمول کلی برای محاسبه تعداد پارامترهای یک لایه‌ی RNN به صورت زیر است:

فرمول محاسبه تعداد پارامترهای یک لایه‌ی:RNN

تعداد پارامترها=(تعداد ورودی‌ها×تعداد واحدها)+(تعداد واحدها×تعداد واحدها)+(تعداد واحدها)

در این فرمول:

تعداد ورودی‌ها: تعداد ویژگی‌های ورودی یا ابعاد ورودی به شبکه.RNN

تعداد واحدها: تعداد واحدهای سلول بازگشتی) مثلاً تعداد واحدهای LSTM یا.(GRU

اولین جمعی که ضرب ورودی‌ها و تعداد واحدها را محاسبه می‌کند نشان دهنده وزن‌های ورودی به واحدها است.

دومین جمعی که ضرب تعداد واحدها در تعداد واحدها را محاسبه می‌کند، نشان‌دهنده وزن‌های بازگشتی بین واحدها است.

جمع سوم نیز تعداد پارامترهای بایاس (مثلاً بایاس واحدها) را نشان می‌دهد.

این فرمول برای یک لایه در شبکه RNN محاسبه شده است و اگر شبکه شما شامل چندین لایه است، بایستی تعداد پارامترهای هر لایه را با یکدیگر جمع کرد.

تعداد پارامترهای یک شبکه‌ی عصبی کانولوشنی (CNN) به صورت کلی بستگی به معماری شبکه، تعداد لایه‌ها، اندازه فیلترها، تعداد فیلترها و سایر پارامترهای موجود دارد. اما فرمول محاسبه تعداد پارامترهای یک لایه از CNN به صورت زیر است:

= تعداد فیلترهاتعداد پارامترها×(بایاس+تعداد ویژگی‌های ورودی×فیلتر اندازه×اندازه فیلتر) = تعداد پارامترها
تعداد فیلترها×(بایاس+تعداد ویژگی‌های ورودی×اندازه فیلتر×اندازه فیلتر)

در این فرمول:

- اندازه فیلتر: ابعاد فیلتر کانولوشنی، به عبارت دیگر تعداد ردیف‌ها و ستون‌های فیلتر.
- تعداد ویژگی‌های ورودی: تعداد کانال‌های ویژگی در ورودی
- بایاس: یک پارامتر بایاس برای هر فیلتر.
- تعداد فیلترها: تعداد فیلترهای استفاده شده در لایه مورد نظر.

برای محاسبه تعداد پارامترهای یک لایه ازCNN ، ابتدا تعداد پارامترهای مربوط به یک فیلتر (شامل وزن‌های فیلتر و بایاس) محاسبه می‌شود و سپس این مقدار بر تعداد فیلترها ضرب می‌شود.

این فرمول فقط تعداد پارامترهای یک لایه از CNN را محاسبه می‌کند و اگر شبکه شما شامل چندین لایه است، بایستی تعداد پارامترهای هر لایه را با یکدیگر جمع کرد.

پاسخ تمرین چهارم **مهدیه نادری: ۹۸۵۲۲۰۷۶**

پارامترهای هر یک از این شبکه‌ها به طور کلی می‌توانند شامل پارامترهای زیر باشند:

شبکه‌های عصبی بازگشتی

1. **تعداد لایه‌ها :(Number of Layers)** تعداد لایه‌های شبکه بازگشتی.
2. **تعداد گره‌ها :(Number of Units)** تعداد واحدهای هر لایه در شبکه بازگشتی.
3. **نوع بازگشت :(Type of Recurrence)** نوع سلول بازگشتی مانندLSTM ، GRUیا سلول بازگشتی ساده.
4. **پارامترهای وزن :(Weight Parameters)** شامل وزن‌های ورودی به واحدهای شبکه و وزن‌های بازگشتی برای حالت پیشین.

شبکه‌های عصبی کانولوشنی

1. **تعداد لایه‌ها :(Number of Layers)** تعداد لایه‌های شبکه کانولوشنی.
2. **اندازه فیلترها :(Filter Size)** ابعاد و اندازه فیلترهای کانولوشنی استفاده شده.
3. **تعداد فیلترها :(Number of Filters)** تعداد فیلترهای مورد استفاده در هر لایه.
4. **مرزها یا پادگان‌ها :(Paddings)** نوع پادگان‌هایی که ممکن است برای ورودی‌ها استفاده شود (مانند پادگان صفر یا پادگان پرشی).
5. **گام‌ها یا گام‌بندی :(Strides)** تعداد پیمایش‌ها و جابجایی‌های انجام شده توسط فیلترها در ورودی.

۲ .قابلیت موازی‌سازی:

- **CNNs:**
  - قابلیت موازی‌سازی برای CNNs معمولاً بالاتر است. از آنجایی که فیلترهای کانولوشن همزمان بر روی ناحیه‌های مختلف تصویر عمل می‌کنند، بخش‌های مختلفی از تصویر می‌توانند به صورت موازی پردازش شوند.
  - این امر از مزیت‌های مهم CNNs است که می‌تواند منجر به سرعت بالاتر در آموزش و پیش‌بینی شود.
- **RNNs:**
  - RNNsدر زمان آموزش و پیش‌بینی قابلیت موازی‌سازی محدودتری دارند. چرا که هر واحد در هر زمان به ورودی وضعیت قبلی خود نیاز دارد، این امر باعث می‌شود که موازی‌سازی بسیار محدودتر و پیچیده‌تر باشد.
  - برای برخی وظایف می‌توان از شبکه‌های بازگشتی موازی بهره برد، اما معمولاً در حالت استفاده معمول، قابلیت موازی‌سازی در RNNs کمتر است و این موضوع می‌تواند سرعت آموزش و پیش‌بینی را کاهش دهد.

سوال پنجم:

۵.الف ابعاد خروجی هر لایه و تعداد پارامترها:

تصویر رنگی ۳ کانال دارد.

$$\text{output size} = \frac{\text{input size} - \text{filter size} + 2 \times \text{padding}}{\text{stride}} + 1$$

فرمول محاسبه تعداد پارامترها:

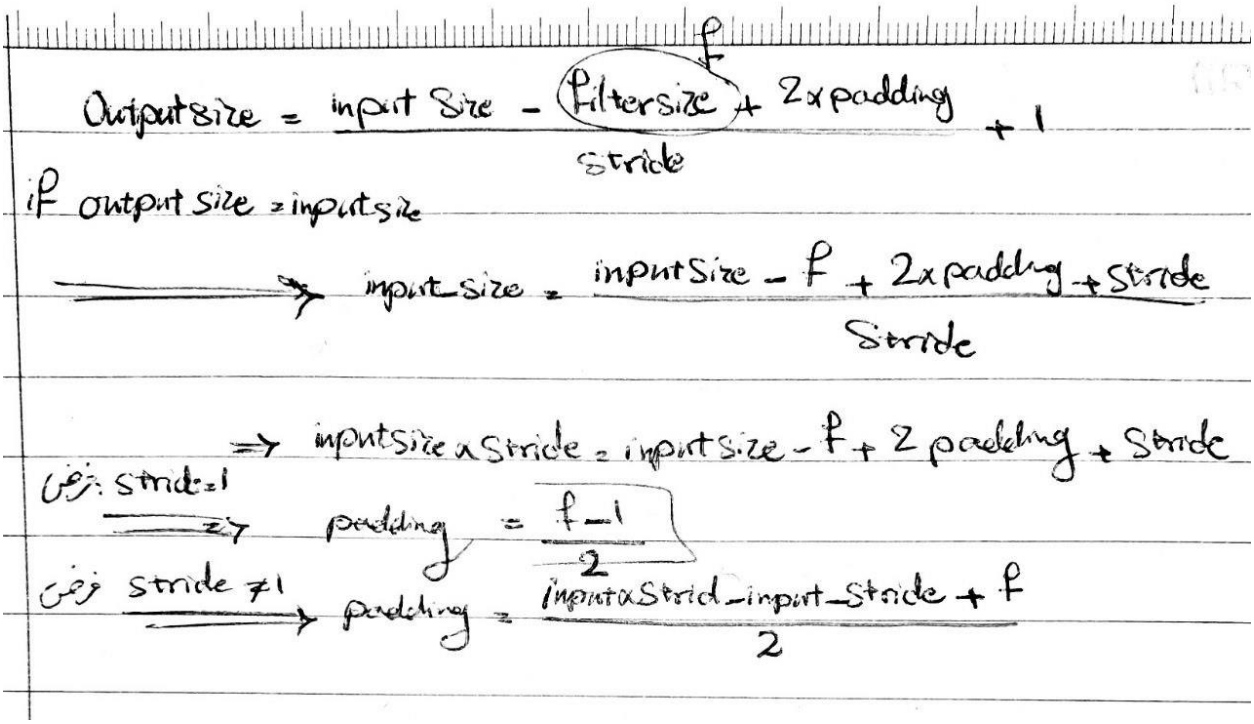
$$\text{تعداد پارامترها} = \text{اندازه فیلتر} \times \text{تعداد کانال‌های ورودی} \times \text{تعداد فیلتر} + \text{تعداد فیلتر}$$

لایه	ابعاد خروجی	توضیحات و محاسبات	تعداد پارامترهای قابل آموزش
Conv(64, (3,3), stride=1, padding='same')	256×256	Parameters = filter_size x input_channels x number_of_filters + number_of_filters Parameters = 3 x 3 x 3 x 64 + 64 = 1,728 + 64 = 1,792 parameters با توجه به پدینگ و استراید1 ابعاد خروجی مثل ورودی میشود.	1,792
Dilated-Conv(32, (5,5), stride=2, dilation rate=2, padding='valid')	126×126	ابعاد خروجی : $\frac{(256-5)+2\times0}{2}+1=126$ Parameters = 5 x 5 x 64 x 32 + 32 = 51,232 + 32 = 51,264 parameters	51,264
Max-pool (size=(2,2), stride=2)	63×63	ابعاد خروجی : $\frac{126-2}{2}+1=63$ تعداد پارامترها: صفر، زیرا Max-pooling بدون پارامتر است.	0
Conv(128, (3,3), stride=1, padding='same')	63×63	Parameters = 3 x 3 x 32 x 128 + 128 = 36,864 + 128 = 36,992 parameters با توجه به پدینگ و استراید1 ابعاد خروجی مثل ورودی میشود.	36,992
Dilated-Conv(64, (5,5), stride=2, dilation rate=4, padding='valid')	30×30	ابعاد خروجی : $\frac{(63-5)+2\times0}{2}+1=30$ Parameters = 5 x 5 x 128 x 64 + 64 = 204,800 + 64 = 204,864	204,864
Max-pool (size=(2,2), stride=2)	15×15	ابعاد خروجی : $\frac{30-2}{2}+1=15$ تعداد پارامترها: صفر، زیرا Max-pooling بدون پارامتر است.	0
Conv(128, (3,3), stride=1, padding='same')	15×15	Parameters = 3 x 3 x 64 x 256 + 256 = 147,456 + 256 = 147,712 parameters با توجه به پدینگ و استراید1 ابعاد خروجی مثل ورودی میشود.	147,712
Dilated-Conv(64, (5,5), stride=2, dilation rate=4, padding='valid')	6×6	ابعاد خروجی : $\frac{(15-5)+2\times0}{2}+1=6$ Parameters = 5 x 5 x 256 x 128 + 128 = 819,200 + 128 = 819,328	819,328

0	ابعاد خروجی : $\frac{6-2}{2}+1=3$ تعداد پارامترها: صفر، زیرا Max-pooling بدون پارامتر است.	3×3	Max-pool (size=(2,2), stride=2)

کل پارامترها: 1,261,952

۵.ب



سوال ششم:

الف)

1. **نرمال سازی دسته‌ای تنها پردازش یک دسته را سریع‌تر می‌کند و زمان آموزش را کاهش می‌دهد و درعین حال تعداد به‌روزرسانی‌ها را ثابت نگه‌می‌دارد. این به شبکه اجازه می‌دهد تا زمان مشابهی را صرف انجام به‌روزرسانی‌های بیشتر کند تا به حداقل برسد.**

درست نیست. نرمال‌سازی دسته‌ای نه تنها به پردازش یک دسته محدود نمی‌شود، بلکه برای هر دسته داده‌ای که وارد شبکه می‌شود، محاسبات نرمال‌سازی انجام می‌شود. همچنین، تعداد به‌روزرسانی‌ها ممکن است متغیر باشد و معمولاً در طول فرآیند آموزش، این تعداد متغیر است. نرمال سازی کمک می کند بتوانیم نرخ آموزش بزرگتری انتخاب کنیم و همگرایی سریعتری اتفاق بیفتد یعنی تعداد به روز رسانی کمتری برای پیدا کردن وزن های صحیح شبکه نیاز باشد. از طرفی تعداد پارامتر های قابل آموزش را افزایش میدهد پس زمان آموزش به طور کلی زیاد میشود.

2. **نرمال سازی دسته‌ای توزیع خروجی را نرمال می‌کند تا در ابعاد یکنواخت‌تر باشد.**

درست است Batch Normalization کمک می‌کند تا توزیع مقادیر خروجی لایه‌های مختلف در هر دسته داده متناسب‌تر و یکنواخت‌تر باشد، که ممکن است منجر به آموزش سریع‌تر و موثرتر شبکه باشد. اگر منظور از خروجی در این جمله خروجی نهایی مدل است اشتباه است و باید قبل از اعمال تابع فعال سازی روی آن نرمالسازی انجام شود.

3. **به شبکه اجازه می‌دهد تا وزن‌های ما را به مقادیر کوچک‌تر نزدیک به صفر مقداردهی کند.**

درست نیست Batch Normalization به شبکه عصبی این امکان را می‌دهد که به واریانس را نزدیک به یک مقداردهی کند، که این موضوع می‌تواند باعث جلوگیری از مشکل موسوم به "gradient vanishing" گردد و برای آموزش سریع‌تر شبکه کمک کند اما اصولا ربطی به نزدیک به صفر کردن داده ها ندارد. بستگی به میانگین مورد نظر برای داده هم دارد و داده ها را حول و حوش میانگین با واریانس کم مقدار دهی میکند. حال اگر میانگین مناسب صفر باشد وزن ها و داده ها مقدار نزدیک به صفر پیدا میکنند وگرنه خیر.

ب)

کد در فایل forward_batchnorm.py تکمیل شده است.

ج)

این ابرپارامتر اضافه شده تا در حالتی که یک فیچر در تمام داده ها مقدار یکسانی دارد و واریانس صفر میشود به واریانس مقداری بزرگتر از صفر بدهد تا وقتی در نرمالسازی در مخرج قرار میگیرد مقدار تعریف نشده برای دیتا به ارمغان نیاورد.

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{ij} - \mu_j)^2 + \epsilon$$

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}$$

$$y_{ij} = \gamma_j \hat{x}_{ij} + \beta_j$$

د)

در این حالت واریانس و میانگین تقریبا بی معناست.

1. **عدم پایداری آموزش** :وقتی از اندازه دسته‌ی یک برای نرمال‌سازی دسته‌ای استفاده می‌شود، این می‌تواند به عدم پایداری در آموزش شبکه‌های عمیق منجر شود. تنها استفاده از یک نمونه به‌عنوان دسته برای محاسبه میانگین و انحراف معیار می‌تواند منجر به اختلالات جدی در آموزش و ناپایداری در همگرایی شبکه شود.
2. **مشکلات نقیض بردار وزن‌ها** :وقتی که اندازه دسته برابر یک است، محاسبه میانگین و انحراف معیار برای نرمال‌سازی دسته‌ای فقط با یک نمونه انجام می‌شود. این ممکن است باعث نقیض بردار وزن‌ها شود که به عدم استفاده موثر از این نرمال‌سازی منجر می‌شود.

3. **عدم تعمیم‌پذیری به داده‌های آزمون** :استفاده از اندازه دسته یک می‌تواند باعث شود که مدل بیش از حد به داده‌های آموزشی خود تکیه کند و اطلاعات کلی مفیدی از داده‌های جدید آموزش نبیند؛ این می‌تواند باعث عدم تعمیم‌پذیری به داده‌های آزمون شود.
4. **مصرف حافظه و زمان محاسباتی بالا** :استفاده از اندازه دسته یک می‌تواند به افزایش مصرف حافظه و زمان محاسباتی نیاز داشته باشد، زیرا هر بار فقط یک نمونه برای نرمال‌سازی در نظر گرفته می‌شود که ممکن است برای مدل‌های بزرگ و داده‌های حجیم مشکل‌ساز باشد.

به طور کلی، استفاده از اندازه دسته یک برای نرمال‌سازی دسته‌ای ممکن است به مشکلاتی مانند عدم پایداری آموزش، نقیض بردار وزن‌ها، عدم تعمیم‌پذیری به داده‌های جدید و افزایش مصرف حافظه و زمان محاسباتی منجر شود. بنابراین، در عمل بهتر است اندازه دسته‌ای متوسط را برای نرمال‌سازی دسته‌ای در نظر گرفته و از این تکنیک با دسته‌های بزرگ‌تر استفاده کرد تا اثربخشی و پایداری بهتری را در آموزش شبکه‌های عمیق حاصل کرد.

ه)

200=20x10 تعداد وزنها

20 تعداد بایاس

2 تعدا پارامترهای بچ نرمالیزیشن

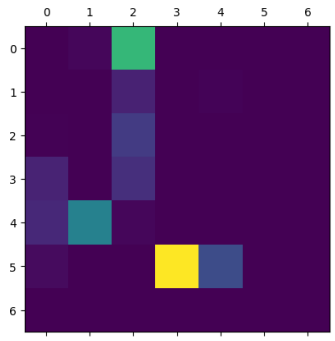
222 تعداد کل پارامترها

سوال هفتم:

با کمک این روش میتوان نقشه حرارتی برای خروجی لایه رسم کرد و با آن فهمیدکدام بخش از تصویر باعث شده است تا شبکه کلاس مربوط به آن تشخیص داده شود و مکان عدد موجود در تصویر را میتوان تخمین زد.

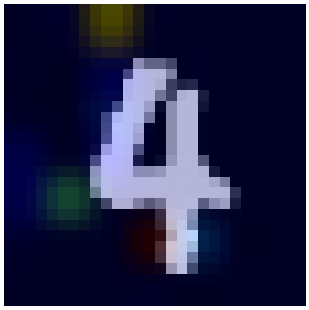
در خروجی آخرین سلول کد در فایل Q7.ipynb میتوانیم ببینیم که برای هر عدد کدام بخش ها فعال شده اند.

مثلا برای تصویری از عدد 4 نقشه حرارتی به صورت زیر در آمده است.



و مدل با توجه به پیکسل های مشخص شده در تصویر توانسته عدد را تشخیص دهد.

پاسخ تمرین چهارم **مهدیه نادری: ۹۸۵۲۲۰۷۶**



مدل به درستی مکان وجود اعداد را پیدا کرده است همچنین برای پیدا کردن هر عدد الگوی مناسبی را در تصویر جست و جو میکند و اینکار را با جست و جوی فعال بودن پیکسل ها در تصویر انجام میدهد و دقت بالایی هم دارد.