سوال اول

الف)

۱. KerasTuner یک کتابخانه متن‌باز برای تنظیم خودکار هایپرپارامترهای شبکه‌های عصبی در کتابخانه Keras می‌باشد. این ابزار به شما کمک می‌کند تا به طور خودکار و بهینه هایپرپارامترهای مربوط به معماری شبکه‌های عصبی خود را تنظیم کنید.

KerasTunerمبتنی بر جستجوی فضای هایپرپارامتر است و می‌تواند انواع هایپرپارامترهای مختلف مانند تعداد لایه‌ها، تعداد واحدها، نرخ یادگیری و توابع فعال‌سازی را به صورت خودکار جستجو کند. با استفاده از KerasTuner، شما می‌توانید یک فضای هایپرپارامترهای تعریف کنید و سپس از الگوریتم‌های جستجوی مانند Random Searchیا Hyperband برای جستجوی بهینه‌ترین مقادیر استفاده کنید.با استفاده از KerasTuner، می‌توانید یک تابع هدف را تعریف کنید که عملکرد شبکه‌ی عصبی را ارزیابی کند. سپس KerasTunerبا اجرای جستجوی هایپرپارامترها و آزمایش شبکه‌های مختلف، بهترین مقادیر هایپرپارامترها را پیدا می‌کند.

۲. برای استفاده از KerasTuner برای مسئله دسته‌بندی، مراحل زیر را می‌توانید دنبال کنید:

۱. تعریف معماری شبکه عصبی: ابتدا باید معماری شبکه عصبی خود را تعریف کنید، از جمله تعداد لایه‌ها، تعداد واحدها، توابع فعال‌سازی و سایر پارامترهای مربوطه.

۲. تعریف فضای هایپرپارامترها: برای هر هایپرپارامتر که قصد تنظیم خودکار آن را دارید، یک فضای جستجو مشخص می‌کنید. مثلاً برای تعداد لایه‌ها می‌توانید یک فضای عدد صحیح مشخص کنید، و برای نرخ یادگیری می‌توانید یک فضای اعداد حقیقی مشخص کنید.

۳. تعریف تابع هدف: شما باید یک تابع هدف تعریف کنید که عملکرد شبکه را ارزیابی کند. برای مسئله دسته‌بندی، می‌توانید از معیارهای ارزیابی معمول مثل دقت(accuracy) ، (confusion matrix)یا معیارهای دیگری که برای مسئله خاص شما مناسب است، استفاده کنید.

۴. تعریف جستجوی هایپرپارامترها: با استفاده ازKerasTuner ، می‌توانید الگوریتم‌های جستجوی مختلف را برای جستجو در فضای هایپرپارامترها استفاده کنید، مانند Random Search یا Hyperband این الگوریتم‌ها به طور خودکار ارزش‌های مختلف هایپرپارامترها را تست می‌کنند و بهترین مقادیر را پیدا می‌کنند.

۵. اجرای جستجوی هایپرپارامترها: با استفاده از تابع مربوطه درKerasTuner ، جستجوی هایپرپارامترها را اجرا کنید. این فرایند شامل آزمایش شبکه‌های عصبی با مقادیر مختلف هایپرپارامترها و ارزیابی عملکرد آن‌ها است.

۶. انتخاب بهترین مدل: پس از اتمام جستجوی هایپرپارامترها، می‌توانید بهترین مدل را بر اساس عملکرد تابع هدف انتخاب کنید. سپس می‌توانید این مدل را بر روی داده‌های آزمون یا داده‌های جدید اعمال کنید و عملکرد آن را ارزیابی کنید.

۳.

درKerasTuner ، چندین تنظیم‌کننده (tuner) برای جستجوی هایپرپارامترها وجود دارد. نام تنظیم‌کننده‌های معروف در KerasTuner عبارتند ازRandomSearch ، Hyperband، BayesianOptimizationوSklearn

۱. RandomSearch : این تنظیم‌کننده از طریق تعیین تعداد تکرارها یک جستجوی تصادفی در فضای هایپرپارامترها انجام می‌دهد. این تنظیم‌کننده به طور تصادفی مقادیر هایپرپارامترها را امتحان می‌کند و عملکرد مدل را ارزیابی می‌کند. این روش سریع است و می‌تواند در صورتی که فضای هایپرپارامترها بزرگ و پیچیده نباشد، نتایج خوبی را به دست آورد.

۲. Hyperbandا : این تنظیم‌کننده یک الگوریتم براساس توقف آسان (early stopping) است که به طور تصادفی تنظیم‌کننده‌های جستجوی هایپرپارامترها را می‌سازد و آنها را به مدل اعمال می‌کند. سپس مدل‌هایی که عملکرد ضعیف‌تری دارند را حذف می‌کند و مدل‌هایی که عملکرد بهتری دارند را نگه می‌دارد. این فرایند را به طور تکراری تکرار می‌کند تا به بهترین مدل برسد. این روش مناسب برای فضاهای هایپرپارامترهای بزرگ است.

۳. BayesianOptimization : این تنظیم‌کننده از الگوریتم بهینه‌سازی بیزی استفاده می‌کند تا به طور هوشمندانه در فضای هایپرپارامترها جستجو کند. این الگوریتم با استفاده از تابع هدف و نتایج قبلی جستجوی هایپرپارامترها را هدایت می‌کند و بهترین مقادیر را پیدا می‌کند. این تنظیم‌کننده برای فضاهای هایپرپارامترهای پیچیده و پیوسته مناسب است.

۴. Sklearn :این تنظیم‌کننده به شما امکان می‌دهد از الگوریتم‌های جستجوی هایپرپارامترهای موجود در کتابخانه Scikit-learn استفاده کنید. این الگوریتم‌ها شامل GridSearch و RandomizedSearch می‌شوند.

من از hyperband استفاده میکنم به دلیل اینکه اگوریتم سریعی هست.

Hyperbandبه دلیل عملکرد سریع و کارآمد خود در جستجوی هایپرپارامترها، به خصوص در مسائل classification، پیشنهاد می‌شود. با استفاده از Hyperband می‌توانید به طور هوشمندانه بهترین مقادیر هایپرپارامترها را پیدا کنید و عملکرد مدل خود را بهبود دهید.

ب)

۱) MNIST یکی از دیتاست‌های پرطرفدار در زمینه بینایی ماشین است. این دیتاست شامل ۶۰,۰۰۰ تصویر سیاه و سفید با ابعاد ۲۸ در ۲۸پیکسل برای آموزش و ۱۰,۰۰۰ تصویر برای آزمون است. هر تصویر در دیتاست MNISTنمایانگر یک عدد از ۰ تا ۹ است و هدف اصلی در این مسئله، تشخیص و دسته‌بندی عدد نمایانگر تصویر است.

۲)

برای استفاده از شبکه CNN بهینه شده با KerasTuner برای دسته‌بندی مسئلهMNIST ، باید مراحل زیر را انجام دهید:

۱. تعریف معماری شبکه :CNN ابتدا باید معماری شبکه CNN را تعریف کنید. این معماری ممکن است شامل لایه‌های کانولوشنال، لایه‌های تراکمی و لایه‌های کاملاً متصل باشد. معماری شبکه را می‌توان به طور دلخواه تعریف کرد با توجه به نیازهای وظیفه دسته‌بندی.

۲. تنظیم‌کننده هایپرپارامتر: برای استفاده ازKerasTuner ، باید یک تنظیم‌کننده هایپرپارامتر را مشخص کنید. شما می‌توانید از تنظیم‌کننده‌های مختلفی مانندHyperband ، BayesianOptimizationو RandomSearchاستفاده کنید. تنظیم‌کننده هایپرپارامتر با استفاده از الگوریتم‌های بهینه‌سازی، بهترین تنظیمات هایپرپارامتر را برای شبکه CNN پیدا می‌کند.

۳. آموزش مدل: با استفاده از تنظیم‌کننده هایپرپارامتر و داده‌های آموزشMNIST ، مدل شبکه CNN را آموزش دهید. برای آموزش می‌توانید از توابع هزینه مناسبی مانند categorical cross-entropy و الگوریتم بهینه‌سازی مانند Adam استفاده کنید. هنگامی که آموزش به پایان رسید، مدل آموزش دیده شبکه CNN برای دسته‌بندی تصاویر MNIST آماده است.

۴. ارزیابی مدل: پس از آموزش مدل، می‌توانید از مدل برای ارزیابی عملکرد در داده‌های تست استفاده کنید. برای این منظور، می‌توانید معیارهای ارزیابی مانند دقتشخیص داده‌های تصویری (MNIST) و شبکه

عصبی کانولوشنال (CNN) را مختلف تنظیم کنید. برای این کار، ابتدا معماری CNN را تعریف کرده و سپس از KerasTuner برای جستجوی بهترین تنظیمات هایپرپارامتر استفاده می‌کنیم.

۳)

۱. Dropout

لایه Dropout به منظور مقابله با مشکل بیش‌برازش (overfitting) در شبکه‌های عصبی استفاده می‌شود. در لایهDropout ، تصادفاً برخی از واحدهای نورونی (نورون‌ها) در هر مرحله آموزش غیرفعال می‌شوند. به عبارت دیگر، خروجی برخی از نورون‌ها در هر مرحله با احتمالی مشخص به صفر تنظیم می‌شود. این عمل باعث می‌شود که شبکه به یادگیری وابستگی‌های غیرضروری و ناهمگون شود و یک نوع میانبر در فرآیند یادگیری ایجاد شود.

استفاده از Dropout مزایا و تأثیرات زیر را به همراه دارد:

- کاهش بیش‌برازش: Dropout مانع از یادگیری وابستگی‌های غیرضروری می‌شود و باعث کاهش بیش‌برازش مدل می‌شود.
- افزایش عمومیت: با کاهش اتکا به جزئیات دقیق داده‌های آموزش، مدل توانایی تعمیم‌پذیری بهتری را در برابر داده‌های تست نشان می‌دهد.

۲. Pooling

لایه‌های Pooling برای کاهش ابعاد فضایی و تعداد ویژگی‌ها در شبکه‌های عصبی استفاده می‌شوند. این لایه‌ها اطلاعات مهم را از تصاویر استخراج کرده و از بین می‌برند، در حالی که ویژگی‌های اساسی را حفظ می‌کنند. این باعث کاهش پیچیدگی محاسباتی و تعداد پارامترها می‌شود، که مزایا استفاده از لایه‌های Poolingعبارتند از:

- کاهش ابعاد فضایی: لایه‌های Pooling با اعمال عملیات مانند Max Pooling یاAverage Pooling ، ابعاد فضایی داده‌های ورودی را کاهش می‌دهند. این کاهش ابعاد باعث کاهش تعداد پارامترها و محاسبات مورد نیاز در شبکه می‌شود که در نتیجه مدل را سریعتر و کارآمدتر می‌کند.
- استخراج ویژگی مهم: عملیات Pooling با استفاده از تابعی مانند Max یاAverage ، ویژگی‌های مهم و قابل تمیز را استخراج می‌کند و ویژگی‌های غیرضروری را کاهش می‌دهد. این کاهش درجه آزادی در داده‌ها باعث می‌شود که مدل بتواند الگوها و ویژگی‌های مهم را با دقت بیشتری تشخیص دهد.

به طور کلی، استفاده از لایه‌های Dropout و Pooling در شبکه‌های عصبی می‌تواند بهبود عملکرد مدل را در مسائل تشخیص الگو، دسته‌بندی تصاویر و پردازش تصویر ارتقا دهد. با اعمالDropout ، مدل مقاومت بیشتری در برابر بیش‌برازش خواهد داشت و با استفاده از لایه‌هایPooling ، میزان پیچیدگی محاسباتی کاهش می‌یابد و ویژگی‌های مهم در داده‌ها استخراج می‌شود.

ج)

در ابتدا درباره کدی که زدم توضیح می دهم یک تابع build_model تعریف کردم و استراکچر مدل و محدوده سرچ هایپرپارامترها را طبق جدول تعیین کردم و بعد preprocess ها را برر وی دیتاست اعمال کردم و در نهایت tuner خود را hyperband انتخاب کردم طبق زیر:

```
tuner = Hyperband(
    build_model,
    objective="val_accuracy",
    max_epochs=5,
    executions_per_trial=2,
    overwrite=True,
    directory='tuner_dir',
    project_name='mnist_model'
)
```

و در ادامه سرچ هایپر پارامترها را شروع کردم و بهترین نتیجه دقت ۹۹ درصد برروی داده های validation بودو

```
Trial 9 Complete [00h 02m 48s]
val_accuracy: 0.5418500006198883

Best val_accuracy So Far: 0.9918000102043152
Total elapsed time: 00h 11m 30s
```

که مدل ان به صورت زیر است:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2  (None, 13, 13, 32)        0
 D)

 dropout (Dropout)           (None, 13, 13, 32)        0

 conv2d_1 (Conv2D)           (None, 11, 11, 96)        27744

 dropout_1 (Dropout)         (None, 11, 11, 96)        0

 conv2d_2 (Conv2D)           (None, 9, 9, 96)          83040

 max_pooling2d_1 (MaxPoolin  (None, 4, 4, 96)          0
 g2D)

 dropout_2 (Dropout)         (None, 4, 4, 96)          0

 flatten (Flatten)           (None, 1536)              0

 dense (Dense)               (None, 32)                49184

 dense_1 (Dense)             (None, 128)               4224

 dense_2 (Dense)             (None, 64)                8256

 dense_3 (Dense)             (None, 10)                650

=================================================================
Total params: 173418 (677.41 KB)
Trainable params: 173418 (677.41 KB)
Non-trainable params: 0 (0.00 Byte)
```

و هم چنین مقدار هایپرپارامترها نیز به صورت زیر است:

و هم چنین مقدار هایپرپارامترها برای بهترین دقت بر روی داده های validation به ص.رت زیر است:

```
Trial 0005 summary
Hyperparameters:
num_conv_layers: 3
conv_0_filters: 32
conv_1_filters: 96
conv_2_filters: 96
num_dense_layer: 3
dense_0_units: 32
dense_1_units: 128
dense_2_units: 64
learning_rate: 0.001
conv_3_filters: 128
conv_4_filters: 96
dense_3_units: 32
dense_4_units: 160
tuner/epochs: 5
tuner/initial_epoch: 2
tuner/bracket: 1
tuner/round: 1
tuner/trial_id: 0003
Score: 0.9918000102043152
```

تمام کد در نوتبوک HW4_Q1 موجود است.

الف)

من اندازه فیلترها را ۳ در ۳ در نظر گرفتم به دلایل زیر:

۱. 3x3باعث کاهش تعداد پارامترهای قابل آموزش در شبکه می‌شود نسبت به فیلترهای با ابعاد بزرگتر مانند۵ یا ۷ این مزیت به معنای کاهش پیچیدگی محاسباتی و استفاده بهینه از منابع محاسباتی است.

۲. افزایش قابلیت تفسیرپذیری: فیلتر ۳ در ۳به دلیل ابعاد کوچک، قابلیت تفسیرپذیری بیشتری دارد. با استفاده از فیلترهای کوچکتر، می‌توان ویژگی‌های جزئی‌تر را از تصاویر استخراج کرد که به تفسیر و تحلیل بهتر نتایج کمک می‌کند.

۳. افزایش قابلیت تشخیص الگوها: استفاده از فیلترهای ۳ در ۳در شبکه کانولوشنال، قابلیت تشخیص الگوهای مختلف را افزایش می‌دهد. این فیلترها قادر به تشخیص الگوهای ساده‌تر مانند خطوط و لبه‌ها و همچنین الگوهای پیچیده‌تر مانند گوشه‌ها و ترکیبات مختلف هستند.

ب)

۱. Pooling

Poolingیک عملیات ساده است که به منظور کاهش ابعاد فضایی ویژگی‌ها در لایه‌های کانولوشنال استفاده می‌شود. عملیات اصلی پولینگ معمولاً Max Pooling است که بزرگ‌ترین عناصر در هر ناحیه مشخص را انتخاب می‌کند و آن‌ها را به عنوان ویژگی‌های مهمتر انتقال می‌دهد. این عملیات می‌تواند دو اثر اصلی داشته باشد:

- کاهش ابعاد فضایی: با کاهش اندازه ناحیه‌ها و انتخاب بزرگ‌ترین عناصر، ابعاد فضایی ویژگی‌ها کاهش می‌یابد. این کاهش می‌تواند منجر به کاهش تعداد پارامترها و محاسبات لازم در لایه‌های بعدی شبکه شود.
- ایجاد اهمیت مکانی: با انتخاب بزرگترین عناصر، پولینگ می‌تواند به شبکه کمک کند تا توجه بیشتری به مکان‌های مهم و ویژگی‌هایی که به طور مکانی مهم هستند، داشته باشد.

۲. Dropout

Dropoutیک روش منتشرکننده است که در مرحله آموزش شبکه استفاده می‌شود. در هر مرحله آموزش، برخی از واحدهای نرونی به طور تصادفی غیرفعال می‌شوند و حذف می‌شوند. این کار باعث جلوگیری از برداشتن وابستگی بیش از حد بین واحدهای نرونی می‌شود و به افزایش تنوع و قابلیت انتقال بیشتر در شبکه کمک می‌کند. این عملیات می‌تواند دو تأثیر اصلی داشته باشد:

- جلوگیری از overfitting :با غیرفعال کردن برخی از واحدهای نرونی در هر مرحله آموزش، Dropoutمی‌تواند از برداشتن وابستگی بیش از حد و حفظ عمومیت شبکه جلوگیری کند. این کار می‌تواند به کاهش پدیده overfitting کمک کند.
- افزایش دقت: با اضافه کردن Dropout به شبکه، میزان تنوع ویژگی‌های استخراج شده توسیط به طور کلی افزایش می یابد. این می‌تواند منجر به افزایش دقت و قابلیت انتقال بهبود یافته در شبکه شود.

سوال دوم

تمامی کدها در فایل medical موجود است مدل پیش اموزش دیده VGG عملکرد بهتری نسبت به resnet داشت.

سوال سوم

از رو نمودار مشخص هست که مدل تغییرات یاد گرفته ولی biased شده به مین علت باید بعد از آخرین LSTM یک لایه batch normalization بذاریم.

سوال چهارم

الف) شبکه های هم‌گشتی(CNN)

شبکه‌های هم‌گشتی، نوعی از شبکه‌های عصبی عمیق هستند که برای پردازش داده‌هایی که قالب فضایی دارند، مانند تصاویر و سیگنال‌های صوتی، استفاده می‌شوند. این شبکه‌ها قدرت بالایی در تشخیص الگوها و ویژگی‌های محلی در داده‌های ورودی دارند.

کاربردها:

شبکه‌های هم‌گشتی به خوبی در بینایی ماشین، تشخیص و تصویربرداری الگوها، تشخیص شیء، ترجمه ماشینی، تشخیص چهره، تشخیص اشیاء، تحلیل موسیقی، تشخیص حرکت و تشخیص عملکرد عضوی و عملکرد مغزی استفاده می‌شوند. به عنوان مثال، در تشخیص تصویر، شبکه‌های هم‌گشتی قادر به تشخیص و تفسیر الگوها و ویژگی‌های مختلف تصاویر هستند.

شبکه‌های بازگشتی (RNN)

شبکه‌های بازگشتی، نوع دیگری از شبکه‌های عصبی عمیق هستند که برای مدل‌سازی و تحلیل داده‌هایی که

وابستگی‌های زمانی دارند، استفاده می‌شوند. این شبکه‌ها قابلیت حفظ حافظه و یادگیری وابستگی‌های طولانی مدت را دارند.

کاربردها:
شبکه‌های بازگشتی در پردازش زبان طبیعی، ترجمه ماشینی، تشخیص گفتار، تولید متن، پردازش سیگنال‌های زمانی مثل سری‌های زمانی و زمان‌بندی، تحلیل احساسات متن و مسائلی که وابستگی‌های زمانی و توالی در آن‌ها مهم هستند، عملکرد بسیار بهتری نسبت به شبکه‌های هم‌گشتی دارند.

ب)

شبکه‌های هم‌گشتی (CNN) و شبکه‌های بازگشتی (RNN) از لحاظ تعداد پارامتر و قابلیت موازی‌سازی با یکدیگر متفاوت هستند.

۱. تعداد پارامترها:
شبکه‌های هم‌گشتی :تعداد پارامترهای شبکه‌های هم‌گشتی به طور کلی کمتر است. این به خاطر استفاده از لایه‌های کانولوشنی که فیلترهای آن‌ها به اشتراک گذاشته می‌شوند می‌باشد. این قابلیت باعث کاهش تعداد پارامترها و منابع مورد نیاز برای آموزش مدل می‌شود.

شبکه‌های بازگشتی : تعداد پارامترهای شبکه‌های بازگشتی عموماً بیشتر است. این به خاطر ماهیت توالی‌بندی وابسته به زمان در شبکه‌های بازگشتی است. هر واحد در لایه بازگشتی نیاز به پارامترهای جداگانه دارد، که تعداد آن‌ها با افزایش تعداد واحدها افزایش می‌یابد. بنابراین، تعداد پارامترها در شبکه‌های بازگشتی معمولاً بیشتر از شبکه‌های هم‌گشتی است.

۲. قابلیت موازی‌سازی:
شبکه‌های هم‌گشتی :یکی از مزایای شبکه‌های هم‌گشتی، قابلیت بالای موازی‌سازی است. به دلیل استفاده از عملیات کانولوشن، تحلیل هر بخش از داده می‌تواند به صورت مستقل و موازی انجام شود. این ویژگی باعث افزایش سرعت پردازش و کاهش زمان آموزش مدل می‌شود.

شبکه‌های بازگشتی :شبکه‌های بازگشتی در مقایسه با شبکه‌های هم‌گشتی، قابلیت موازی‌سازی کمتری دارند. زیرا در هر مرحله از زمان، خروجی لایه قبلی به عنوان ورودی لایه جاری استفاده می‌شود و باید منتظر تکمیل محاسبات لایه قبلی باشد. این وابستگی زمانی باعث محدود شدن قابلیت موازی‌سازی در شبکه‌های بازگشتی می‌شود.

به طور کلی، شبکه‌های هم‌گشتی (CNN) برای پردازش داده‌های با ساختار مکانی و محلی مناسب‌تر هستند، در حالی که شبکه‌های بازگشتی (RNN) برایپردازش داده‌های با ساختار زمانی و وابستگی‌های زمانی مناسب‌تر هستند. همچنین، شبکه‌های هم‌گشتی (CNN) با تعداد پارامترهای کمتر و قابلیت موازی‌سازی بالاتر، معمولاً برای مسائل بزرگتر و محاسباتی پیچیده مورد استفاده قرار می‌گیرند، در حالی که شبکه‌های بازگشتی (RNN) برای مسائلی که نیاز به مدلسازی وابستگی‌های زمانی دارند، مثل ترجمه ماشینی و تولید متن مناسب‌تر هستند.

سوال پنجم

پاسخ به صورت زیر است:

سوال ۵ : الف)

Layer 1:

ابعاد خروجی : $(256, 256, 64)$ و تعداد پارامترها :

$$(3 \times 3 \times 3 + 1) \times 64 = 1792$$

Layer 2:

ابعاد خروجی :

از طریق فرمول : $\text{output\_size} = \dfrac{\text{input size} - \text{filter size} + 2 \times \text{padding}}{\text{stride}} + 1$ [پیش فرض]

در نتیجه فیلتر 5×5 با نرخ 2 ، dialate میشود که در این صورت ابعاد آن 9×9 میشود

$$\left\lfloor \frac{256 - 9}{2} \right\rfloor + 1 = 124$$

ابعاد خروجی : $(124, 124, 32)$

تعداد پارامترها : $(5 \times 5 \times 64 + 1) \times 32 = 51232$

Layer 3:

ابعاد خروجی : $(62, 62, 32)$

تعداد پارامترها : 0

Layer 4:

ابعاد خروجی : $\dfrac{62 - 3 + 2}{1} + 1 = 62$

$\Leftarrow (62, 62, 128)$

تعداد پارامترها : $(3 \times 3 \times 32 + 1) \times 128 = 36992$

Layer 5:

در نتیجه فیلتر 5×5 با نرخ 4 ، dialate میشود که ابعادش برابر با 17×17 میشود

$$\left\lfloor \frac{62 - 17}{2} \right\rfloor + 1 = 23$$

ابعاد خروجی : $(23, 23, 64)$

تعداد پارامترها : $(5 \times 5 \times 128 + 1) \times 64 = 204864$

Layer6:

ابعاد خروجی: (11,11,64)

پارامترها: 0

Layer7:

$$\frac{11-3+2}{1}+1=11 \quad , \quad (11,11,256)$$ ابعاد خروجی:

پارامترها: $(3\times3\times64+1)\times256=147712$

Layer 8:

زمانی که 5x5 با rate 2، dilate می‌شود مثل این است ابعاد فیلتر برابر با (26,26) می‌شود

[illegible] می‌شود.

ب)

Dilated(S, (f,f), stride=2, dilation rate=2) (D = dilation rate)

$$outsize=\frac{W-f-(f-1)\times(D-1)+2\times padding}{S}+1$$

اگر dialation, stride برابر با 2 باشد [illegible]

$$outsize=\frac{W-f-(f-1)\times1+2\times padding}{2}+1$$

و W برابر با 256 بود

$$outsize=\frac{256-2f+1+2\times padding}{2}+1=256$$

$$\rightarrow\frac{256-2f+1+2P}{2}=255 \rightarrow 256-2f+1+2P=510 \Rightarrow$$

$$\boxed{P=\frac{253+2f}{2}}$$

سوال ششم

الف)

عبارت اول:نادرست است به دست آوردن میانگین و واریانس و تقسیم کردن به یکسری بچ زمان میبرد و هم چنین و هم چنینی میانگین و واریانس هم نیار به اپدیت دارند.

عبارت دوم:درست است.

عبارت سوم:نادرست است. نرمال‌سازی دسته‌ای از روش‌های استفاده شده در شبکه عصبی برای کاهش مشکل انفجار گرادیان و کنترل گرادیان‌ها در طول آموزش است. با استفاده از نرمال‌سازی دسته‌ای، ورودی‌های هر دسته مورد محاسبه قرار می‌گیرند و پس از محاسبه میانگین و واریانس آن‌ها، ورودی‌ها نرمال می‌شوند. این کار باعث کاهش واریانس گرادیان‌ها می‌شود و پایداری آموزش را افزایش می‌دهد.

بنابراین، هدف اصلی نرمال‌سازی دسته‌ای اصلاح واریانس گرادیان‌ها است و نه کوچک‌سازی وزن‌ها به صفر.

ب)

کد در فایل مورد نظر کامل شده است.

ج)

ممکن است واریانس برابر با ۰ بشود در این صورت با یک اپسیلون جمع می کنیم تا بر ۰ تقسیم نکنیم.

د)

استفاده از نرمال‌سازی دسته‌ای با اندازه یک مشکلات زیر را ممکن است داشته باشد:

۱. بزرگی اثر تاثیر عملکرد: وقتی تعداد نمونه‌ها در هر دسته بسیار کم است (مثلاً برابر یک)، نرمال‌سازی دسته‌ای ممکن است توانایی خود را در کاهش واریانس گرادیان‌ها از دست دهد. در این حالت، تخمین میانگین و واریانس دسته بر اساس تعداد بسیار کمی نمونه‌ها صورت می‌گیرد و ممکن است دقت تخمین بسیار پایین باشد. علاوه بر این، نرمال‌سازی دسته‌ای در این حالت می‌تواند باعث افزایش نویز و عدم استقلال میان گرادیان‌ها شود.

۲. حساسیت به دسته‌بندی نادر: در صورتی که در یک دسته تمام نمونه‌ها یک شکل و یا نمونه‌های نادر وجود داشته باشند، تخمین میانگین و واریانس بر اساس این نمونه‌ها محدود به دسته خواهد بود. این موضوع می‌تواند باعث تحریک زیادی در مقادیر واریانس و نرمال‌سازی دسته‌ای شود که ممکن است باعث اشتباهات در تخمین گرادیان‌ها و کاهش عملکرد شبکه شود.

۳. تأخیر در آموزش: استفاده از اندازه یک برای نرمال‌سازی دسته‌ای ممکن است باعث افزایش تأخیر در آموزش شود. زیرا هر بار فقط یک نمونه برای محاسبه میانگین و واریانس استفاده می‌شود، که ممکن است زمان بیشتری برای آموزش نیاز داشته باشد.

در کل، استفاده از اندازه بسیار کوچکی برای نرمال‌سازی دسته‌ای ممکن است تاثیر معکوسی بر عملکرد و کارایی نرمال‌سازی دسته‌ای داشته باشد. برای دستیابی به نتایج بهتر، معمولاً توصیه می‌شود تعداد نمونه‌ها در هر دسته را به حدی انتخاب کنید که تخمین مقادیر آماری به خوبی انجام شود و در عین حال تأثیرات نامطلوب مذکور را تا حد امکان کاهش دهید.

ه)

لایه خروجی ۲۰ است و ما دوپارامتر میانگین و واریانس را داریم به همین علت ۲۰*۲ برابر با ۴۰ پارامتر داریم.

سوال هفتم

معماری گفته شده در سوال پیاده سازی شد و بعد در ادامه از tf-keras-vis استفاده کردم یک تابع generate_gradcam تعریف کردم که

```
def generate_gradcam(model, images, labels, target_layer):
    for image, label in zip(images, labels):
        print(label)
        gradcam = Gradcam(model,model_modifier=ReplaceToLinear(), clone=False)
        score = CategoricalScore(np.argmax(label))
        cam = gradcam(score, image, penultimate_layer=target_layer)
        # Min-max normalization
        cam_min = np.min(cam)
        cam_max = np.max(cam)
        cam = (cam - cam_min) / (cam_max - cam_min)
        yield cam
```

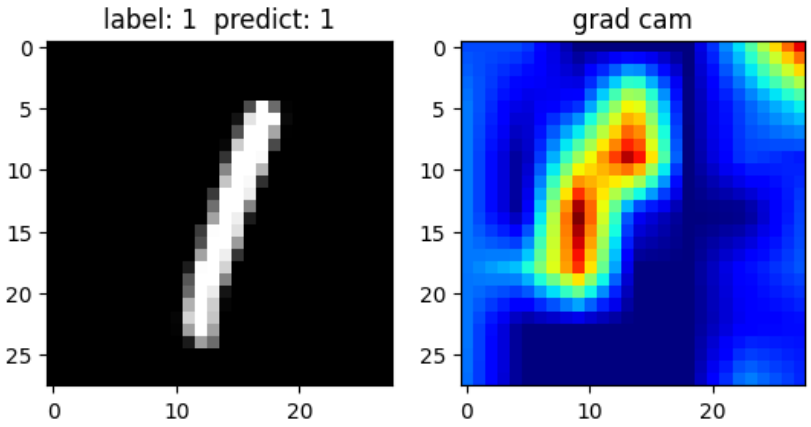
- این تابع شامل ورودی‌های"model" ، "images"، "labels"و "target_layer" است.
- با استفاده از حلقه "for" و تابع "zip" بر روی تصاویر و برچسب‌ها حرکت می‌کند.
- در داخل حلقه، یک نمونه از کلاس "Gradcam" ساخته می‌شود و با انتقال آرگومان‌های"model" ، "model_modifier"و "clone" به آن، مدل Grad-CAM ایجاد می‌شود.

- با استفاده از کلاس"CategoricalScore" ، یک شیء "score" ساخته می‌شود که برچسب تصویر فعلی را نشان می‌دهد.
- سپس با استفاده از شیء"gradcam" ، نقشه حرارتی Grad-CAM با انتقال آرگومان‌های"score" ، "image"و "target_layer" تولید می‌شود.
- مقادیر نقشه حرارتی با استفاده از نرمال‌سازی min-max استاندارد می‌شوند.
- در نهایت، نقشه حرارتی نرمال‌شده ("cam") برای هر تصویر تولید شده را خروجی می‌دهد.

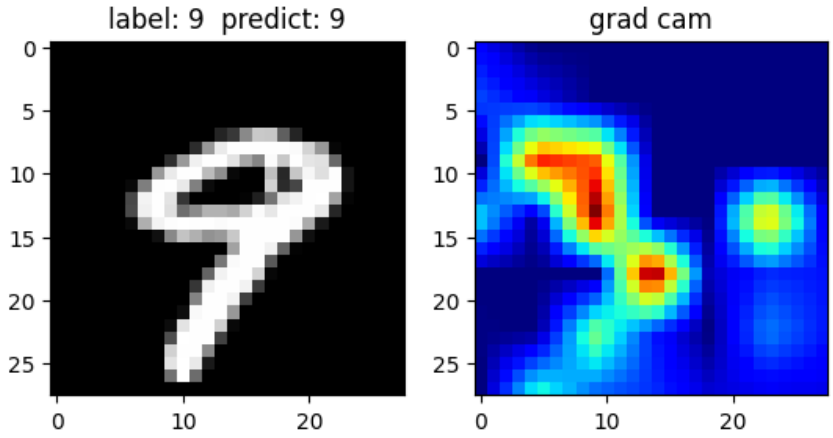
و مطابق شکل زیر heatmap ها را میگیریم:

```
heatmaps = generate_gradcam(model, images, labels, 'conv2d_5')
```

کل کد ها در فایل HW4_Q7 موجود است. نمونه ای از شکل ها خروجی gradcam :



مثلا اینجا به لب عدد ۱ توجه کرده.



یا مثلا برای عدد ۹ به اون انحنا بالا بیشتر توجه کرده است.