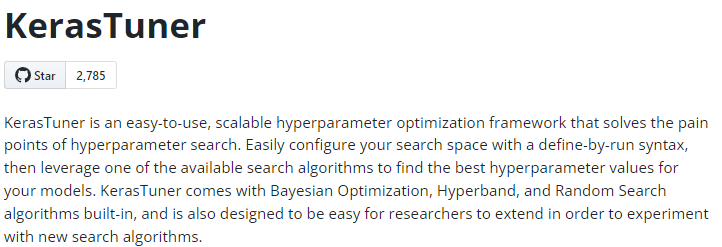
به نام خدا

# تمرین سری چهارم درس مبانی یادگیری عمیق دکتر مرضیه داوود آبادی

# فرزان رحمانی 99521271

## سوال اول

الف)  
**ابزار KerasTuner را معرفي كنيد.**  
KerasTuner یک ماژول تنظیم هایپرپارامتر است که به طور خاص برای مدل های Keras طراحی شده است. این به شما امکان می‌دهد مدل‌های Keras را به راحتی و با کارایی بالا تنظیم کنید، مانند اندازه لایه‌ها، نرخ یادگیری، ترک تحصیل و غیره.  
مزیت اصلی، اتوماسیون تنظیم هایپرپارامتر برای به دست آوردن بهترین مدل با عملکرد است. تنظیم دستی هایپرپارامترها بسیار زمان بر و منابع فشرده(resource intensive) است. KerasTuner به شما اجازه می دهد تا به طور خودکار بسیاری از ترکیبات فراپارامترها را سریعتر آزمایش کنید.  
به بیان دیگر، KerasTuner یک کیت ابزار تنظیم هایپرپارامتر سطح بالا است که بر روی Keras ساخته شده است. این به کاربران اجازه می دهد تا یک فضای جستجو برای فراپارامترهای خود تعریف کنند و به طور خودکار آن را برای یافتن بهترین پیکربندی برای مدل خود کاوش کنند. این می تواند به طور قابل توجهی عملکرد مدل های یادگیری عمیق را با یافتن مقادیر بهینه برای فراپارامترهایی مانند نرخ یادگیری، تعداد لایه ها و توابع فعال سازی بهبود بخشد.  


**توضيح دهيد چگونه مي توان از اين ابزار براي بهينه سازي شبكه هم گشتي جهت دسته بندي داده ها استفاده كرد.**

* ما می توانیم یک معماری شبکه عصبی کانولوشنال را در KerasTuner با استفاده از HyperModel تیونر تعریف کنیم. این ساختار مدل را از تنظیم هایپرپارامتر جدا(abstract) می کند.
* ما فقط hyperparameters را برای تنظیم به عنوان کلاس های HyperParameters مشخص می کنیم. برای CNN ها مواردی مانند num\_filters، kernel\_size، pooling، dropout و غیره است.
* KerasTuner سپس به طور خودکار این هایپرپارامترها را در بسیاری از تکرارها تنظیم می کند تا بهترین ترکیب عملکردی را پیدا کند که ضرر را به حداقل برساند.

همچنین KerasTuner چندین ویژگی را ارائه می دهد که آن را برای بهینه سازی شبکه برای طبقه بندی ارزشمند می کند:

* تنظیم خودکار فراپارامتر(Automatic hyperparameter tuning): نیاز به جستجوی شبکه دستی یا جستجوی تصادفی را که می تواند زمان بر و ناکارآمد باشد را از بین می برد.
* انعطاف پذیری(Flexibility): از توزیع های فراپارامترهای مختلف پشتیبانی می کند و به شما امکان می دهد فضاهای جستجوی مختلف را به طور موثر کاوش کنید.
* ادغام با Keras(Integration with Keras): به طور یکپارچه با Keras API ادغام می شود و استفاده از آن را با مدل ها و گردش کار موجود آسان می کند.
* توقف زودهنگام(Early stopping): می تواند به طور خودکار مرحله های آموزشی را که بعید است عملکرد خوبی داشته باشند، متوقف کند و در زمان و منابع صرفه جویی کند.

**Tuner موجود در *KerasTuner* را معرفي كنيد. شما در پياده سازي خود از كدام يك استفاده مي نماييد؟ چرا؟**

KerasTuner دارای سه تیونر اصلی است: RandomSearch، Hyperband و BayesianOptimization. همچنین تیونر های دیگری مانند Sklearn و GridSearch نیز دارد. همچنین به گونه ای طراحی شده است که توسعه آن برای محققان به منظور آزمایش الگوریتم های جستجوی جدید آسان باشد.

توضیح مختصر در مورد تیونر ها:

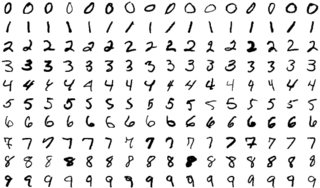
1. RandomSearch: این تیونر به صورت تصادفی از ترکیب هایپرپارامترها برای ارزیابی نمونه برداری می کند که آن را برای کاوش اولیه فضای فراپارامتر کارآمد می کند. زمانی مفید است که فضای جستجو بزرگ است و می‌خواهید بدون جستجوی جامع درک کنید که چه چیزی کار می‌کند.
2. Hyperband: Hyperband یک الگوریتم تطبیقی است که از مکانیزم براکت برای آموزش تعداد زیادی مدل به صورت موازی استفاده می‌کند، مدل‌های ضعیف را کنار می‌گذارد و منابع بیشتری را به مدل‌های امیدوارکننده‌تر تخصیص می‌دهد. از نظر منابع محاسباتی کارآمد است.
3. BayesianOptimization: این تیونر از بهینه سازی بیزی برای کشف فضای فراپارامتر استفاده می کند. تابع هدف را مدل می کند و مجموعه بعدی فراپارامترها را برای ارزیابی بر اساس نتایج قبلی تعیین می کند. زمانی موثر است که فضای جستجو پیچیده باشد و به راحتی به طور تصادفی کاوش نشود.
4. GridSearch: این تیونر به طور کامل تمام ترکیبات فراپارامترهای ارائه شده در فضای جستجو را جستجو می کند. به دلیل ماهیت جامع آن که می تواند از نظر محاسباتی گران باشد، معمولاً برای فضاهای جستجوی بزرگ توصیه نمی شود.
5. Sklearn: تیونر برای مدل های Scikit-Learn. جستجوی فراپارامترهای معتبر متقابل را برای مدل‌های Scikit-learn انجام می‌دهد.

برای این مسئله، من BayesianOptimization را استفاده می کنم. بسیار کارآمد است، بهره برداری و اکتشاف(exploitation and exploration) را به خوبی متعادل می کند و می تواند مدل های بسیار خوبی را با تکرارهای نسبتاً کم به دست آورد.  
یک مدل احتمالی از تابع هدف (test loss) می سازد و از آن برای انتخاب ابرپارامترهای امیدوارکننده برای امتحان بعدی استفاده می کند.   
ما می خواهیم پیکربندی بهینه را با دقت بالا پیدا کنیم، حتی اگر جستجوی فضای جستجو بیشتر طول بکشد.  
BayesianOptimization می تواند در چنین مواردی کارآمدتر از RandomSearch باشد، به خصوص زمانی که با روابط پیچیده بین فراپارامترها سروکار داریم.

ولی باید در نظر داشته باشیم که انتخاب تیونر برای استفاده به آزمایش و یافتن تعادل مناسب بین هزینه محاسباتی و اثربخشی کاوش در فضای فراپارامتر بستگی دارد. همچنین انتخاب تیونر به عواملی مانند اندازه فضای جستجو، منابع محاسباتی موجود و سطح مطلوب بهینه سازی بستگی دارد.

ب)  
**ديتاست *MNIST*را معرفي كنيد.**

مجموعه داده MNIST مجموعه ای از ارقام دست نویس (0-9) است که به طور گسترده در جامعه یادگیری ماشین برای کارهای طبقه بندی تصویر استفاده می شود. این شامل 60000 تصویر آموزشی و 10000 تصویر آزمایشی است که هر کدام از آنها در مقیاس خاکستری و 28\*28 پیکسل است. مجموعه داده MNIST یک مجموعه داده معیار محبوب برای وظایف طبقه بندی تصویر است و مجموعه داده "hello world" برای شروع طبقه بندی تصاویر رقمی در نظر گرفته می شود. و مجموعه داده ای مناسب برای ارزیابی عملکرد مدل های یادگیری ماشین در وظایف طبقه بندی تصویر است. همچنین در زیر نمونه هایی از تصاویر mnsit را مشاهده می کنید.



**توضيح دهيد چگونه مي توان از شبكه هم گشتي بهينه سازي شده با *KerasTuner*براي دسته بندي تصاوير اين ديتاست استفاده كرد.**

* ما یک HyperModel CNN در KerasTuner با پارامترهای قابل تنظیم مانند تعداد لایه های کانولوشن/ادغام، تعداد فیلتر در هر لایه، اندازه هسته(kernel)، نوع ادغام، مقدار dropout و غیره می سازیم.
* KerasTuner با آموزش این مدل بر روی 60 هزار تصاویر آموزشی MNIST و ارزیابی ضرر در مجموعه اعتبار سنجی(10 هزار تصویر)، بهترین مقادیر را برای این هایپرپارامترها پیدا می کند.
* نتیجه نهایی یک معماری CNN بهینه شده برای طبقه بندی تصاویر مجموعه تست MNIST با دقت بالا است.

جواب این قسمت با جزییات بیشتر به زبان bard:

A convolutional neural network (CNN) is a powerful type of deep learning model specifically designed for image classification. It utilizes convolutional layers to extract features from images, followed by pooling layers to reduce the dimensionality and fully-connected layers to perform the final classification.

Here's how KerasTuner can be used to optimize a CNN for MNIST classification:

1. Define the hyperparameter search space: We need to specify which hyperparameters we want to tune and the range of values they can take. This includes parameters like learning rate, number of convolutional layers, number of filters per layer, kernel size, pooling type, pooling size, dropout rate, and activation function.
2. Build the model with KerasTuner: We build the CNN architecture using Keras, but instead of specifying fixed values for hyperparameters, we use KerasTuner's hyperparameters. This allows KerasTuner to automatically explore different combinations of hyperparameters during the search.
3. Compile the model: We compile the model with an appropriate loss function (e.g., categorical cross-entropy) and optimizer (e.g., Adam).
4. Train the model: We train the model on the MNIST dataset, using KerasTuner's search function to explore the specified hyperparameter space. During training, KerasTuner will evaluate different configurations of the model and select the one with the best performance based on a defined metric (e.g., validation accuracy).
5. Evaluate the best model: Once the search is complete, KerasTuner will return the best model configuration. We can then evaluate this model on the test set to assess its final performance.

**اهميت و تأثير استفاده از *dropout* و *pooling* در بهبود عملكرد شبكه را توضيح دهيد.**

* Dropout به طور تصادفی واحدهای ورودی را در حین آموزش روی 0 تنظیم می کند (نویز ضرب شونده باینری تصادفی) تا از بیش برازش جلوگیری شود. این باعث بهبود عملکرد در داده های دیده نشده می شود. در واقع dropout با ایجاد redundancy و اطمینان از اینکه هیچ نورون منفردی بیش از حد به هیچ نورون خاص دیگری وابسته نمی شود، به جلوگیری از برازش بیش از حد کمک می کند. این شبکه را تشویق می‌کند تا ویژگی‌های قوی‌تری را بیاموزد و شانس به خاطر سپردن نویز در داده‌های آموزشی را کاهش می‌دهد، که در نهایت منجر به تعمیم بهتر داده‌های دیده نشده می‌شود.
* لایه های ادغام به تدریج اندازه فضایی را کاهش می دهند تا بار محاسباتی کمتری داشته باشند، به طور طبیعی ویژگی ها را ترکیب می کنند، و جابجایی ها/اعوجاج های کوچک را مدیریت می کنند. به بیان دیگر، لایه های ادغام ابعاد فضایی ورودی را کاهش می دهند و تعداد پارامترها و محاسبات در شبکه را کاهش می دهند و در عین حال مهم ترین اطلاعات را حفظ می کنند. این کمک می کند تا شبکه نسبت به تغییرات جزئی در ورودی تغییرناپذیرتر و قوی تر شود و منجر به استخراج بهتر ویژگی ها شود.
* افزودن dropout معقول و ادغام گاه به گاه بین لایه های کانولوشن بر اساس بهینه سازی Kerastuner، استحکام مدل و دقت طبقه بندی را بهبود می بخشد.

جواب bard با جزییات بیشتر:

* Dropout: Dropout is a regularization technique that randomly removes neurons from the network during training. This helps to prevent overfitting and improve the generalization of the model. By including dropout as a hyperparameter, KerasTuner can determine the optimal dropout rate for the network.
* Pooling: Pooling layers reduce the dimensionality of the feature maps by summarizing the information in local neighborhoods. This can improve the efficiency of the network and reduce the risk of overfitting. KerasTuner can also be used to tune the type of pooling (e.g., max pooling or average pooling) and the size of the pooling window.

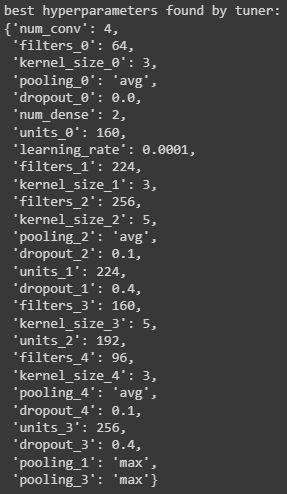
By incorporating dropout and pooling into the network and tuning their parameters with KerasTuner, we can achieve better generalization and improved performance on the MNIST dataset.

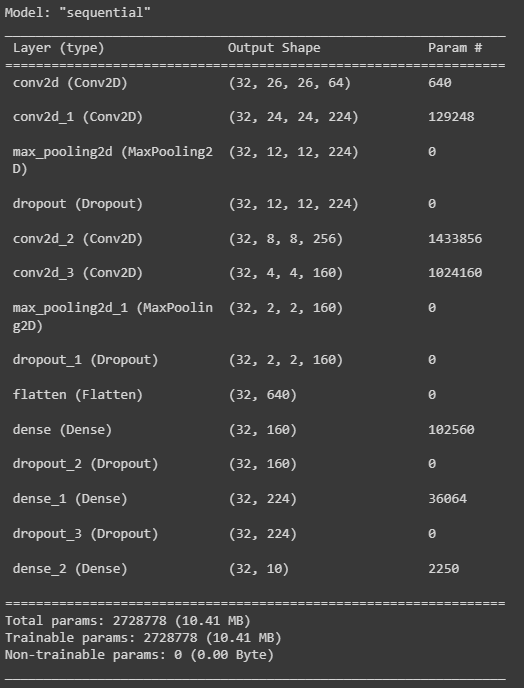
ج)  
**پیاده سازی:**

یک HyperModel CNN را پیاده‌سازی خواهیم کرد که فراپارامترهای قابل تنظیم آن با توجه به صورت سوال به شکل زیر است:

* Number of Conv2D layers: 1 to 5   
  (حداکثر از 5 و حداقل از یک لایه(داشتن حداقل یک لایه برای اینکه شبکه هم گشتی باشد.) استفاده میکنیم)
* Filters per Conv2D layer: 32 to 256  
  (حداکثر از 256 فیلتر و طبق تجربه حداقل از 32 فیلتر با گام 32 برای جستجو استفاده میکنیم.)
* Kernel size: 3x3 or 5x5
* Activation: ReLU (طبق تجربه و مطالب داخل درس تابع فعال ساز مناسبی است)
* Pooling layers: Max or Average
* Dropout rate: 0 to 0.5
* Dense layers: 2 to 5  
  (حداکثر از 5 و حداقل از دو لایه (داشتن حداقل یک لایه پنهان و یک لایه طبقه بندی 10 کلاسه در پرستپترون چند لایه آخر لازم است.) استفاده میکنیم)
* Neurons per layer: 32 to 256  
  (حداکثر از 256 فیلتر و طبق تجربه حداقل از 32 نورون با گام 32 برای جستجو استفاده میکنیم.)
* Optimzer: Adam (با توجه به تجربه)

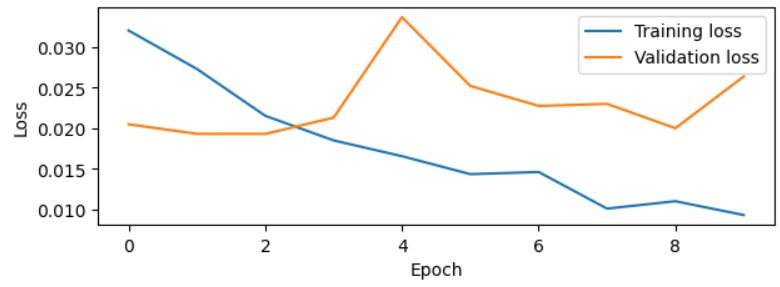
تیونر BayesianOptimization به طور موثر این فضا را برای یافتن بهترین مقادیر جستجو می کند.  
هایپر پارامتر ها و معماری مدلی که BayesianOptimization پیدا کرده است:

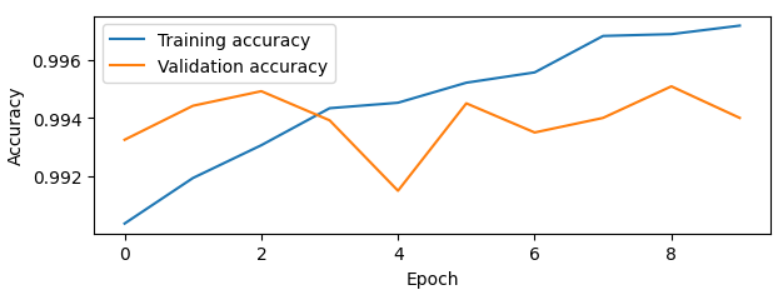




این مدل به دقت 99.36٪ در مجموعه اعتبار سنجی(validation set) دست یافته است.

همچنین در زیر می توانید نمودار های ضرر و دقت را در طور آموزش مشاهده کنید.





توضیح کد به شکل مختصر:  
با مراجعه به فایل DL\_HW4\_Q1.ipynb میتوانید جزییات کامل پیاده سازی را با خواندن کامنت ها مشاهده کنید. در اینجا به طور خلاصه توضیح می دهیم:

* ابتدا keras-tuner را با استفاده از pip نصب میکنیم.
* سپس کتابخانه های لازم را import می کنیم.
* دیتاست mnist را لود و سپس پیش پردازش میکنیم.
* تابع تیونر مدل (CNN HyperModel) را با توجه به خواسته سوال و تجربیات خود کامل کردیم. کامنت ها به طور کامل جزییات پیاده سازی را توضیح داده اند.
* سپس BayesianOptimization را با مدل تیونر تعریف شده می سازیم. از آن برای جستجوی هایپر پارامتر های بهینه استفاده میکنیم. پس از پیدا کردن مدل بهینه هایپر پارامتر های آن را چاپ میکنیم. max\_trials تعداد گام های یافتن مدل های مختلف را نشان می دهد و هر چه بیشتر باشد جستجو بیشتر طول میکشد ولی دقیق تر می شود.
* با بهترین مدلی که یافتیم روی مجموعه داده آموزشی به آموزش آن می پردازیم. ساختار آن را چاپ میکنیم.
* نمودار های دقت و ضرر را در طول آموزش رسم میکنیم.
* در نهایت نیز دقت و ضرر مدل روی داده های تست را میبینیم.

**نحوه انتخاب اندازه فيلترها در لايه هاي *Convolutional*را توضيح دهيد.**

اندازه فیلترها در لایه‌های کانولوشن، اندازه میدان تاثیر(receptive field) را تعیین می‌کند، که ناحیه‌ای از تصویر ورودی است که فیلتر هنگام استخراج ویژگی‌ها در نظر می‌گیرد.

* فیلترهای کوچک: برای ثبت جزئیات کوچک(fine details) در تصویر و مثلا تشخیص لبه مفید هستند و پارامتر و هزینه محاسباتی کمتری دارند.
* فیلترهای بزرگ: برای ثبت الگوها و روابط گسترده‌تر بین پیکسل‌ها مفید است و پارامتر و هزینه محاسباتی بیشتری دارند.

اندازه بهینه فیلتر به مسئله و مجموعه داده خاص بستگی دارد. در عمل، اغلب سودمند است که اندازه های مختلف فیلتر را امتحان کنید و ببینید کدامیک بهترین عملکرد را دارند. می‌توانید از KerasTuner برای جستجوی خودکار اندازه فیلتر بهینه استفاده کنید. معمولا شروع با فیلتر های کوچکتر و در ادامه استفاده از فیلتر های بزرگتر یک الگوی مشترک هست.

جواب سوال به بیان chat gpt:

* The choice of filter sizes in convolutional layers is crucial. Smaller filters (e.g., 3x3) capture finer details and are computationally efficient. Larger filters (e.g., 9x9) capture more global patterns but may increase computational cost.
* Typically, starting with smaller filter sizes and gradually increasing the size while maintaining model performance is a good strategy. For example, a common pattern might be to start with 3x3 filters and then use 5x5 filters and 9x9 in deeper layers.

**چگونه مي توان از *pooling* و *dropout* در شبكه هم گشتي براي جلوگيري از *overfitting* و افزايش دقت استفاده كرد؟**

* Dropout به طور تصادفی واحدهای ورودی را روی 0 قرار می دهد(نویز ضرب شونده باینری)، مدل را مجبور می کند به فعال سازی های خاص تکیه نکند، و تعمیم را بهبود می بخشد. به بیان دیگر، به‌طور تصادفی نورون‌ها را در حین آموزش از بین می‌برد و از تطبیق مشترک آنها جلوگیری می‌کند و اتکای شبکه به ویژگی‌های خاص و نویز های موجود در داده های آموزشی را کاهش می‌دهد.
* لایه های ادغام، نمایش را فشرده تر و نسبت به تغییرات کوچک تغییر نمی دهند. اندازه نقشه‌های ویژگی را کاهش می‌دهد، میدان تاثیر را بیشتر می کنند و شبکه را کمتر مستعد بیش از حد برازش می‌کند. این همچنین به بهبود کارایی محاسباتی با کاهش ابعاد و پارامتر ها کمک می کند.

هم ادغام و هم انصراف می توانند در جلوگیری از برازش بیش از حد و بهبود دقت شبکه مفید باشند. می‌توانید از KerasTuner برای تنظیم پارامترهای این لایه‌ها (به عنوان مثال، نوع ادغام، اندازه پنجره ادغام، نرخ dropout) استفاده کنید تا بهترین پیکربندی را برای مدل خود پیدا کنید.

بیان chat gpt:

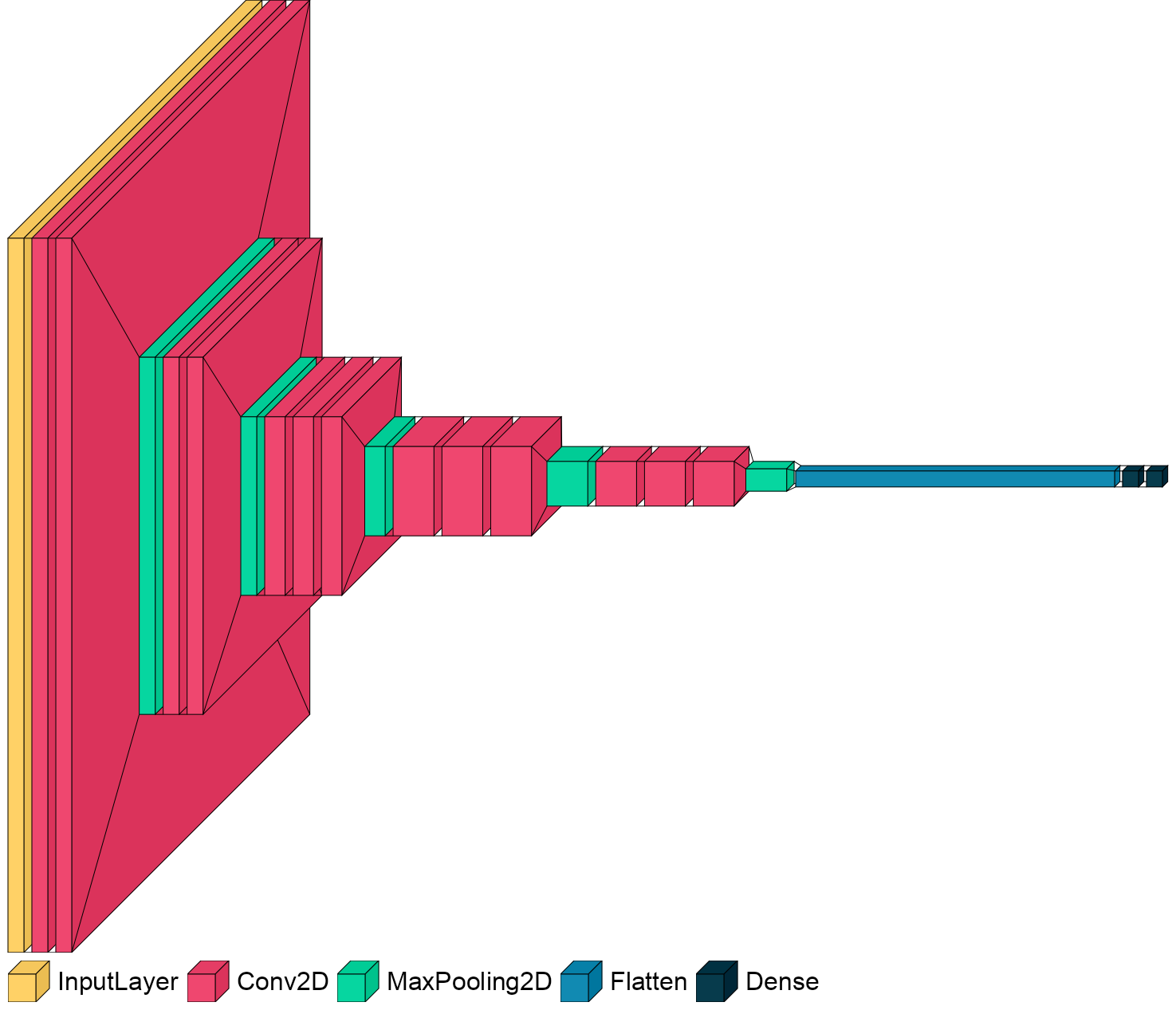
* **Pooling**: MaxPooling2D layers reduce the spatial dimensions of the feature maps, retaining important information while reducing computational complexity. It's used after convolutional layers to downsample the feature maps.
* **Dropout**: Dropout layers help prevent overfitting by randomly dropping a percentage of neurons during training. Placing dropout after convolutional and dense layers helps in regularization and improving the network's generalization ability.

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>   
<https://keras.io/keras_tuner/>   
<https://www.tensorflow.org/tutorials/keras/keras_tuner>

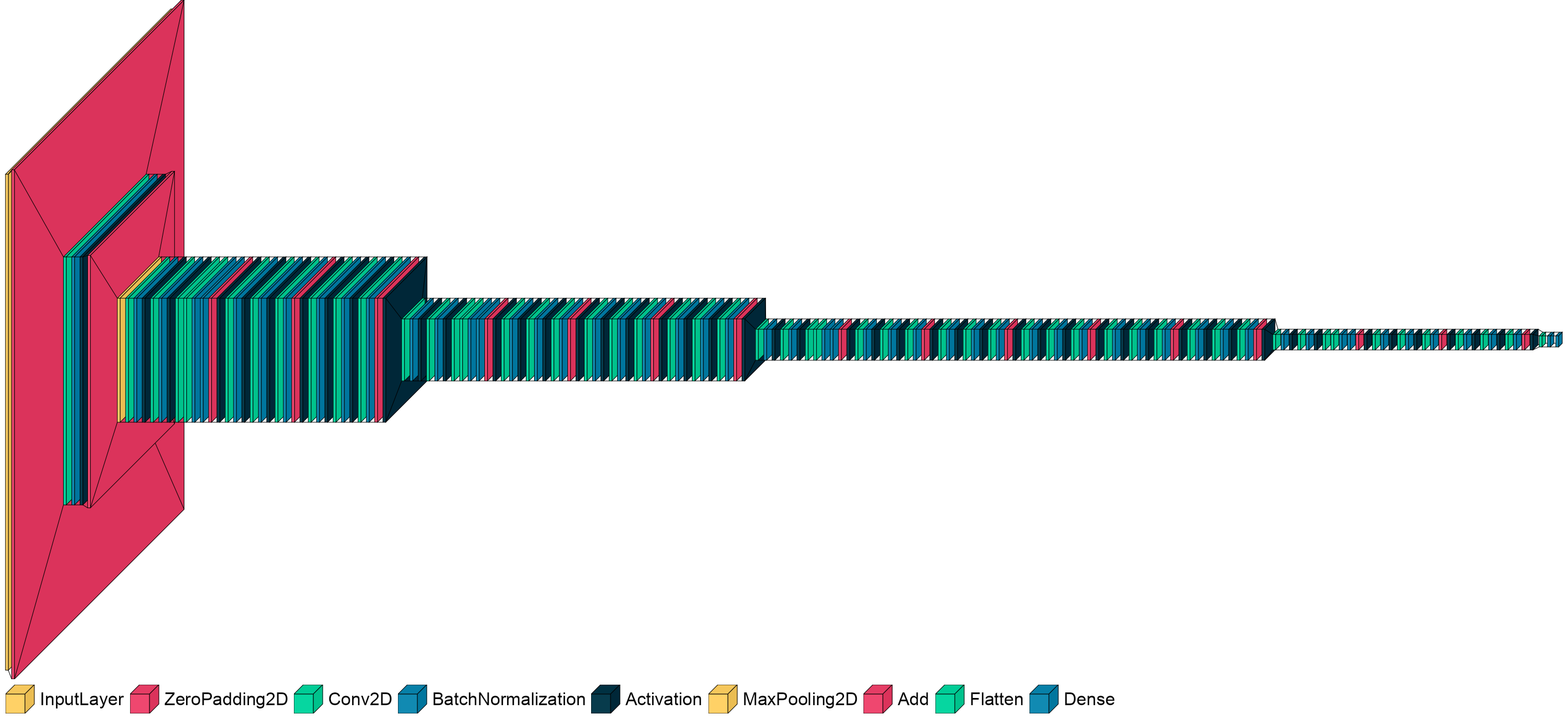
## سوال دوم

با مراجعه به نوتبوک medical.ipynb و کامنت ها می توانید جزییات دقیق تر را مشاهده کنید. در اینجا کد هایی که پیاده سازی کردیم را به طور خلاصه توضیح می دهیم. ابتدا لینک دیتا ست داده شده را باز کردیم و ساختار دیتاست مربوطه را دیدیم. سپس لینک های دانلود را از سایت داده شده برداشتیم و با استفاده از دستور wget آن ها را داخل محیط collab دانلود کردیم. سپس با دستور unzip این شیش فایل را استخراج کردیم. فایل هایی که شامل عکس بودند را در دیرکتوری dataset قرار دادیم. پس از آن با دستور rm فایل های زیپ را حذف کردیم تا حافظه پر نشود.   
با استفاده از کتابخانه panda فایل csv دیتاست را خواندیم. بجای NaN از 0 استفاده کردیم، پسوند .png را به نام عکس ها افزودیم و سپس دیتا فریمی که فقط نام عکس و لیبل(شکستگی یا عدم آن) دارد ساختیم. 10 داده اول دیتا فریم و ابعاد آن را نیز چک کردیم.  
سپس عکس ها را دیرکتوری reshaped\_dataset خواندیم، با استفاده از کتابخانه cv آنها را به ابعاد 3\*224\*224 تغییر دادیم و داخل دیرکتوری reshaped\_dataset ذخیره نمودیم.   
سپس با استفاده از ImageDataGenerator و flow\_from\_dataframe با توجه به خواسته سوال داده های reshape شده را از دیکتوری جدید میخوانیم(در واقع دیتا لودر میسازیم و موقع آموزش و تست خود آن به صورت دسته دسته داده ها را می خواند). دو تا data\_generator یکی برای آموزش و یکی برای تست ساختیم.  
سپس با استفاده از تابع خود سوال چندین نمونه را به تصویر کشیدیم. تابع define\_model را با توجه به خواسته سوال طوری نوشتیم که فقط دو لایه آخر مدل داده شده قابل آموزش باشند و در آخر نیز خودمان یک لایه پنهان و یک لایه با دو نورون برای طبقه بندی قرار دادیم.   
تابع acc\_plot صورت سوال را اصلاح کردیم. استفاده از متغیر model\_history بجای VGG\_history. سپس مدل vgg16 را با وزن های imageNet بارگذاری کردیم ساختار آن را کشیدیم.



سپس مدل را با چک پوینت های ModelCheckpoint که برای ذخیره بهترین حالت مدل است و EarlyStopping که برای توقف آموزش هنگام پیشرفت نکردن است آموزش دادیم. همچنین با کتابخانه time مدت زمان آموزش را پیدا کردیم و تعداد متغیر ها را ذخیره کردیم. نتایج حاصل از مدل را از متغیر VGG\_history پیدا کردیم و طبق خواسته سوال در متغیر های داده شدیم قرار دادیم.   
نمودار های دقت و ضرر را کشیدیم. به علاوه confusion matrix را نیز نمایش دادیم و نتیجه مدل را روی چندین تصویر به نمایش گذاشتیم.

سپس مدل resnet50 را با وزن های imageNet بار گذاری کردیم ساختار آن را کشیدیم.



سپس مدل را با چک پوینت های ModelCheckpoint که برای ذخیره بهترین حالت مدل است و EarlyStopping که برای توقف آموزش هنگام پیشرفت نکردن است آموزش دادیم. همچنین با کتابخانه time مدت زمان آموزش را پیدا کردیم و تعداد متغیر ها را ذخیره کردیم. نتایج حاصل از مدل را از متغیر resnet\_history پیدا کردیم و طبق خواسته سوال در متغیر های داده شدیم قرار دادیم.   
نمودار های دقت و ضرر را کشیدیم. به علاوه confusion matrix را نیز نمایش دادیم و نتیجه مدل را روی چندین تصویر به نمایش گذاشتیم.  
در نهایت، با کشیدن جدولی نتایج هر دو مدل را نمایش دادیم و عملکرد آنها را با هم مقایسه کردیم و دریافتیم در اینجا با تداد پارامتر های کمترvgg16 از resnet50 عملکرد بهتری دارد.

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

## سوال سوم

توضیح در مورد دقت r2:

R2

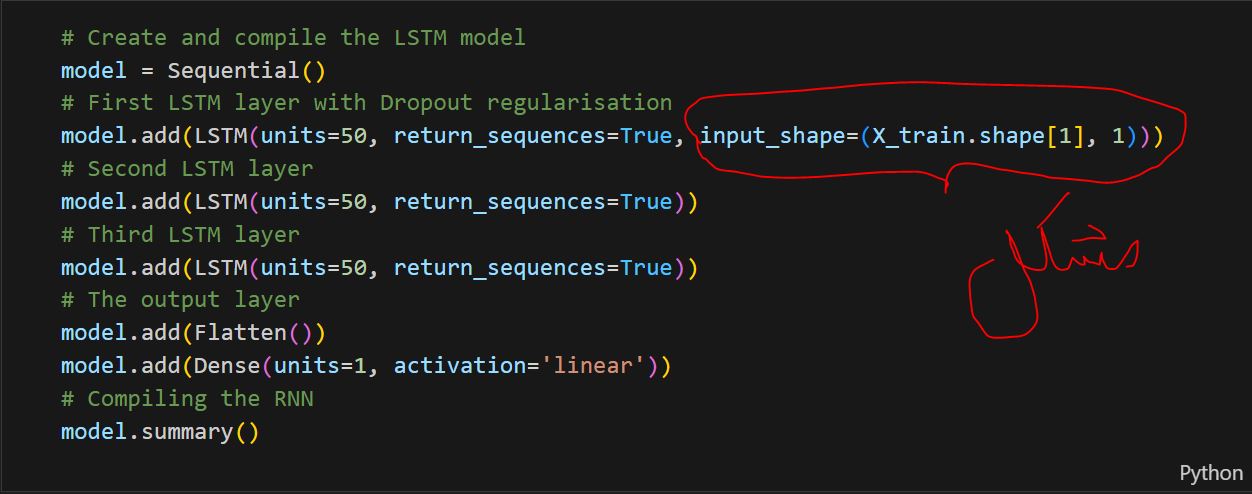

In [statistics](https://en.wikipedia.org/wiki/Statistics), the **coefficient of determination**, denoted *R*2 or *r*2 and pronounced "R squared", is the proportion of the variation in the dependent variable that is predictable from the independent variable(s).

It is a [statistic](https://en.wikipedia.org/wiki/Statistic) used in the context of [statistical models](https://en.wikipedia.org/wiki/Statistical_model) whose main purpose is either the [prediction](https://en.wikipedia.org/wiki/Prediction#Statistics) of future outcomes or the testing of [hypotheses](https://en.wikipedia.org/wiki/Hypotheses), on the basis of other related information. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

The computational definition of *R*2, however, is divorced from both the notation and this common interpretation. Apart from the special case of a linear regression model with an intercept term, *R*2 is not actually equal to the square of any particular quantity. It is calculated by taking the mean of the squared errors, dividing by the variance of the dependent variable, and subtracting this ratio from 1. Since there is no limit to how bad a model’s predictions can be—and thus no limit to how big the errors can get—it’s possible for this ratio to become arbitrarily large, and 1 minus a large value is negative

In practice, *R*2 will be negative whenever your model’s predictions are worse than a constant function that always predicts the mean of the data.

در بالا تعریف دقت r2 و معنی اینکه منفی شده است را مشاده میکنید. حال علت و مشکل کد داده شده را بیان میکنیم و پیشنهادی برای اصلاح آن ارائه میکنیم. به نظر می رسید مشکل داده یک ورودی در هر زمان و استفاده نکردن از خاصیت input sequences در LSTM (sequence modeling capabilities of LSTMs) است. در واقع می توانیم مثلا با داده های 24 ساعت قبل به پیش بینی بپردازیم نه صرفا با داده های همان ساعت.



پنجره‌بندی توالی‌های ورودی(Windowing input sequences): به‌جای ویژگی‌های ورودی مسطح(flattened input features)، قالب‌بندی مجدد داده‌ها به دنباله‌ای از مشاهدات گذشته(sequences of past observations) به عنوان پنجره‌های ورودی می‌تواند قابلیت‌های مدل‌سازی توالی LSTMها(sequence modeling capabilities of LSTMs) را بهتر به کار گیرد. این شامل نگاه کردن به X ساعت/روز گذشته به عنوان یک توالی ورودی برای پیش‌بینی ساعت/روز بعدی است.  
در اینجا نحوه به‌روزرسانی کد برای استفاده از دنباله‌های ورودی پنجره‌دار برای LSTM به‌جای ورودی‌های مسطح آمده است:

# Load data and split into train/test

# ...same as before...

# Parameters

SEQUENCE\_LENGTH = 24

N\_FEATURES = 6

# Reshape into sequences

def to\_sequences(X, y, seq\_len):

  X\_sequences = []

  y\_sequences = []

  for i in range(len(X) - seq\_len):

    X\_seq = X[i:i+seq\_len]

    y\_seq = y[i+seq\_len]

    X\_sequences.append(X\_seq)

    y\_sequences.append(y\_seq)

  return np.array(X\_sequences), np.array(y\_sequences)

X\_train\_seq, y\_train\_seq = to\_sequences(X\_train, y\_train, SEQUENCE\_LENGTH)

X\_test\_seq, y\_test\_seq = to\_sequences(X\_test, y\_test, SEQUENCE\_LENGTH)

# Reshape features

X\_train\_seq = X\_train\_seq.reshape(X\_train\_seq.shape[0], X\_train\_seq.shape[1], N\_FEATURES)

X\_test\_seq = X\_test\_seq.reshape(X\_test\_seq.shape[0], X\_test\_seq.shape[1], N\_FEATURES)

# Create LSTM model

model = Sequential()

model.add(LSTM(units=50, return\_sequences=True, input\_shape=(SEQUENCE\_LENGTH, N\_FEATURES)))

model.add(LSTM(units=50, return\_sequences=True))

model.add(LSTM(units=50, return\_sequences=True))

model.add(Flatten())

model.add(Dense(units=1, activation='linear'))

این کد داده ها را به دنباله های ورودی/خروجی قالب بندی می کند، ورودی ها را به درستی برای لایه LSTM تغییر شکل می دهد و مدل را بر روی توالی ها به جای داده های مسطح آموزش می دهد.

انتخاب طول دنباله(SEQUENCE\_LENGTH) می تواند برای یافتن مقدار بهینه مقداری آزمایش نیاز داشته باشد. در اینجا ما از 24 استفاده کردیم.

چند دستورالعمل در مورد طول دنباله:

* توالی های طولانی تر زمینه تاریخی بیشتری را برای مدل فراهم می کنند اما تعداد نمونه های قابل استفاده را کاهش می دهند. دنباله های کوتاهتر برعکس عمل می کنند.
* با بررسی دوره های فصلی رایج در داده ها شروع می‌کنیم. برای داده‌های ساعتی، فصلی بودن روزانه طول دنباله 24 را پیشنهاد می‌کند. برای داده‌های هفتگی، با طول دنباله یک یا چند هفته شروع می‌کنیم.
* می‌توانید چند مقدار طولانی‌تر و کوتاه‌تر را امتحان کنید، مانند 12، 18، 36 و غیره. موردی که سریع‌تر با کمترین ضرر اعتبار همگرا شود ترجیح داده می‌شود.
* هیچ قانون مشخصی وجود ندارد و طول توالی کاملاً به فراوانی و فصلی بودن داده‌های سری زمانی خاص بستگی(seasonality of the particular time series data) دارد.
* محدوده معمولی بین چند فریم (6-12 برای داده های فرکانس بالا) تا چند فصل(season) (20-60 برای داده های ساعتی/روزانه) است.

بنابراین در این مورد، 24 برای پوشش الگوی کل روز انتخاب شده است. اما می‌توانیم پنجره‌های کوتاه‌تر (۱۲ ساعت) یا کمی طولانی‌تر (۳۶ ساعت) را آزمایش کنیم. ما باید تاثیر روی همگرایی مدل و عملکرد اعتبارسنجی را بررسی کنیم.

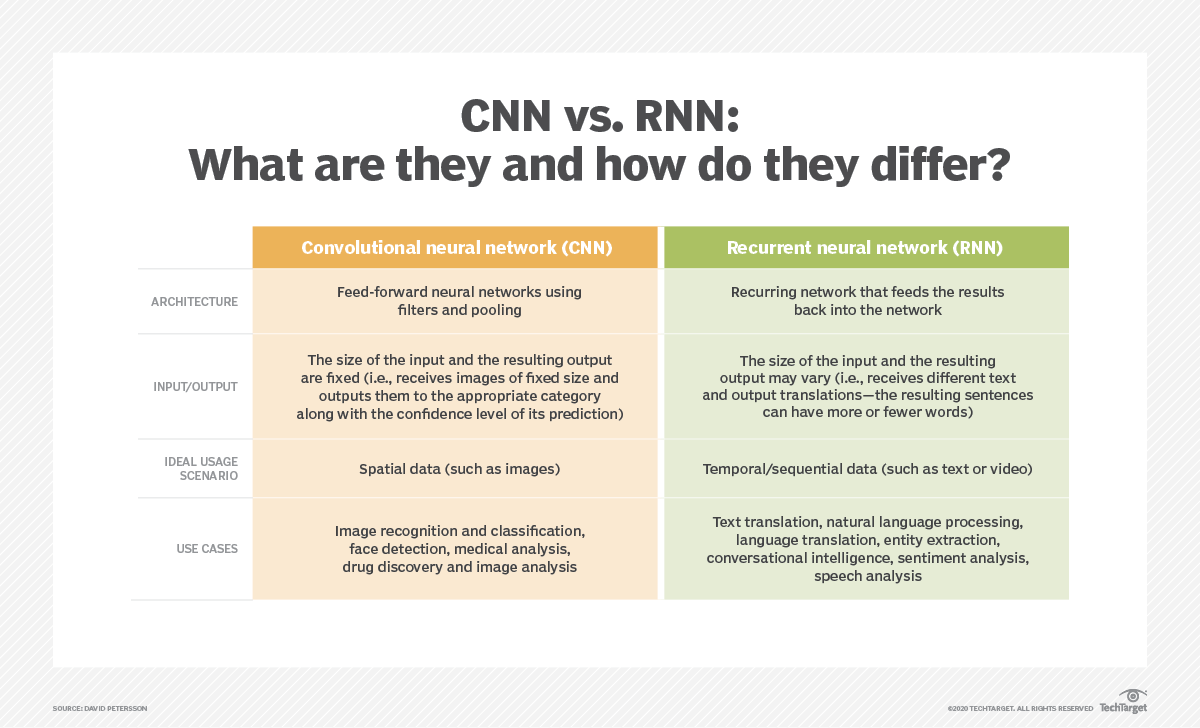
همچنین پیشنهاد های دیگری نیز برای بهبود r2 score به شرح زیر است:

1. پیش پردازش داده ها: اطمینان حاصل کنید که داده های شما به درستی پیش پردازش شده اند. در این مورد، میتوانید از MinMaxScaler استفاده کنید که برای مدل های LSTM مناسب است. با این حال، هر گونه مقادیر دورافتاده(outliers)، مقادیر از دست رفته(missing values) یا تبدیل داده های نادرست که ممکن است بر عملکرد مدل تأثیر بگذارد را بررسی کنید.
2. مهندسی ویژگی(Feature Engineering): ویژگی هایی را که استفاده می کنید مرور کنید. گاهی اوقات، افزودن ویژگی های مرتبط تر یا ایجاد ویژگی های مشتق شده می تواند عملکرد مدل را افزایش دهد. اطمینان حاصل کنید که ویژگی های مورد استفاده مرتبط هستند و تأثیر قابل توجهی بر پیش بینی مصرف گاز دارند.
3. پیچیدگی مدل: معماری مدلی که انتخاب کرده اید ممکن است برای داده هایی که دارید بسیار پیچیده باشد. سعی کنید معماری را ساده کنید یا تعداد لایه های LSTM را کاهش دهید تا از برازش بیش از حد جلوگیری کنید.
4. دوره ها و اندازه دسته ای: در طول آموزش مقادیر مختلف را برای تعداد دوره ها و اندازه دسته آزمایش کنید. این مدل ممکن است به دوره های بیشتری برای همگرایی مناسب نیاز داشته باشد یا می تواند از اندازه های دسته کوچکتر/بزرگتر بهره مند شود.
5. تکنیک‌های منظم‌سازی: تکنیک‌های منظم‌سازی مانند Dropout یا منظم‌سازی L1/L2 را برای جلوگیری از برازش بیش از حد و بهبود تعمیم اجرا کنید.
6. نرخ یادگیری: نرخ یادگیری بهینه ساز را تنظیم کنید. گاهی اوقات، نرخ یادگیری بالا می تواند از همگرایی جلوگیری کند، در حالی که نرخ یادگیری بسیار پایین می تواند منجر به همگرایی کند شود.

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>   
<https://benjaminobi.medium.com/what-really-is-r2-score-in-linear-regression-20cafdf5b87c>   
<https://help.desmos.com/hc/en-us/articles/202529139-Why-am-I-seeing-a-negative-R-2-value->   
<https://en.wikipedia.org/wiki/Coefficient_of_determination>

## سوال چهارم



الف) تعاریف، کاربردها و نقاط قوت:

1. شبکه های عصبی هم گشتی (CNN):

* تعریف: CNNها شبکه‌های عصبی پیش‌خور(feedforward neural networks) هستند که برای پردازش داده‌ها با ساختار شبکه‌ای(grid-like structure) مانند تصاویر یا ویدئوها طراحی شده‌اند. آنها از فیلترها برای استخراج ویژگی ها از داده های ورودی استفاده می کنند، سپس نتایج را برای کاهش ابعاد ادغام می کنند.
* کاربرد ها:  
  تشخیص تصویر(Image recognition): طبقه بندی اشیاء(Classifying objects) در تصاویر (به عنوان مثال، سگ، گربه، اتومبیل)، تشخیص اشیاء(detecting objects) (مانند چهره ها، نشانه ها)، و شرح تصاویر(image captioning).  
  تصویربرداری پزشکی: تشخیص بیماری ها بر اساس اسکن های پزشکی (به عنوان مثال، اشعه ایکس، اسکن MRI).  
  تجزیه و تحلیل ویدئو: تشخیص حرکت(Action recognition)، تشخیص ناهنجاری و خلاصه سازی ویدئو.
* نقاط قوت:  
  استخراج کارآمد ویژگی: CNN ها می توانند به طور خودکار ویژگی های مرتبط را از تصاویر و ویدیوها استخراج کنند و نیاز به مهندسی ویژگی های دستی را از بین ببرند.  
  تغییر ناپذیری فضایی(Spatial invariance): CNN ها نسبت به جابه‌جایی داده های ورودی قوی هستند.  
  موازی سازی: CNN ها را می توان به راحتی روی GPU ها موازی کرد که منجر به آموزش و استنتاج سریعتر می شود.

2. شبکه های عصبی بازگشتی (RNN):

* تعریف: RNN ها )که پیش خور نیستند و به خود باز می‌گردند( برای پردازش داده های متوالی(sequential data) مانند متن، گفتار و داده های سری زمانی طراحی شده اند. آنها حافظه داخلی دارند که به آنها اجازه می دهد ورودی های گذشته را به خاطر بسپارند و از آنها برای پیش بینی آینده استفاده کنند.
* کاربرد ها:  
  پردازش زبان طبیعی (NLP): ترجمه ماشینی، خلاصه سازی متن، تجزیه و تحلیل احساسات و توسعه ربات چت(chatbot).  
  تشخیص گفتار(Speech recognition): تبدیل زبان گفتاری به متن.  
  پیش بینی سری های زمانی: پیش بینی مقادیر آینده یک سری زمانی بر اساس داده های گذشته.
* نقاط قوت:  
  مدل‌سازی داده‌های متوالی: RNN‌ها می‌توانند به طور موثر وابستگی‌های بلندمدت در داده‌های متوالی را ثبت کنند، که برای کارهایی مانند مدل‌سازی زبان و تشخیص گفتار بسیار مهم است.  
  انعطاف پذیری: RNN ها را می توان با تغییر معماری و پارامترهای آموزشی با انواع مختلف داده های متوالی سازگار کرد.

ب) مقایسه

1. تعداد پارامترها:

* CNN ها معمولاً پارامترهای کمتری نسبت به RNN برای پردازش همان مقدار داده دارند. این به این دلیل است که CNN ها از اشتراک گذاری و ادغام وزن استفاده می کنند که باعث کاهش تعداد وزن های منحصر به فرد می شود که باید یاد بگیرند.
* به عنوان مثال، یک CNN که برای تشخیص تصویر استفاده می شود ممکن است میلیون ها پارامتر داشته باشد، در حالی که یک RNN که برای ترجمه ماشینی استفاده می شود ممکن است میلیاردها پارامتر داشته باشد.

2. موازی سازی:

* CNN ها به دلیل معماری فید فوروارد شان بسیار قابل موازی سازی هستند. این بدان معنی است که لایه های مختلف را می توان به طور همزمان پردازش کرد که منجر به آموزش و استنتاج سریعتر می شود.
* RNN ها کمتر قابل موازی سازی هستند زیرا حافظه داخلی آنها وابستگی هایی را بین لایه ها ایجاد می کند. با این حال، تکنیک‌های مختلفی مانند شبکه‌های حافظه کوتاه‌مدت (LSTM) و واحدهای بازگشتی دروازه‌ای (GRU) می‌توانند پتانسیل موازی‌سازی RNN را بهبود بخشند.

به طور خلاصه، CNN ها برای پردازش داده های شبکه مانند(grid-like data) و استخراج ویژگی های فضایی(spatial) مناسب تر هستند، در حالی که RNN ها در مدیریت داده های متوالی(sequential data) و در بر گرفتن وابستگی های طولانی مدت بهتر هستند. انتخاب بین CNN و RNN به مسئله خاص و نوع داده ای که با آن کار می کنید بستگی دارد.

جواب سوال به بیان chat gpt:

الف)

**Convolutional Neural Networks (CNNs):**

* **Definition**: CNNs are a class of deep neural networks primarily designed for processing structured grid-like data. They use convolutional layers to automatically and adaptively learn spatial hierarchies of features from the input data.
* **Applications**: CNNs are best suited for tasks involving image and video recognition, object detection, image segmentation, and other computer vision tasks. They excel in recognizing patterns in spatial data due to their ability to capture spatial relationships.
* **Strengths**: CNNs can efficiently handle high-dimensional input data like images. They utilize parameter sharing and local connectivity, reducing the number of parameters and enabling effective feature extraction from input data.
* **Best for**: Tasks where spatial locality and spatial hierarchies are crucial, such as image classification, object detection, and image segmentation.

**Recurrent Neural Networks (RNNs):**

* **Definition**: RNNs are a type of neural network designed to work with sequential data by maintaining an internal state/memory. They process sequences step-by-step, capturing dependencies and relationships between elements in the sequence.
* **Applications**: RNNs are well-suited for natural language processing (NLP) tasks, time series analysis, speech recognition, and any sequential data processing tasks. They can handle variable-length input/output sequences.
* **Strengths**: RNNs can model sequences of arbitrary lengths, capturing temporal dependencies in data. They're effective in tasks where the order of data matters and previous context influences the current prediction or output.
* **Best for**: Sequential data tasks such as language modeling, machine translation, sentiment analysis, and time series prediction.

ب)

Number of Parameters:

* **CNNs**: Typically have fewer parameters compared to RNNs due to parameter sharing in convolutional layers, making them more computationally efficient for processing high-dimensional data like images.
* **RNNs**: Have more parameters, especially in models like LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit), as they maintain internal states across time steps, making them more complex.

Parallelization:

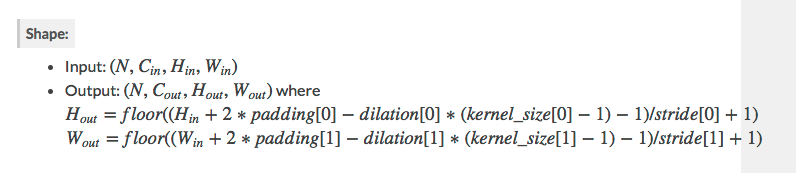
* **CNNs**: Highly parallelizable due to the independence of convolutional filters, allowing for efficient utilization of GPU hardware for simultaneous processing of different parts of an image.
* **RNNs**: Sequential nature limits parallelization. While some parts of RNNs can be parallelized, the dependencies between time steps often restrict full parallel execution.

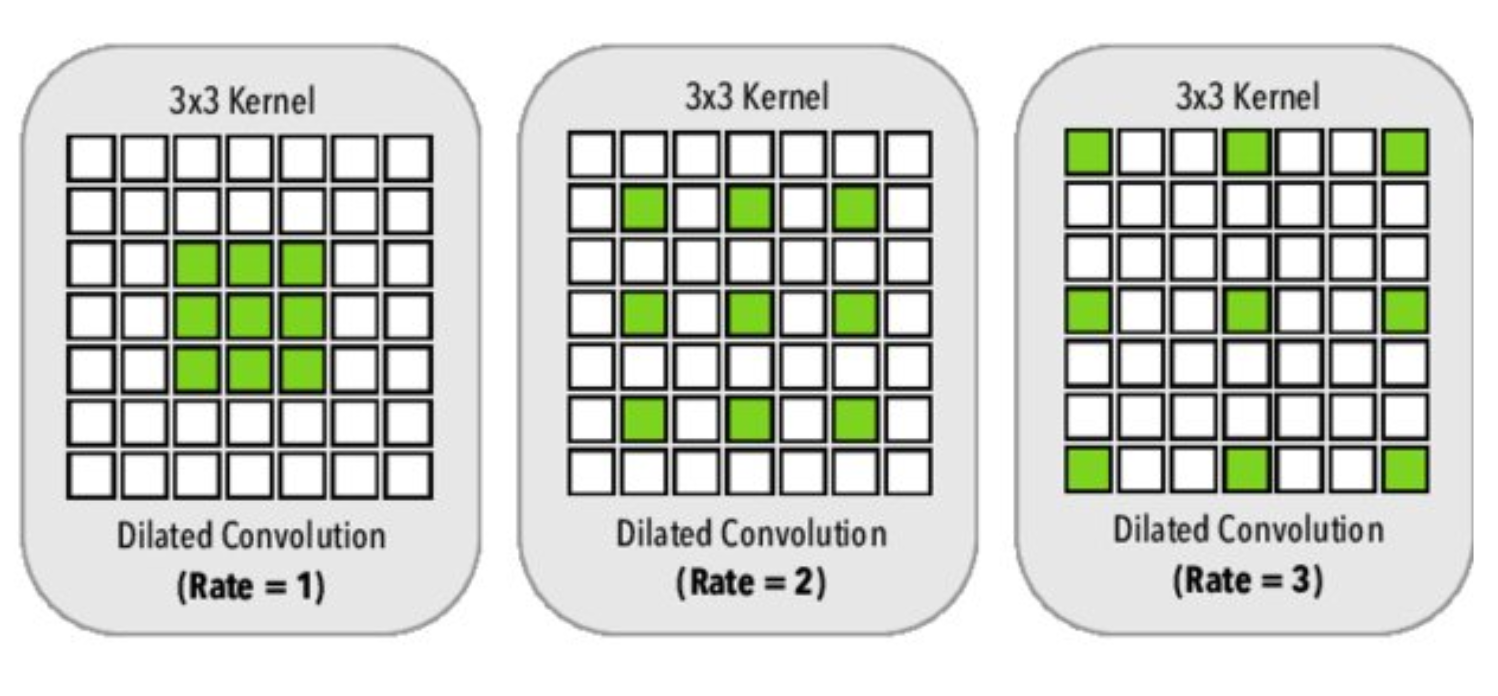
In summary, CNNs excel in tasks involving spatial data, leveraging parameter sharing and local connectivity. On the other hand, RNNs are ideal for sequential data tasks due to their ability to capture temporal dependencies. The choice between CNNs and RNNs depends on the nature of the data and the specific problem being addressed.

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>   
<https://www.techtarget.com/searchenterpriseai/feature/CNN-vs-RNN-How-they-differ-and-where-they-overlap>

## سوال پنجم





الف) بیایید اندازه خروجی و تعداد پارامترها را برای هر لایه در شبکه کانولوشنی داده شده با توجه به فرمول بالا محاسبه کنیم:

**Layer1: Conv(64, (3,3), stride=1, padding='same')**

* Input size: 256x256x3 (RGB image)
* Number of filters: 64
* Filter size: 3x3
* Stride: 1
* Padding: 'same'

Output size:

* With 'same' padding, the output size will be the same as the input size: 256x256x64

Number of parameters:

* Parameters per filter: (3x3x3) + 1 (bias) = 28 parameters
* Total parameters: 64 filters \* 28 parameters = 1,792 parameters

**Layer2: Dilated-Conv(32, (5,5), stride=2, dilation rate=2, padding='valid')**

* Input size: 256x256x64
* Number of filters: 32
* Filter size: 5x5
* Stride: 2
* Dilation rate: 2
* Padding: 'valid'

Output size:

* Output size calculation:  
     
  (Note: Output will be a fractional value, but in practical scenarios, it's often rounded down)
* Rounded output size: 124x124x32

Number of parameters:

* Parameters per filter: (5x5x64) + 1 (bias) = 1,601 parameters
* Total parameters: 32 filters \* 1,601 parameters = 51,232 parameters

**Layer3: Max-pool (size=(2,2), stride=2)**

* Input size: 124x124x32
* Pooling size: 2x2
* Stride: 2

Output size:

* Output size calculation:
* Output size: 62x62x32

No additional parameters in Max-pooling layers. Parameters=0

**Layer4: Conv(128, (3,3), stride=1, padding='same')**

* Input size: 62x62x32
* Number of filters: 128
* Filter size: 3x3
* Stride: 1
* Padding: 'same'

Output size:

* With 'same' padding, the output size will be the same as the input size: 62x62x128

Number of parameters:

* Parameters per filter: (3x3x32) + 1 (bias) = 289 parameters
* Total parameters: 128 filters \* 289 parameters = 36992 parameters

**Layer5: Dilated-Conv(64, (5,5), stride=2, dilation rate=4, padding='valid')**

* Input size: 62x62x128
* Number of filters: 64
* Filter size: 5x5
* Stride: 2
* Dilation rate: 4
* Padding: 'valid'

Output size:

* Output size calculation:
* Output size: 23x23x64

Number of parameters:

* Parameters per filter: (5x5x128) + 1 (bias) = 3,201 parameters
* Total parameters: 64 filters \* 3,201 parameters = 204,864 parameters

**Layer6: Max-pool (size=(2,2), stride=2)**

* Input size: 23x23x64
* Pooling size: 2x2
* Stride: 2

Output size:

* Output size calculation:   
   (Rounded down for practical purposes)
* Output size: 11x11x64

No additional parameters in Max-pooling layers. Parameters=0

**Layer7: Conv(256, (3,3), stride=1, padding='same')**

* Input size: 11x11x64
* Number of filters: 256
* Filter size: 3x3
* Stride: 1
* Padding: 'same'

Output size:

* With 'same' padding, the output size will be the same as the input size: 11x11x256

Number of parameters:

* Parameters per filter: (3x3x64) + 1 (bias) = 577 parameters
* Total parameters: 256 filters \* 577 parameters = 147,712 parameters

**Layer8: Dilated-Conv(128, (5,5), stride=2, dilation rate=8, padding='valid')**

* Input size: 11x11x256
* Number of filters: 128
* Filter size: 5x5
* Stride: 2
* Dilation rate: 8
* Padding: 'valid'

Output size:

* Output size calculation:
* Output size: -10 x -10 x 128

Number of parameters:

* Parameters per filter: (5x5x256) + 1 (bias) = 6,401 parameters
* Total parameters: 128 filters \* 6,401 parameters = 819,328 parameters

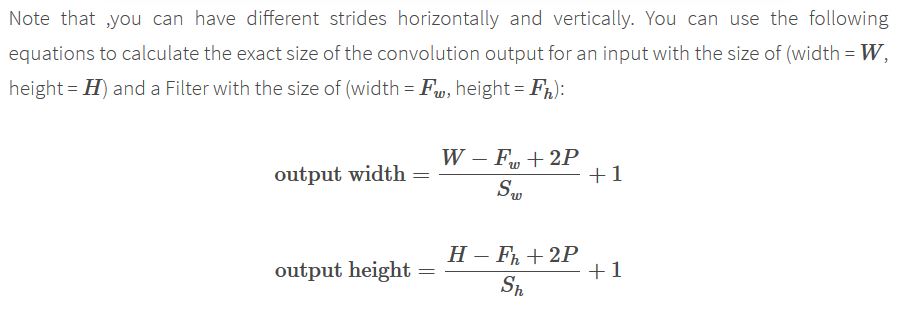
**Layer9: Max-pool (size=(2,2), stride=2)**

* Input size: -10 x -10 x 128
* Pooling size: 2x2
* Stride: 2

Output size:

* Output size calculation:   
   (Rounded down for practical purposes)
* Output size: -5 x -5 x 128

No additional parameters in Max-pooling layers. Parameters=0

ب)  


در لایه‌های کانولوشن دوبعدی، مقدار padding مورد نیاز برای حفظ اندازه ورودی (یعنی حفظ ابعاد خروجی با ابعاد ورودی) را می‌توان بر اساس اندازه فیلتر، ورودی و گام مورد استفاده در عملیات پیچش(convolution) تعیین کرد.

بیایید سناریوی زیر را برای کانولوشن دو بعدی در نظر بگیریم:

* اندازه ورودی: (که در آن N هم ارتفاع و هم عرض ورودی را نشان می دهد)
* اندازه فیلتر: (که در آن F هم ارتفاع و هم عرض فیلتر را نشان می دهد)
* گام(stride): S (فیلتر چقدر به صورت افقی و عمودی حرکت می کند)

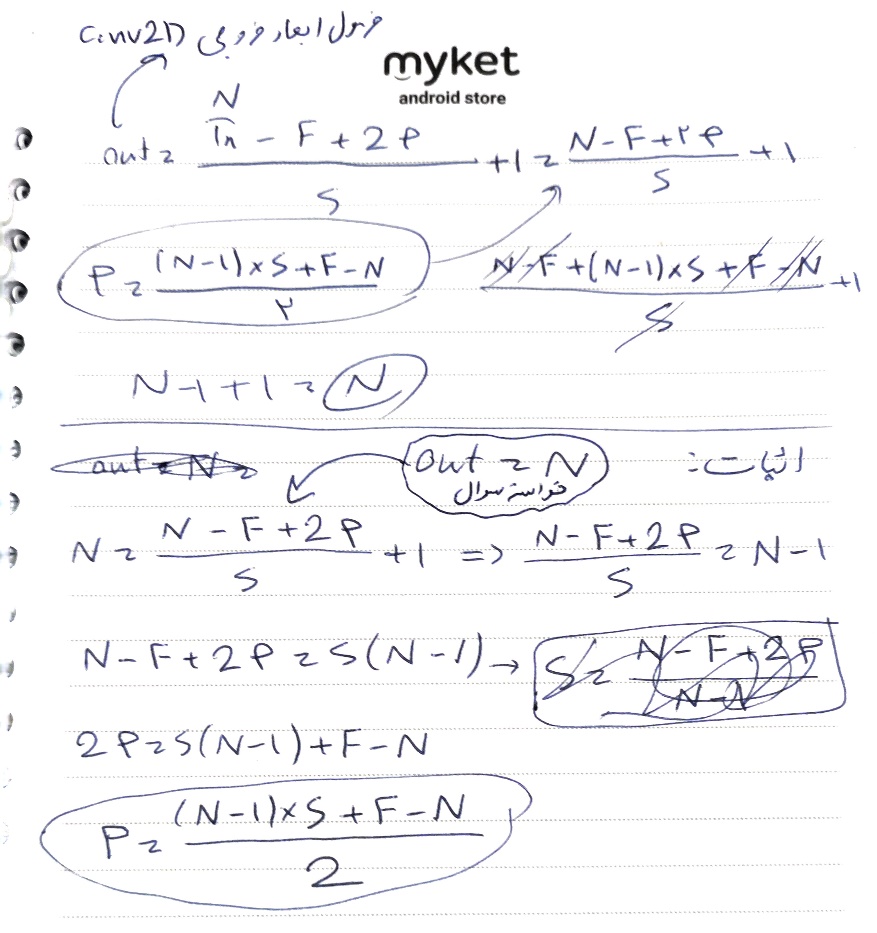
فرمول محاسبه padding مورد نیاز برای حفظ اندازه خروجی با توجه به اندازه ورودی در یک لایه کانولوشن به شرح زیر است:

این فرمول تضمین می کند که اندازه خروجی پس از عملیات کانولوشن با اضافه کردن padding مناسب مانند اندازه ورودی باقی می ماند و ابعاد خروجی تغییر نمی‌کند.

توضیح فرمول:

* تغییر کل فیلتر را در بعد ورودی (به استثنای موقعیت اولیه) محاسبه می کند.
* تغییر اندازه را پس از کانولوشن محاسبه می کند.
* تقسیم بر 2 تضمین می کند که padding مورد نیاز به طور مساوی در اطراف ورودی توزیع شده است.

تنظیم padding روی این مقدار محاسبه شده منجر به یک اندازه خروجی می شود که هنگام استفاده از یک اندازه فیلتر خاص و گام برداشتن در یک لایه کانولوشنال دو بعدی با اندازه ورودی مطابقت دارد.



مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

## سوال ششم

الف) مورد دوم درست است.  
از بین سه عبارت، فقط عبارت "نرمال سازی دسته ای توزیع خروجی را نرمال می کند تا در ابعاد یکنواخت تر باشد." درست است.  
در اینجا دلیل آن را می بینید:

* عبارت نادرست: نرمال سازی دسته ای تنها پردازش یک دسته را سریع تر می کند و زمان آموزش را کاهش می دهد و در عین حال تعداد به روز رسانی ها را ثابت نگه می دارد. این به شبکه اجازه می دهد تا زمان مشابهی را برای انجام به روز رسانی های بیشتر کند تا به حداقل برسد.

دلیلی وجود ندارد که فرض کنیم نرمال سازی دسته ای پردازش یک دسته را سریعتر می کند. برعکس، ما می‌توانیم انتظار داشته باشیم که نرمال‌سازی دسته‌ای هر تکرار تمرین را کندتر کند، زیرا پردازش اضافی مورد نیاز در طول پاس رو به جلو(forward pass) و فراپارامترهای اضافی که نیاز به آموزش در طول پس انتشار (back propagation) دارند. همانطور که گفته شد، ما انتظار داریم که شبکه سریعتر از آن بدون نرمال سازی دسته ای همگرا شود، بنابراین انتظار می رود آموزش شبکه به طور کلی سریعتر تکمیل شود.

* عبارت درست: نرمال سازی دسته ای توزیع خروجی را نرمال می کند تا در ابعاد یکنواخت تر باشد.

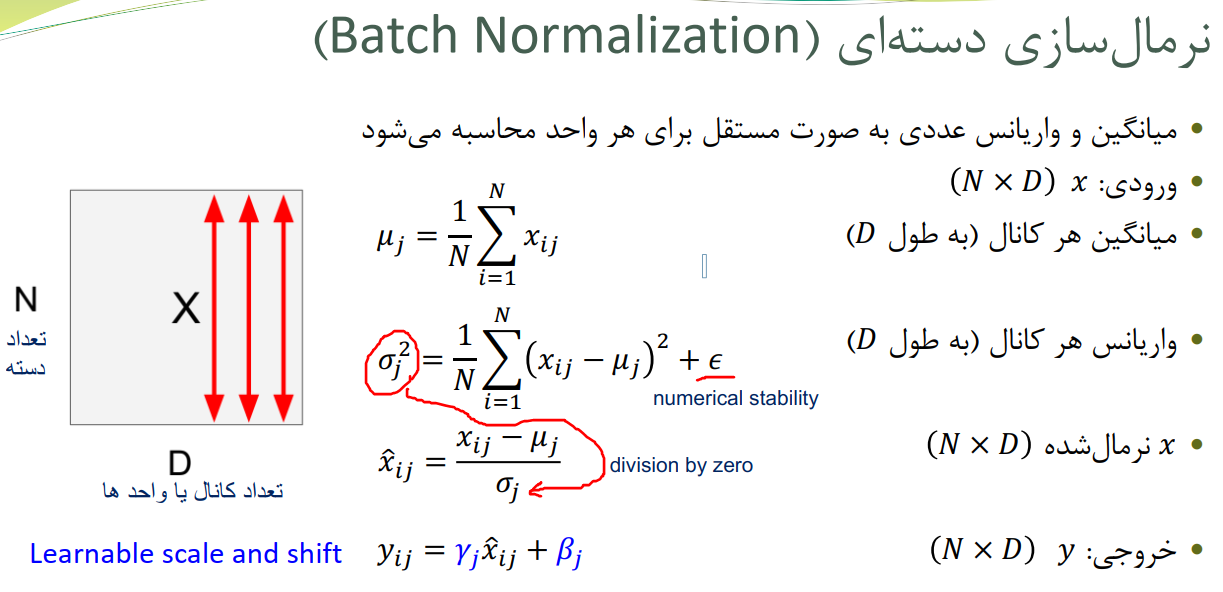
این اساساً هدف نرمال‌سازی دسته‌ای است، یعنی عادی‌سازی ورودی‌های هر لایه (که به نوبه خود خروجی‌های لایه قبل از آن هستند) با توجه به میانگین و واریانس مقادیر در مینی دسته فعلی(current mini-batch). همانطور که در مطالب درسی دیده می شود، ما معمولاً به میانگین صفر و واریانس واحد نرمال می کنیم.

* عبارت نادرست: نرمال سازی دسته ای به شبكه اجازه مي دهد تا وزن های ما را به مقادير كوچك تر نزديك به صفر مقداردهي كند.

این عبارت معتبر نیست، زیرا ما به سادگی اثر مقدار اولیه وزن خود را محدود می کنیم و سپس ورودی را برای هر لایه عادی(نرمال) می کنیم، تاثیر وزن های اولیه تعیین شده را ضعیف می کنیم، اما لزوماً به شبکه اجازه نمی دهیم وزن های ما را به مقادیر کوچکتر نزدیک تر به صفر مقداردهی کند.

ب) با توجه به تعریف های داده شده در ابتدای تابع و کمک گرفتن از نمونه کدی که داخل اسلاید های درس وجود داشت به پیاده سازی این سوال پرداختیم. فقط نیاز به پیاده سازی 6 خط داشتیم که با TODO و کامنت نشان داده شده بودند و بسیار ساده بودند. ابتدا با استفاده از np.mean میانگین را (برای هر کانال) محاسبه کردیم. سپس با np.mean یا np.var میانگین را (برای هر کانال) محاسبه کردیم. در زمان تست آنها را لود کردیم از دیکشنری داده شده. با کم کردن از میانگین و تقسیم بر انحراف معیار به علاوه اپسیلون داده ها را نرمال کردیم. در نهایت ضرب در گاما و به علاوه بتا کردیم. همچنین برای تست کردن آن نیز یک آرایه 5\*3 ساختیم و با صدا زدن تابع دیدیم که تابع درست است.

ج) از تقسیم بر 0 برای ویژگی های با واریانس 0 جلوگیری می کند.  
در ادامه توضیحات بیشتر و کامل تر آمده است.



در نرمال سازی دسته ای، فراپارامتر ε (epsilon) نقش مهمی در تضمین ثبات عددی(numerical stability) و جلوگیری از تقسیم بر صفر ایفا می کند. این دو هدف اصلی را دنبال می کند:

1. اجتناب از تقسیم بر صفر: هنگام محاسبه فعال سازی نرمال شده (Z\_norm) در نرمال سازی دسته ای، شامل تقسیم بر جذر واریانس (var) ورودی است. در مواردی که واریانس بسیار کوچک یا نزدیک به صفر باشد، این امر می‌تواند منجر به خطای تقسیم بر صفر شود و باعث بی‌ثباتی عددی و مانع از فرآیند آموزش شود. افزودن یک مقدار اپسیلون(ε) کوچک به واریانس با اطمینان از مخرج غیر صفر، حتی برای واریانس های کوچک، از این امر جلوگیری می کند.

2. نرم کردن گرادیان(Smoothing the gradient): وجود اپسیلون همچنین به نرم تر شدن گرادیان در حین انتشار پسین کمک می کند. بدون اپسیلون، وقتی واریانس نزدیک به صفر است، گرادیان می‌تواند بسیار تیز و سیخ‌دار(sharp and spiky) شود. این می تواند منجر به آموزش ناپایدار و ناکارآمد شود. افزودن یک مقدار اپسیلون کوچک، گرادیان را صاف و نرم تر می‌کند و بهینه‌ساز را برای حرکت در فضای پارامتر و یافتن راه‌حل‌های بهینه آسان‌تر می‌کند.

به طور خلاصه، اپسیلون نقش مهمی در نرمال سازی دسته ای ایفا می کند:

* جلوگیری از تقسیم بر صفر خطا.
* نرم کردن شیب(Smoothing the gradient) برای آموزش کارآمد.

مقدار معمولی برای اپسیلون 1e-5 است، اما بسته به مجموعه داده‌های خاص و معماری شبکه می‌توان آن را تنظیم کرد.

جواب سوال به بیان chat gpt:

The hyperparameter ϵ (epsilon) in batch normalization plays a crucial role in stabilizing the division operation within the normalization process, specifically in the denominator.

In batch normalization, the standard normalization step involves subtracting the mean and then dividing by the standard deviation:

Z\_norm = (Z – mu) / sqrt{sigma^2 + epsilon}

Where:

* *Z* is the input to the batch normalization layer.
* *μ* is the mean of the input batch.
* sigma^2 is the variance of the input batch.
* *ϵ* (epsilon) is a small constant added to the variance to avoid division by zero.

The primary role of *ϵ* is to prevent division by zero and handle situations where the calculated variance is extremely small or zero. This addition of *ϵ* in the denominator ensures numerical stability during training by preventing potential numerical instabilities that might arise when the variance is close to zero.

If the variance of a particular feature is very small or zero, without *ϵ*, the division by nearly zero could lead to numerical issues such as exploding gradients or overly amplified effects of small variations in the input. Adding *ϵ* prevents these issues and ensures a smoother and more stable training process by regularizing the normalization process, especially in cases where the variance might be small.

Typical values for *ϵ* are small positive numbers, often in the range of 1e-5 to 1e-8. The specific choice of *ϵ* can vary based on the dataset and the characteristics of the neural network being trained.

د)  
به طور خاص برای این سوال، اگر ما فقط یک اندازه کوچک 1 داشته باشیم، خروجی از لایه BN همیشه 0 خواهد بود (x−µ برابر 0 است، آن را در γ ضرب می کنیم که همچنان 0 خواهد بود، و β را اضافه می کنیم. یک پارامتر "بایاس" است که به 0 مقداردهی اولیه می شود)، و بنابراین ما نمی توانیم ویژگی های معنی دار را یاد بگیریم. به طور کلی، زمانی که شما فقط تعداد کمی نمونه در هر دسته دارید، نرمال سازی یک دسته منطقی نیست. نرمال سازی دسته در زمان آموزش بهم می ریزد زیرا تخمین های میانگین / واریانس شما فوق العاده noisy خواهد بود (فقط با استفاده از چند نمونه سعی می کنیم جمعیت / توزیع واقعی را تخمین بزنیم). همین امر در زمان آزمون نیز صدق می‌کند - ما معمولاً از میانگین متحرک میانگین/واریانس، جمع‌آوری‌شده در حین آموزش، در زمان آزمون استفاده می‌کنیم، اما این میانگین متحرک نیز نویزدار خواهد بود و نشان‌دهنده توزیع واقعی نیست.

توضیحات بیشتر در زیر آمده است:  
چندین مشکل در استفاده از نرمال سازی دسته ای (BN) با اندازه دسته یک وجود دارد:

1. آمار ناپایدار(Unstable Statistics):  
BN بر محاسبه میانگین و واریانس دسته ورودی برای عادی سازی مقادیر فعال سازی تکی (individual activation values) تکیه دارد. با اندازه دسته یک، این آمار ناپایدار و بی معنی می شود. یک نقطه داده منفرد نمی تواند به طور دقیق توزیع کل مجموعه داده را نشان دهد، که منجر به عادی سازی غیرقابل اعتماد و به طور بالقوه مانع عملکرد مدل می شود. معمولا هر چه تهداد دسته بیشتر باشد نتایج قابل اعمتاد تر و با معنی تر می شوند.

2. از دست دادن استقلال(Loss of Independence):  
یکی از مزایای کلیدی BN این است که فعال‌سازی‌ها را در یک لایه مرتبط می‌کند و آنها را از یکدیگر مستقل می‌کند. این باعث بهینه سازی بهتر و آموزش سریعتر می شود. با این حال، با اندازه دسته ای یک، هر نقطه داده به صورت مجزا نرمال می شود، فرض استقلال را شکسته و به طور بالقوه مانع از فرآیند یادگیری می شود.

3. مشکلات عددی(Numerical Issues):  
BN شامل تقسیم بر جذر واریانس است. هنگامی که اندازه دسته یک است، واریانس بر اساس یک نقطه داده واحد محاسبه می شود که می تواند بسیار کوچک یا حتی صفر باشد. این می تواند منجر به بی ثباتی عددی شود، مانند تقسیم بر صفر خطا، که باعث می شود آموزش ناپایدار شود یا حتی با شکست مواجه شود.

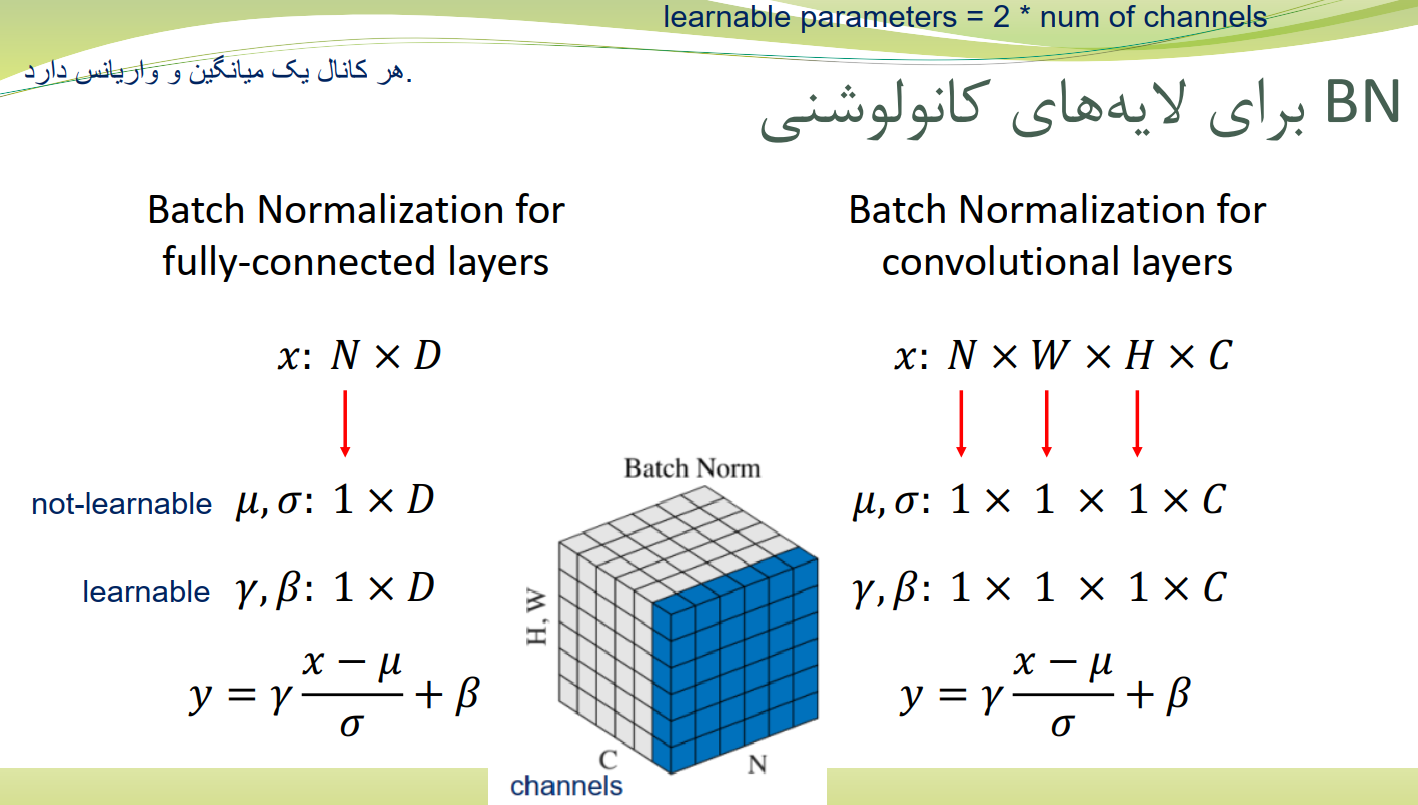
4. کاهش اثر منظم سازی(Reduced Regularization Effect):  
BN با عادی سازی فعال سازی ها و کاهش واریانس آنها به عنوان نوعی منظم سازی عمل می کند. این به جلوگیری از تطبیق بیش از حد مدل به داده های آموزشی کمک می کند. با این حال، با اندازه یک دسته، اثر تنظیم به طور قابل توجهی کاهش می‌یابد، و این مدل را مستعد برازش بیش از حد(overfitting) می‌کند.

5. محاسبات ناکارآمد(Inefficient Computation):  
BN شامل محاسبات اضافی برای محاسبه میانگین و واریانس دسته ورودی است. در حالی که این محاسبات برای اندازه‌های دسته‌ای بزرگ‌تر نسبتاً ارزان هستند(بدلیل موازی سازی)، برای اندازه دسته‌ای یک ناکارآمد می‌شوند، زیرا سربار بیشتر از مزایای آن است.

6. عدم تقریب گرادیان دسته ای کوچک(Lack of Mini-Batch Gradient Approximation):  
BN برای تخمین گرادیان واقعی تابع ضرر به تقریب شیب دسته ای کوچک(Mini-Batch Gradient Approximation) متکی است. با اندازه دسته‌ای یک، این تقریب نادرست می‌شود و منجر به یادگیری کمتر کارآمد و بالقوه نابهینه می‌شود.

به طور خلاصه، استفاده از BN با اندازه دسته ای یک به دلیل محدودیت ها و معایب احتمالی آن به طور کلی توصیه نمی شود. جایگزین هایی مانند عادی سازی لایه(Layer Normalization) یا عادی سازی گروه(Group Normalization) اغلب برای موقعیت هایی که اندازه دسته کوچک یا متغیر است ترجیح داده می شوند.

ه)



هنگام اضافه کردن نرمال سازی دسته ای (BN) به یک شبکه کاملاً متصل، تعداد پارامترهای قابل آموزش به دلیل پارامترهای اضافی معرفی شده توسط لایه BN افزایش می یابد. در اینجا نحوه محاسبه تعداد کل پارامترهای قابل آموزش آورده شده است:

نرمال سازی دسته ای معمولاً برای پیش فعال سازی ها(pre-activations) در یک لایه متراکم اعمال می شود که 20 مورد برای این لایه وجود دارد. گاما و بتا برای هر ورودی اسکالر هستند. به این ترتیب ما 20 گاما و 20 بتا داریم. این علاوه بر وزن‌های  
10 × 20 = 200 در لایه متراکم است (هنگامی که نرمال سازی دسته‌ای اعمال می‌شود، هیچ بایاسی وجود ندارد زیرا بتا آنها را اضافی می‌کند). پس در مجموع 240 پارامتر داریم.

شبکه اصلی:

* لایه ورودی: 10 نورون
* لایه خروجی: 20 نورون
* ماتریس وزن: 10 x 20 = 200 پارامتر قابل آموزش

اضافه کردن نرمال سازی دسته ای:

* بردار گاما(Gamma): 20 پارامتر قابل آموزش (یکی برای هر نورون خروجی)
* بردار بتا(Beta): 20 پارامتر قابل آموزش (یکی برای هر نورون خروجی)

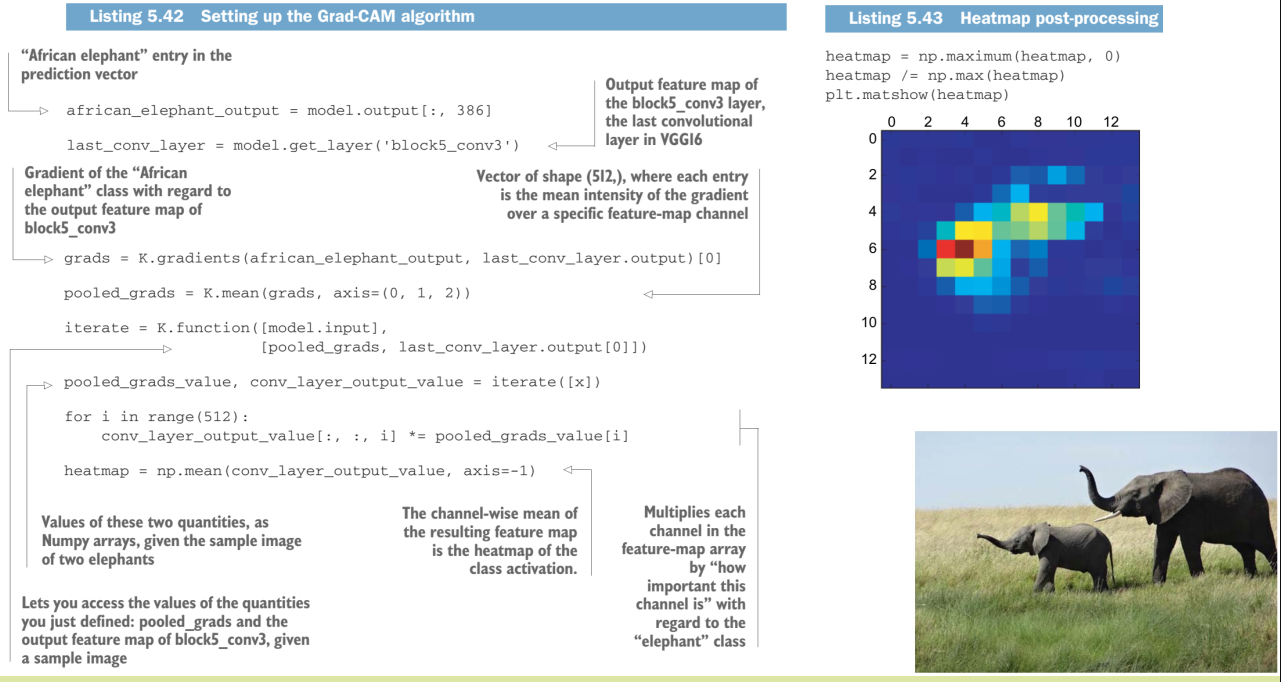
مجموع پارامترهای قابل آموزش:

* شبکه اصلی: 200
* لایه BN: 20 + 20 = 40
* مجموع: 200 + 40 = 240

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

## سوال هفتم



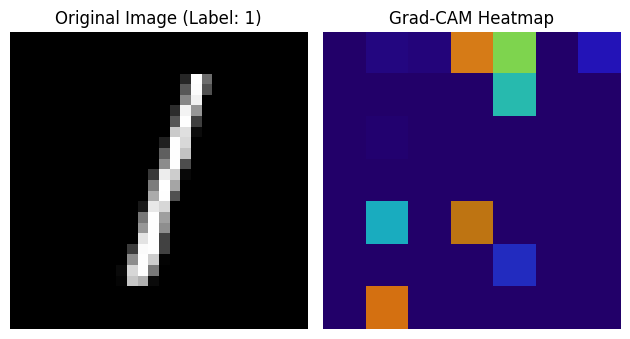
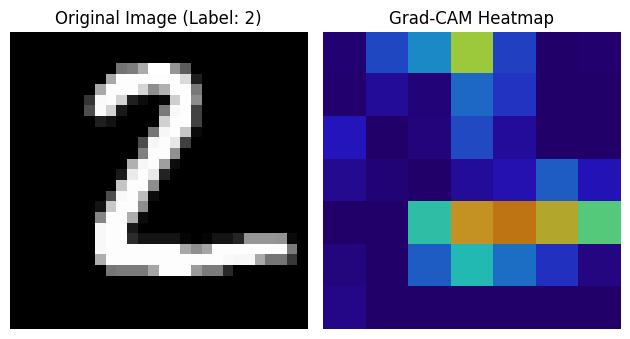
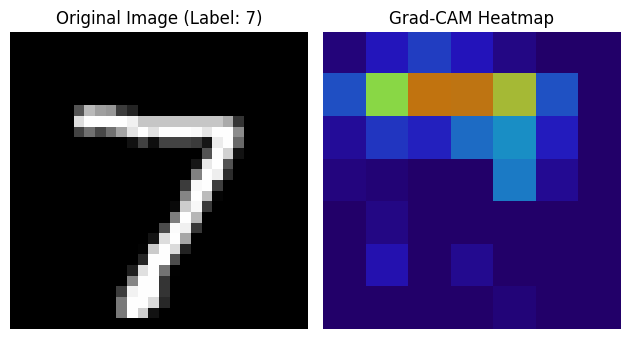
جواب این سوال به طور مفصل در DL\_HW4\_Q7.ipynb آمده است. با خواندن کامنت ها و markdown ها می توانید آن را به طور کامل آنالیز کنید. در اینجا به طور مختصر آن را توضیح می دهیم.

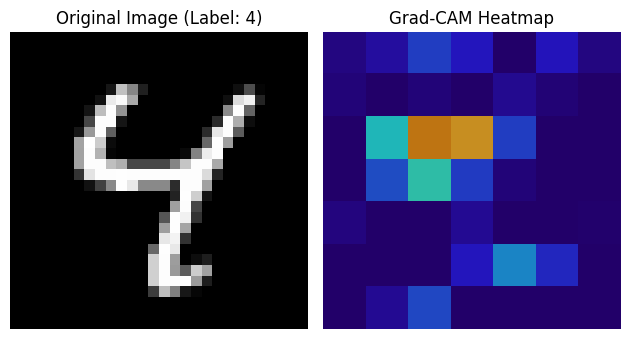
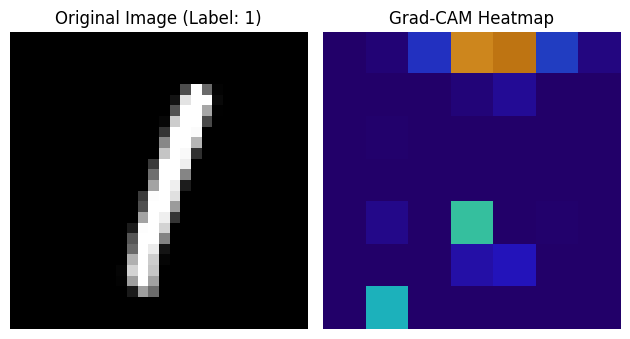
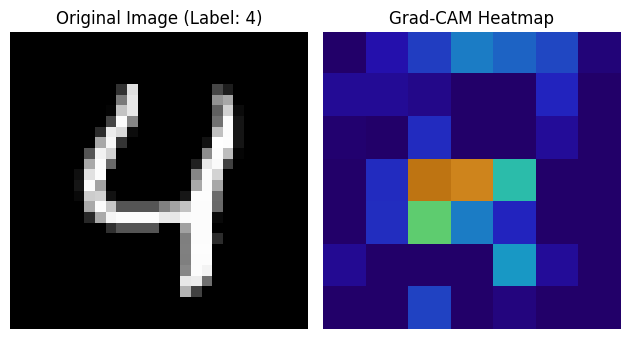
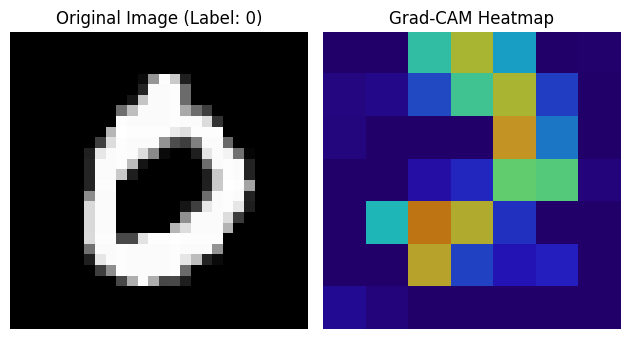
* ابتدا کتابخانه ها، توابع و کلاس های خواسته شده را import می کنیم.
* ابتدا به كمك *api* فراهم شده در كتابخانه *Keras* مجموعه داده *MNIST* را بارگذاري می نماییم.
* داده هاي آموزشي را *shuffle* میکنیم و ابعاد داده ها را چاپ می كنيم. همچنین از seed استفاده میکنیم تا ترتیب لیبل ها ها نیز به همان ترتیب باشد.
* ۱۰ تصوير نخست موجود در مجموعه داده آموزشي را به همراه برچسب آن ها نمايش می دهيم.
* مقادير پيكسل ها با تقسیم به 255 به محدوده ۰تا ۱نرمال سازی می کنیم.
* برچسب ها را به نمايش *categorical* تغيير می دهیم.(one hot)
* مدل خواسته شده را با توجه به معماری خواسته شده تعریف می کنیم و با استفاده از بهینه ساز آدام و تابع ضرر CategoricalCrossentropy آن را کامپایل میکنیم و سپس آن را روی مجموعه داده آموزشی، 15 دوره آموزش می دهیم.
* ساختار مدل را چاپ میکنیم.
* سپس با توجه به اسلاید ها و کمک از مدل های زبانی وسیع الگوریتم Grad-CAM را اجرا میکنیم و نتایج را چاپ میکنیم.

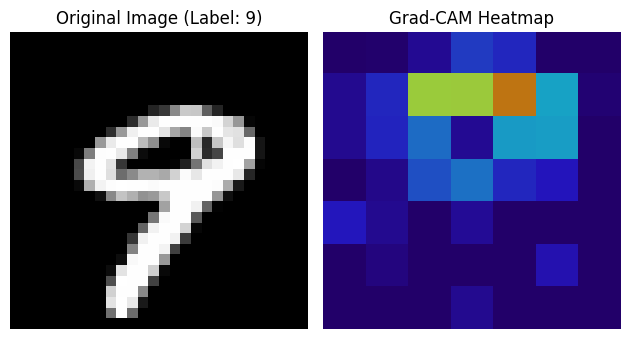
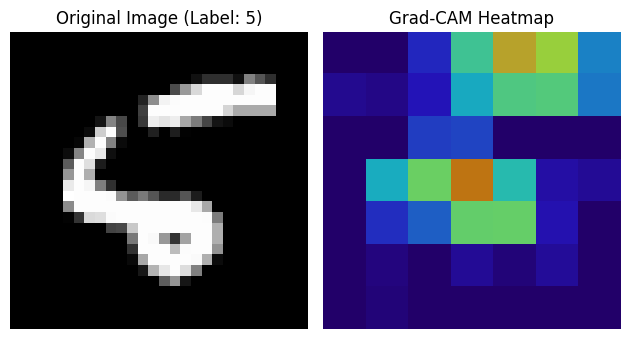
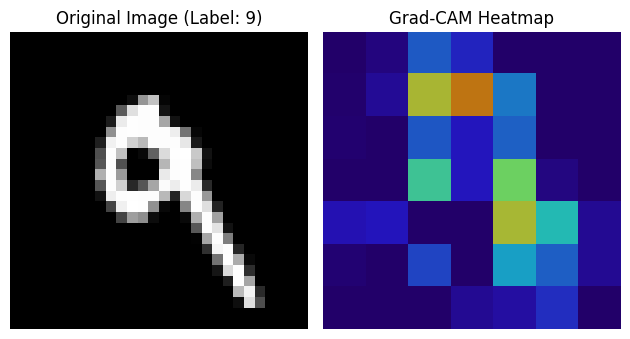
نقشه های حرارتی Grad-CAM مناطق تصویر برجسته مربوط به کلاس رقمی پیش بینی شده را برجسته می کند. می‌توانیم ببینیم که به جای پس‌زمینه، روی رقم واقعی تمرکز می‌کند و نشان می‌دهد که ویژگی‌های متمایز را شناسایی می‌کند. این می تواند به توضیح و تحلیل پیش بینی های مدل کمک کند.

نتیجه اجرای الگوریتم Grad-CAM بر روی آخرین لایه هم گشتی روی ده تصویر و تصویر متناظر آن در ادامه آمده است. به دنبال آن تحلیل و نتیجه استفاده از *Grad-CAM* در تشخيص ويژگي هاي مهم در تصاوير *MNIST* بررسی شده است.

نتیجه Grad-CAM:







ما الگوریتم Grad-CAM را برای تجسم ویژگی های مهم تصاویر از آخرین لایه کانولوشن اعمال می کنیم. ویژگی هایی که بیشترین تاثیر را در طبقه بندی تصویر به کلاس مطلوب دارند. می‌توانید مشاهده کنید که Grad-CAM چگونه مناطق مهمی را که توسط مدل برای پیش‌بینی استفاده می‌شود، برجسته می‌کند و به درک اینکه کدام مناطق در طبقه‌بندی ارقام در مجموعه داده MNIST مهم هستند کمک می‌کند. همان طور که دیده می شود پس زمینه کمترین اهمیت را دارند و مدل بیشتر بر اساس پیش زمینه به طبقه بندی می پردازد و پیکسل هایی که شامل رقم هستند بیش ترین اهمیت را دارند. همچنین میتوانیم مشاهده کنیم کهمدل بر اساس کدام پیکسل ها بین ارقام مختلف تمیاز قائل میشود.

چند نکته دیگر برای تجزیه و تحلیل خروجی Grad-CAM وجود دارد:

* به دنبال نواحی با شدت بالا باشیم: این نواحی با ویژگی هایی مطابقت دارند که مدل آن ها را برای تشخیص رقم در تصویر مهم می داند.
* مقایسه نقشه های حرارتی برای تصاویر مختلف: این می تواند به ما کمک کند تا بفهمیم مدل چگونه بین ارقام مختلف تفاوت قائل می شود.
* مقایسه نقشه های حرارتی با ویژگی های واقعی: به تصویر اصلی نگاه کنید و ببینید آیا مناطق برجسته شده توسط Grad-CAM با ویژگی های واقعی رقم مانند لبه ها، منحنی ها یا گوشه ها مطابقت دارد یا خیر.

با تجزیه و تحلیل خروجی Grad-CAM، می توانیم بینشی در مورد نحوه عملکرد مدل به دست آوریم و ویژگی های مهم برای تشخیص رقم را شناسایی کنیم. این می تواند برای بهبود عملکرد مدل و توسعه درک بهتر از نحوه عملکرد شبکه های عصبی کانولوشنی مفید باشد.

مراجع:

<https://chat.openai.com/>   
<https://bard.google.com/>   
<https://claude.ai/chats>

لینک چت های من با chat gpt:

<https://chat.openai.com/share/28e94fdc-da59-472d-b94a-bd377cacf4ea>  
<https://chat.openai.com/share/9aa36a48-f1ad-411b-af6e-29e5328868eb>

پایان