به نام خدا

تمرین سری چهارم

درس یادگیری عمیق

سینا علی‌نژاد 99521469

سوال 1-

الف) طراحی شبکه همگشتی با استفاده از KerasTuner:

* KerasTuner یک ابزار قدرتمند در TensorFlow است که به تنظیم هایپرپارامتر برای شبکه های عصبی، به ویژه برای مدل های Keras کمک می کند. این ابزار فرآیند جستجوی بهترین ترکیب از هایپرپارامترها را خودکار می کند و در زمان و تلاش شما صرفه جویی می کند. KerasTuner به‌جای آزمایش دستی ترکیب‌های مختلف، از الگوریتم‌های جستجوی مختلف استفاده می‌کند تا به طور مؤثر پیکربندی را پیدا کند که بهترین عملکرد را در داده‌های خاص شما دارد.
* جواب Bard:

KerasTuner can significantly optimize convolutional neural networks (CNNs) for image classification tasks. Here's how it works:

* Define the Model Architecture: You define the basic structure of your CNN using Keras layers like Conv2D, MaxPooling2D, Flatten, and Dense.
* Hyperparameter Search Space: You specify the range of values for hyperparameters like number of filters, kernel size, activation functions, optimizers, and learning rate.
* Search Algorithm and Tuner: You choose a search algorithm like RandomSearch or Bayesian Optimization and a tuner like Hyperband or RandomTuner.
* Training and Evaluation: KerasTuner trains multiple models with different hyperparameter combinations on your data and evaluates them based on a chosen metric (accuracy, F1 score, etc.).
* Best Model Selection: The tuner selects the model with the best performance metric and provides you with the optimal hyperparameter values.

که به طور خلاصه می‌گوید:

* + ابتدا مدل را در Keras تعریف میکنیم.
  + سپس برای هر هایپرپارامتر که میخواهیم بهترین مقدار آن پیدا شود، محدوده جستجو را مشخص میکنیم.
  + سپس الگوریتم جستجو و همچنین یک تنظیم کننده (Tuner) را مشخص میکنیم.
  + مدل را با استفاده از KerasTuner آموزش می‌دهیم و ارزیابی می‌کنیم.
  + در نهایت، بهترین مدل از لحاظ performance توسط tuner انتخاب می‌شود.
* KerasTuner تونرهای مختلفی را ارائه می دهد که هر کدام نقاط قوت و ضعف خود را دارند. در اینجا چند انتخاب وجود دارد:
  + Hyperband: به طور موثر فضای جستجوی بزرگ را با توقف زودهنگام (early stopping) کاوش می‌کند که برای مجموعه داده‌های بزرگ مناسب است.
  + RandomSearch: ساده و سریع، برای مجموعه داده های کوچک یا کاوش اولیه مناسب است.
  + Bayesian Optimization: قدرتمند اما از نظر محاسباتی گران است.

انتخاب من برای Tuner برای دیتاست MNIST، همان Hyperband است به دلایل زیر:

* + Balanced Exploration: این دیتاست نسبت به دیتاست‌های دیگری مثل ImageNet کوچک بشمار می‌آید. این تونر، به طور موثری فضای جستجو را کاوش می‌کند بدون اینکه منابعی را بر روی پیکربندی‌هایی که امیدی به آنها نیست، هدر بدهد.
  + Early stopping: آموزش مدلهایی که پیشرفت آنها نامحتمل است را متوقف می‌کند که باعث صرفه جویی در زمان و منابع می‌شود که این موضوع در دیتاست‌های کوچک مثل MNIST بسیار موثر است.
  + Good performance: نشان داده شده است که Hyperband روی دیتاست MNIST خوب عمل می‌کند و مدلهایی با دقت بالا و قدرت تعمیم‌دهی بالا می‌دهد.

جواب Bard:

Pros of Hyperband for MNIST:

* Balanced exploration: MNIST is a relatively small dataset compared to others like ImageNet. Hyperband efficiently explores the search space without wasting resources on unpromising configurations.
* Early stopping: It stops training models that are unlikely to improve, saving you time and computational resources. This is particularly valuable for smaller datasets like MNIST.
* Good performance: Hyperband has been shown to perform well on MNIST, often finding high-performing models with good accuracy and generalization.

ب) استفاده از KerasTuner بر روی دیتاست MNIST:

* مجموعه داده MNIST یک معیار (benchmark) کلاسیک برای طبقه بندی تصاویر است که از 70000 تصویر خاکستری از ارقام دست نویس (0-9) با برچسب های مربوطه تشکیل شده است. هر تصویر 28\*28 پیکسل است که آن را به مجموعه ای نسبتا کوچک و قابل مدیریت تبدیل می کند. MNIST به طور گسترده ای برای ارزیابی و مقایسه الگوریتم های مختلف یادگیری ماشین، از جمله شبکه های عصبی کانولوشنی (CNN) استفاده می شود.
* KerasTuner یک چارچوب تنظیم فراپارامتر است که یافتن پیکربندی بهینه برای CNN شما را ساده می کند. در اینجا نحوه استفاده از آن با MNIST آمده است:
  + تعریف مدل: یک تابع ساخت مدل را تعریف کنید که فراپارامترها (به عنوان مثال، تعداد لایه ها، فیلترها در هر لایه، توابع فعال سازی) را به عنوان ورودی می گیرد و بر اساس آن یک CNN می سازد.
  + فضای جستجو: محدوده مقادیری را که می خواهید برای هر هایپرپارامتر بررسی کنید، مشخص کنید.
  + تنظیم ابرپارامترها: از بهینه سازی بیزی یا الگوریتم های دیگر برای جستجوی خودکار فضای تعریف شده و یافتن ترکیبی که منجر به بهترین عملکرد در مجموعه اعتبار سنجی می شود استفاده می کند.
  + آموزش و ارزشیابی: پس از شناسایی فراپارامترهای بهینه، مدل نهایی بر روی کل مجموعه آموزشی آموزش داده می شود و بر روی مجموعه آزمون ارزیابی می شود.
* اهمیت لایه‌های Dropout و Pooling:

Dropout: این تکنیک به طور تصادفی نورون‌ها را در طول تمرین حذف می‌کند، از تطبیق بیش از حد جلوگیری می‌کند و شبکه را تشویق می‌کند تا ویژگی‌های قوی‌تری را بیاموزد. همچنین به عنوان یک تنظیم کننده عمل می کند و پیچیدگی کلی مدل را کاهش می دهد.

Pooling: این عملیات ابعاد فضایی نقشه های ویژگی را کاهش می دهد و به شبکه امکان می دهد ویژگی ها را در مقیاس های مختلف یاد بگیرد و روی تکه های تصویر کوچکتر پیش بینی کند. همچنین تعداد پارامترها را کاهش می دهد و کارایی محاسباتی را بهبود می بخشد.

تاثیرات این لایه ها در عملکرد مدل:

کاهش Overfitting: Dropout به شبکه کمک می‌کند تا داده‌های دیده نشده را بهتر تعمیم دهد، که منجر به دقت بالاتر در مجموعه آزمایشی می‌شود.

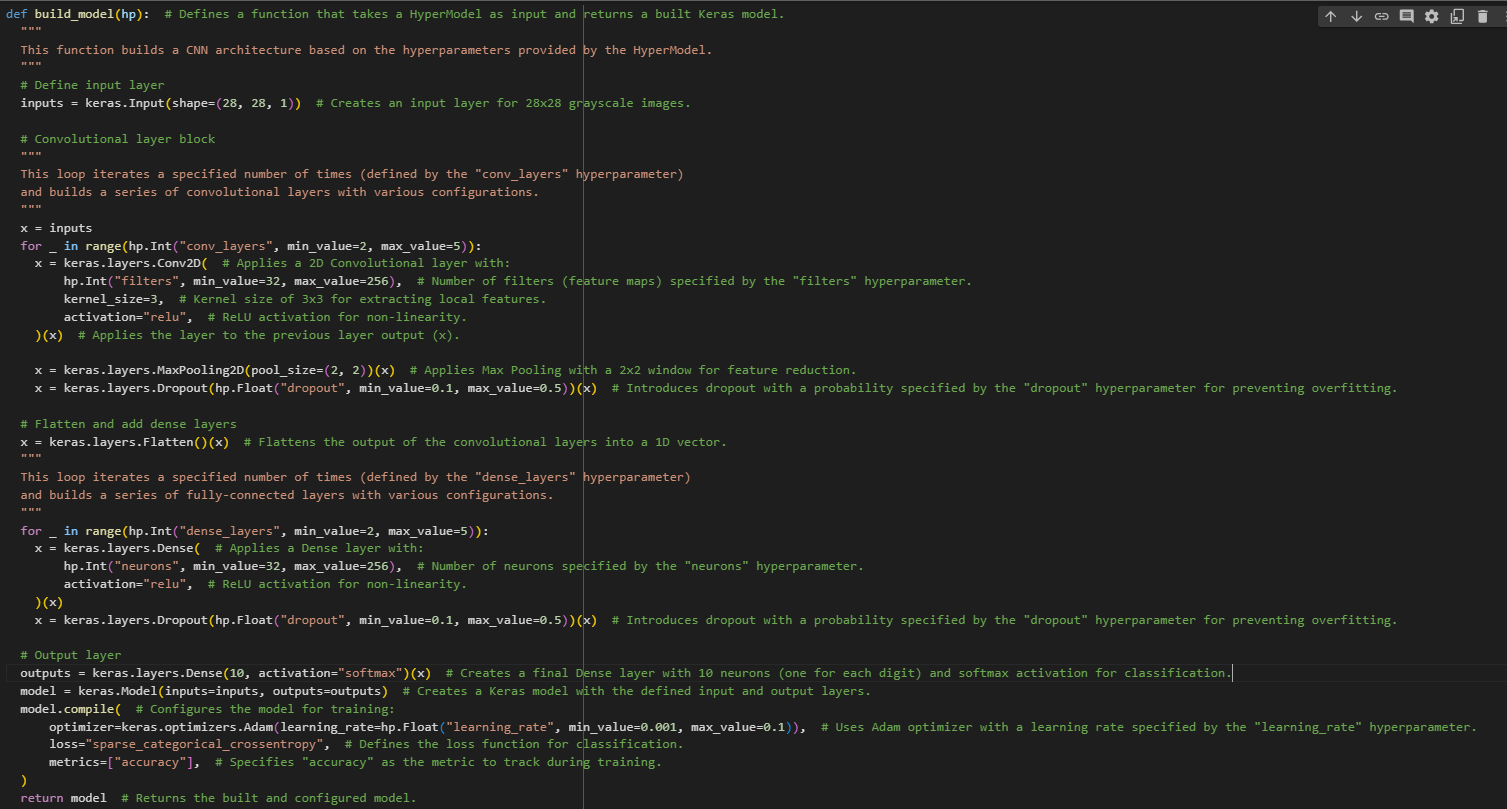
یادگیری ویژگی های بهبود یافته: ادغام به شبکه اجازه می دهد تا ویژگی های عمومی و قوی تری را به تصویر بکشد که منجر به عملکرد بهتر در اشکال و تغییرات مختلف رقم می شود.

کارایی محاسباتی: ادغام تعداد پارامترها و عملیات را کاهش می‌دهد و باعث می‌شود مدل سریع‌تر آموزش و اجرا شود.

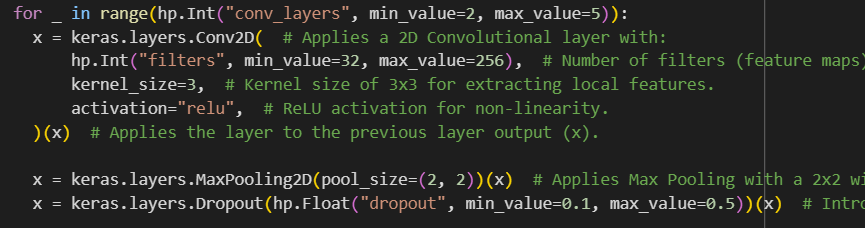
ج) پیاده سازی شبکه مورد نظر:

* پیاده سازی:

تابع build\_model:

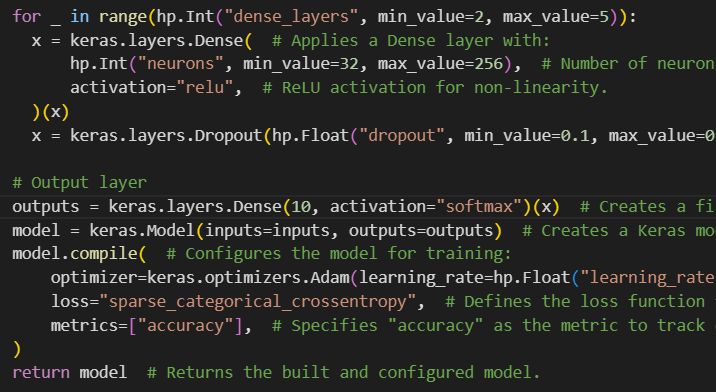


این تابع مدل ما را بر اساس هایپرپارامترهایی که به آن میدهیم و این کار را با استفاده از keras tuner انجام میدهیم و فضای جستجو را با پارامترهای min value و max value برای پارامترهایی که میخواهیم مشخص میکنیم.

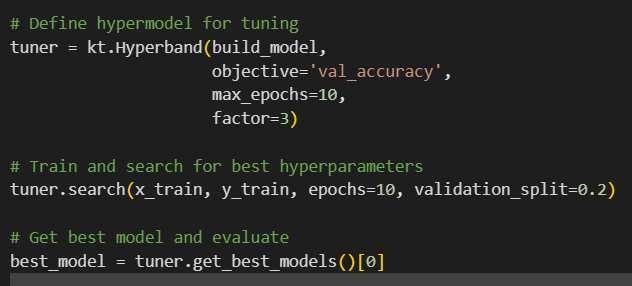


برای مثال در اینجا تعداد لایه های کانولوشنی توسط عددی که به تابع range داده میشود، مشخص میشود و ما این عدد را یک عدد صحیح بین 2 تا 5 قرار میدهیم و این کار tuner خواهد بود که مقدار بهینه را پیدا کند.

همچنین بعد از هر لایه کانولوشنی، یک لایه max pooling و بعد از آن لایه dropout داریم که مقدار keep probabality را به tuner سپردیم، عددی بین 0.1 و 0.5 .

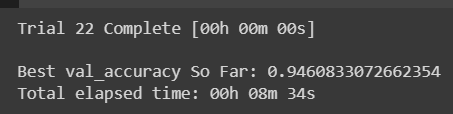


بعد از آن نوبت لایه های کاملا متصل است. تعداد این لایه ها نیز بین 2 تا 5 تعیین میشود و در آخر یک لایه خطی نهایی برای دسته بندی نهایی میگذاریم.

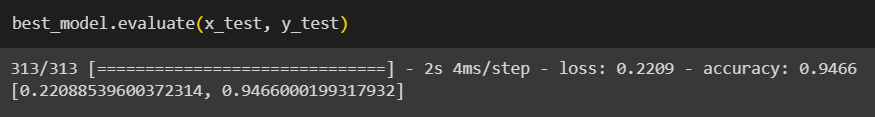


در اینجا نوع tuner را مشخص کردیم. دلیل استفاده از hyperband را در قسمت های قبلی گفتم. پارامتر اول همان تابعی است که مدل را میسازد و پارامتر دوم، معیار tuner برای یافتن بهترین مقادیر ابرپارامترها است، پارامتر سوم تعیین میکند که ماکزیمم تا 10 epoch برای هر پیکربندی پیش برود و بهترین ها را انتخاب کند.

در نهایت جستجو توسط تابع search انجام میشود و بهترین مدل را توسط تابع get\_best\_models میگیریم.



ارزیابی بر روی داده تست:



برای هر خط در کد کامنت گذاری دقیقتری شده است.

* اندازه فیلترها در لایه های کانولوشنی یک ابرپارامتر مهم است که می تواند بر عملکرد یک شبکه عصبی کانولوشن (CNN) تأثیر بگذارد. انتخاب اندازه فیلتر به مشکل موجود و ویژگی های داده های ورودی بستگی دارد.

محبوب ترین انتخاب برای اندازه فیلتر 3\*3 است. این به این دلیل است که اندازه فیلترهای کوچکتر هزینه های محاسباتی و اشتراک وزن را کاهش می دهد که در نهایت منجر به وزن کمتر برای back propagation می شود. اگر مقدار زیادی پیکسل برای شبکه برای تشخیص شی مورد نیاز باشد، ممکن است از اندازه فیلترهای بزرگتر، مانند 11\*11 یا 9\*9 استفاده شود. اگر قرار است تفاوت‌های موضعی بیشتر مورد توجه قرار گیرند، اندازه‌های فیلتر کوچک‌تر، مانند 3\*3 یا 5\*5 ترجیح داده می‌شوند.

به طور کلی، هیچ قانون سختی برای انتخاب اندازه فیلترها در لایه های کانولوشن وجود ندارد. بهتر است با اندازه های مختلف فیلتر آزمایش کنیم و ببینیم چه چیزی برای مشکل خاص ما بهتر است.

برای مثال در دیتاست MNIST تعداد کل پیکسلهای تصویر 28\*28 است بنابراین استفاده از فیلتر با سایز بزرگ، ناگهان ابعاد تصویر را کاهش میدهد و نمیتواند خوب عمل کند برای همین از سایز 3\*3 استفاده کردم.

* ادغام و dropout دو تکنیکی هستند که می توانند برای جلوگیری از بیش برازش(overfitting) و افزایش دقت در شبکه های عصبی کانولوشن (CNN) استفاده شوند.

ادغام تکنیکی است که اندازه مکانی نقشه های ویژگی را با نمونه برداری از آنها کاهش می دهد. این می تواند به کاهش تعداد پارامترها در شبکه و جلوگیری از بیش برازش کمک کند. انواع مختلفی از لایه های ادغام وجود دارد، مانند ادغام ماکزیمم و ادغام متوسط، که می توانند در یک CNN استفاده شوند.

Dropout یک تکنیک منظم سازی است که به طور تصادفی برخی از نورون های شبکه را در طول آموزش حذف می کند. این می‌تواند با کاهش co-adaption نورون‌ها و وادار کردن شبکه به یادگیری ویژگی‌های قوی‌تر، به جلوگیری از بیش برازش کمک کند. Dropout را می توان با استفاده از یک لایه اضافی بر روی لایه های کانولوشن در یک CNN اعمال کرد، که کل برخی نقشه های ویژگی را از لایه کانولوشنال حذف می کند.

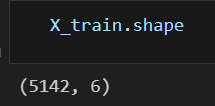
به طور کلی، ادغام و dropout می توانند با هم برای بهبود عملکرد CNN استفاده شوند. ادغام می‌تواند به کاهش اندازه فضایی نقشه‌های ویژگی کمک کند، در حالی که dropout می‌تواند با کاهش co-adaption نورون‌ها به جلوگیری از overfitting کمک کند. با این حال، توجه به این نکته مهم است که ترکیب بهینه ادغام و dropout به مشکل خاص در دست و ویژگی های داده های ورودی بستگی دارد.

سوال 3-

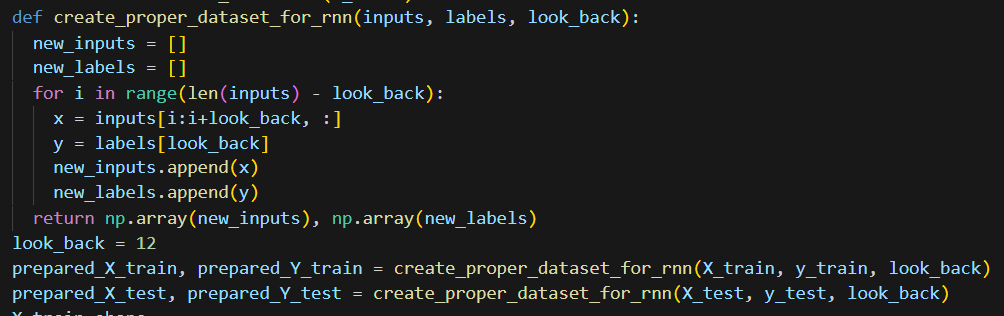
طبق این [لینک](https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/)، دیتای ورودی که یک لایه LSTM از ما انتظار دارد، به فرم زیر است:

[samples, time steps, features]

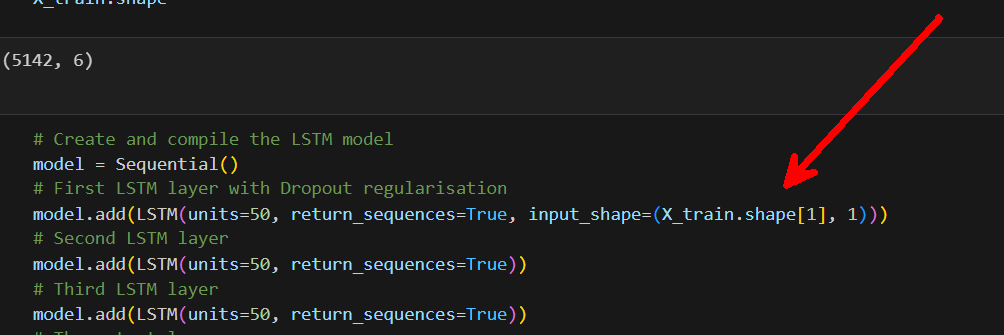
بعد اول تعداد نمونه ها و بعد دوم تعداد استیت های قبلی برای پیش بینی استیت فعلی و بعد سوم تعداد فیچرهای هر کدام از استیت ها است که در مسئله پیش بینی گاز مصرفی، تعداد آن 6 است. در این مسئله تعداد نمونه‌های آموزشی 5142 است و ابعاد دیتاست آموزشی به صورت زیر است:



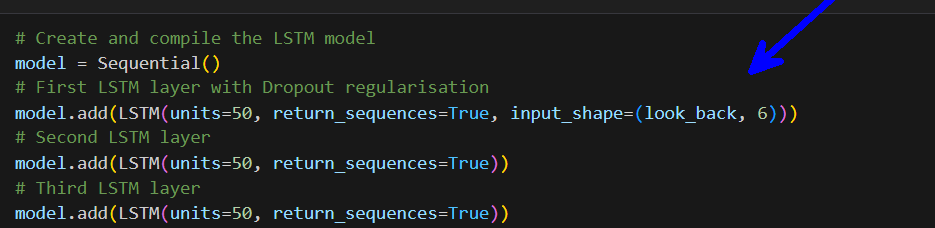
در واقع ما باید یک تابعی بنویسیم که این دیتاست را بگیرد و یک دیتاست مناسب شبکه بازگشتی ایجاد کند، ویژگی این دیتاست، وجود پنجره زمانی است، یعنی ما مصرف گاز در چند ساعت گذشته را داشته باشیم و با استفاده از آنها و ویژگی های زمان حاضر، پیش بینی جدید را انجام دهیم. کد این تابع به صورت زیر خواهد بود:



که پارامتر look\_back اشاره به تعداد بازه زمانی قبلی دارد که برای پیش بینی مقدار زمان حاضر استفاده میشود. پس از اعمال این تابع، ابعاد دیتاست به صورت روبرو میشود: [samples, look\_back, num of features] که تعداد فیچرها در اینجا عدد 6 است و بهترین مقدار look\_back باید با آزمون و خطا بدست آید. برای مثال در اینجا مقدار آن را برابر با 12 گذاشتم که یعنی مقادیر 12 ساعت اخیر را در محاسبه خود دخالت دهد.



همچنین قسمت مشخص شده در بالا باید به شکل زیر تغییر کند:



که عدد 6 همان تعداد ویژگی هاست.

همچنین best practice این است که دیتای ورودی به LSTM نرمالایز باشد که با تغییر زیر در کد امکان پذیر است:



سوال 4-

الف) شبکه های عصبی کانولوشن (CNN) و شبکه های عصبی بازگشتی (RNN) دو نوع شبکه عصبی هستند که معمولاً در یادگیری عمیق استفاده می شوند. CNN ها برای پردازش داده ها با توپولوژی شبکه مانند (grid)، مانند تصاویر، طراحی شده اند، در حالی که RNN ها برای پردازش داده های متوالی، مانند متن یا گفتار طراحی شده اند.

CNN ها برای طبقه بندی تصویر، تشخیص اشیا و image segmentation به خوبی کار می کنند زیرا می توانند الگوهای موجود در ساختار فضایی داده های ورودی را به صورت محلی تشخیص دهند. آنها همچنین از نظر محاسباتی کارآمد هستند زیرا از اشتراک پارامتر برای کاهش تعداد پارامترها در مدل استفاده می کنند.

RNN ها برای وظایف پردازش زبان طبیعی، مانند مدل سازی زبان، ترجمه ماشینی، و تشخیص گفتار به خوبی کار می کنند، زیرا می توانند مدل سازی وابستگی های زمانی در داده های ورودی را بیاموزند. آنها همچنین قادر به مدیریت توالی های ورودی با طول متغیر هستند که باعث انعطاف پذیری بیشتری نسبت به CNN ها می شود.

ب) CNN و RNN از نظر تعداد پارامترها و امکان موازی سازی با هم تفاوت دارند. CNN ها معمولاً پارامترهای کمتری نسبت به RNN دارند زیرا از اشتراک پارامتر برای کاهش تعداد پارامترها در مدل استفاده می کنند. این باعث می شود که CNN ها از نظر محاسباتی کارآمدتر و آموزش آسان تر از RNN ها باشند.

از سوی دیگر، موازی سازی RNN ها نسبت به CNN ها دشوارتر است، زیرا آنها به خروجی مرحله زمانی قبلی به عنوان ورودی به مرحله زمانی فعلی نیاز دارند. این امر پردازش چندین مرحله زمانی را به صورت موازی دشوار می کند. با این حال، پیشرفت های اخیر در سخت افزار و نرم افزار، موازی سازی RNN ها را تا حدودی ممکن کرده است.

به طور خلاصه، CNN ها برای وظایفی که شامل ساختار فضایی هستند، مناسب تر هستند، در حالی که RNN ها برای وظایفی که شامل وابستگی های زمانی هستند، مناسب تر هستند. CNN ها از نظر محاسباتی کارآمدتر هستند و آموزش آنها آسان تر از RNN ها است، اما RNN ها انعطاف پذیرتر هستند و می توانند توالی های ورودی با طول متغیر را مدیریت کنند.

پرامپت:

a) Give a brief definition of convolutional networks and recursive networks and say for what applications and problems each one works better and why. b) Compare these networks with each other regarding the number of parameters and the possibility of parallelization.

جواب bing:

a) Convolutional neural networks (CNNs) and recursive neural networks (RNNs) are two types of neural networks that are commonly used in deep learning. CNNs are designed to process data with a grid-like topology, such as images, while RNNs are designed to process sequential data, such as text or speech.

CNNs work well for image classification, object detection, and image segmentation tasks because they can learn to recognize patterns in the spatial structure of the input data. They are also computationally efficient because they use parameter sharing to reduce the number of parameters in the model.

RNNs work well for natural language processing tasks, such as language modeling, machine translation, and speech recognition, because they can learn to model the temporal dependencies in the input data. They are also capable of handling variable-length input sequences, which makes them more flexible than CNNs.

b) CNNs and RNNs differ in terms of the number of parameters and the possibility of parallelization. CNNs typically have fewer parameters than RNNs because they use parameter sharing to reduce the number of parameters in the model. This makes CNNs more computationally efficient and easier to train than RNNs.

On the other hand, RNNs are more difficult to parallelize than CNNs because they require the output of the previous time step as input to the current time step. This makes it difficult to process multiple time steps in parallel. However, recent advances in hardware and software have made it possible to parallelize RNNs to some extent.

In summary, CNNs are better suited for tasks that involve spatial structure, while RNNs are better suited for tasks that involve temporal dependencies. CNNs are more computationally efficient and easier to train than RNNs, but RNNs are more flexible and can handle variable-length input sequences.

سوال 5-

الف) فرمولی که همه پارامترهای stride,padding,dilation rate را در محاسبه ابعاد خروجی در نظر بگیرد، به شکل زیر است:

که P همان مقدار padding است که از هر طرف چقدر به آن اضافه میشود. D مقدار dilate کردن و Wk عرض کرنل است و S مقدار stride است.

خروجی لایه اول:

256\*256\*64 (because of padding=same, input spatial dimensions will stay unchanged)

تعداد پارامتر لایه اول:

64\* (3\*3\*3 + 1) = 1792

خروجی لایه دوم:

We use the formula provided earlier

Win = 256, S=2, D=2, P=0(padding=valid), Wk = 5

Wout = 124 and Hout = Wout = 124

Output shape = 124\*124\*32

تعداد پارامتر لایه دوم:

32\*(5\*5\*64 + 1) = 51232

خروجی لایه سوم:

62\*62\*32

تعداد پارامتر لایه سوم: صفر

خروجی لایه چهارم:

62\*62\*128 (spatial dimensions unchanged due to padding=same

تعداد پارامتر لایه چهارم:

128\*(3\*3\*32 + 1) = 36992

خروجی لایه پنجم:

S=2, D=4, Wk=5, Win=62, P=0

Wout = 23, Hout = Wout = 23

Output shape: 23\*23\*64

تعداد پارامتر لایه پنجم:

64\*(5\*5\*128 + 1) = 204864

خروجی لایه ششم:

11\*11\*64

تعداد پارامتر لایه ششم: صفر

خروجی لایه هفتم:

11\*11\*256 (because of padding=same)

تعداد پارامتر لایه هفتم:

256\*(3\*3\*64 + 1) = 147712

خروجی لایه هشتم:

S=2, D=8, P=0, Wk=5, Win=11

Wout = -10

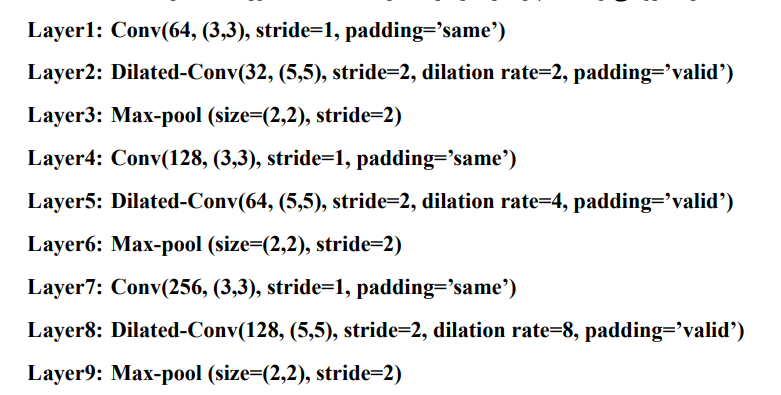
Output shape: -10\*-10\*128

تعداد پارامتر لایه هشتم:

128\*(5\*5\*256 + 1) = 819328

خروجی لایه نهم: خروجی لایه قبل منفی شد و معنا ندارد. وگرنه دو بعد اول نصف و بعد سوم ثابت میماند.

تعداد پارامتر لایه نهم: صفر



ب) فرمول P از رابطه زیر محاسبه میشود:

که مقدار محاسبه شده مقداری از پدینگ است که باید به هر طرف (بالا، پایین، چپ، راست) اضافه شود.

اگر مقدار f عددی زوج بود، در آن صورت، جمع پدینگ های چپ و راست باید برابر با f-1 شود، همینطور برای بالا و پایین.

سوال 6-

الف)

* غلط است. پردازش را سریع نمیکند زیرا یک سری پردازش بیشتر اضافه میکند(البته در زمان تست، این پردازش اضافه میتواند در لایه قبل از BN ادغام شود ولی در زمان آموزش قطعا پردازش اضافه‌ای مثل محاسبه میانگین و واریانس دسته برای هر ویژگی را داریم و باید moving avg و moving var را نیز برای زمان تست محاسبه کنیم، همچنین یک لایه خطی اضافه تر با پارامترهای گاما و بتا داریم پس تعداد به روز رسانی‌ها از این جهت در back propagation افزایش پیدا میکند و ثابت نمی‌ماند. البته نرمال سازی دسته ای باعث میشود همگرایی بهتر و سریعتر صورت بگیرد ولی زمان هر forward\_pass را در زمان آموزش افزایش میدهد.
* درست است.
* غلط است. نرمالسازی دسته ای چنین کاری انجام نمیدهد.

ب) کد زده شد.

ج) وجود مقدار اپسیلون بخاطر کدزنی است، ممکن است یک ویژگی برای تمام ورودی های آن دسته مقدار یکسان داشته باشد و واریانس آن صفر شود، در این صورت در فرمول زیر اگر اپسیلون اضافه نشود، مقدار مخرج برابر با صفر خواهد بود و به Division by zero exception میخوریم. اضافه کردن اپسیلون برای جلوگیری از این موضوع است.



د) نرمال‌سازی دسته‌ای تکنیکی است که برای سریع‌تر و پایدارتر کردن آموزش شبکه‌های عصبی مصنوعی از طریق rescale و re-center کردن مقادیر دسته ورودی استفاده می‌شود. با این حال، استفاده از نرمال سازی دسته ای با اندازه دسته‌ی یک می تواند منجر به مشکلاتی شود.

هنگامی که اندازه دسته یک است، واریانس صفر خواهد بود و تقسیم بر صفر در هنگام نرمالسازی منجر به خطا می شود. با این حال، برخی از محققان پیشنهاد کرده اند که این خطا ممکن است به دلیل دو مورد رخ ندهد. اول، خطا در یک بلوک try-catch رخ دهد. دوم، یک عدد گویا کوچک (1e-19) به عبارت واریانس اضافه می شود تا هرگز صفر نشود.

با این حال، استفاده از نرمال سازی دسته ای با اندازه دسته ای یک توصیه نمی شود. دلایل نظری قوی علیه آن وجود دارد، و مقالات متعدد نشان داده‌اند که عملکرد نرمال‌سازی دسته‌ای برای اندازه‌های دسته‌ای کمتر از 32 و به خصوص برای اندازه‌‌ی دسته‌ای کمتر یا مساوی 8 کاهش می‌یابد. همچنین معیارهای moving avg و moving var در این حالت از مقدار واقعی خود فاصله دارند.

ه)

محاسبه تعداد پارامترهای خود لایه خطی با 20 نورون:

20 \* (10(input dimension) + 1(one bias for each neuron)) = 220

محاسبه تعداد پارامترهای لایه نرمالسازی:

2(gamma,beta) \* 20(input channel dimension) = 40

Because each channel has its own gamma and beta

پس تعداد کل پارامترهای قابل آموزش برابر است با:

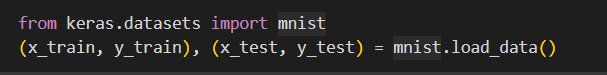
220 + 40 = 260

البته میتوان برای لایه Dense در اینجا بایاس در نظر نگرفت، زیرا در لایه BN بایاس داریم که میتواند با لایه خطی قبل از خودش ترکیب شود. در آن صورت 20 تا از پارامترها کم خواهد شد و 240 پارامتر قابل آموزش خواهد ماند.

سوال 7-

الف)

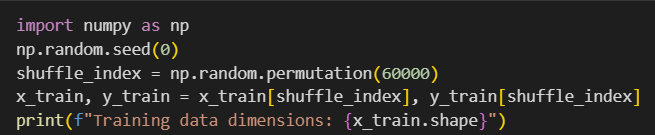
دانلود دیتاست mnist:



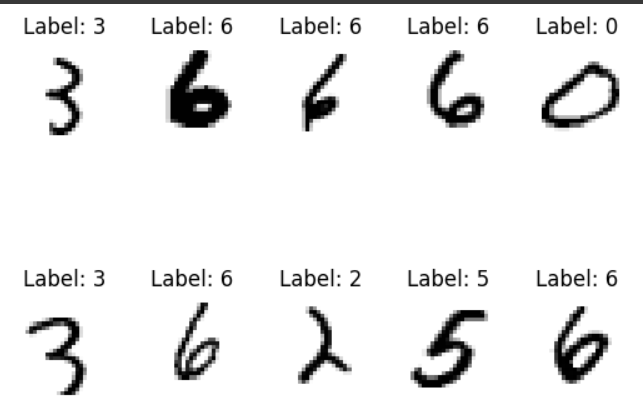
بر زدن دیتای آموزشی:

تنظیم seed باعث میشود نتایج قابل تولید دوباره باشند.

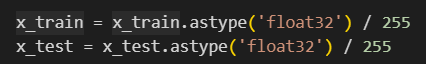
از مقدار shuffle\_index برای بر زدن هردوی ورودی و برچسب ها استفاده میشود تا هر برچسب ایندکس یکسانی با دیتای ورودی مربوط به خودش باشد.



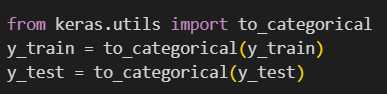
نمایش چند عکس با برچسب مربوط به آن به کمک matplotlib:



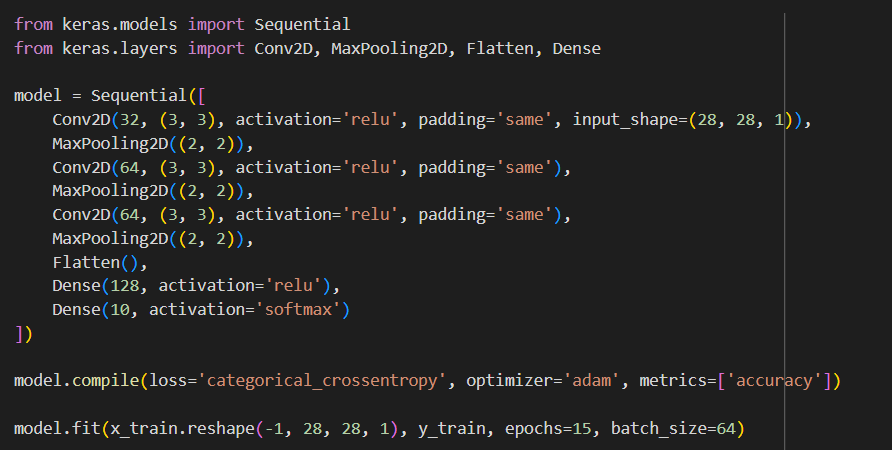
با کد زیر، دیتاست آموزشی و تست را نرمالایز میکنیم تا فرایند آموزش بهتر و سریعتر صورت بگیرد:



از کد زیر برای one-hot encode کردن برچسب ها استفاده میشود تا تابع ضرر categorical cross entropy روی آن کار کند:

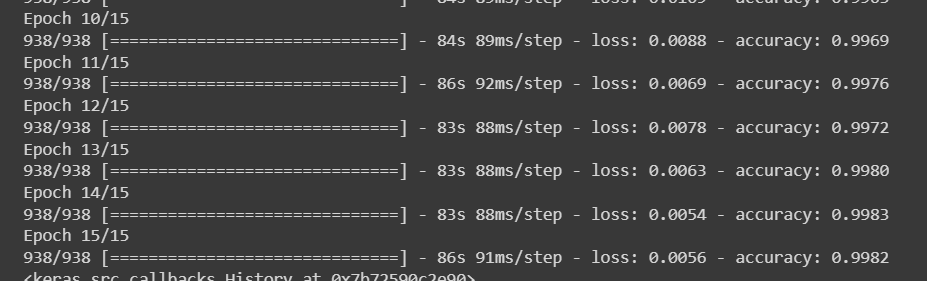


تعریف مدل بر اساس لایه های ذکر شده در سوال:

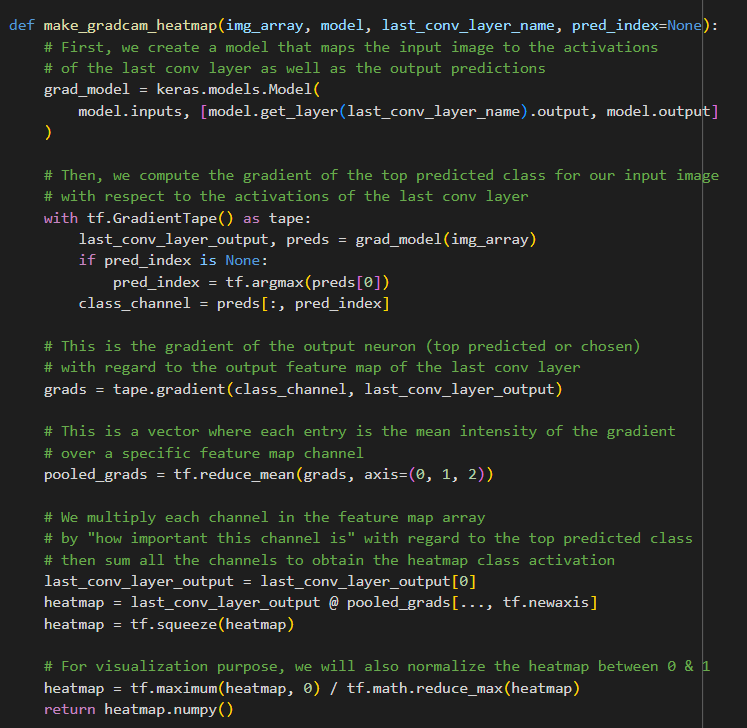


استفاده از بهینه ساز Adam

حاصل آموزش:

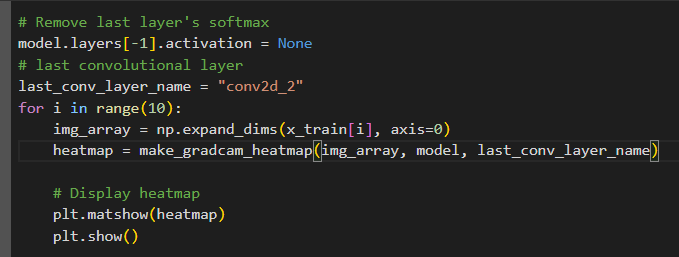


ب) برای الگوریتم Grad-CAM من از این [لینک](https://keras.io/examples/vision/grad_cam/) استفاده کردم و کد الگوریتمش به شکل زیر است:

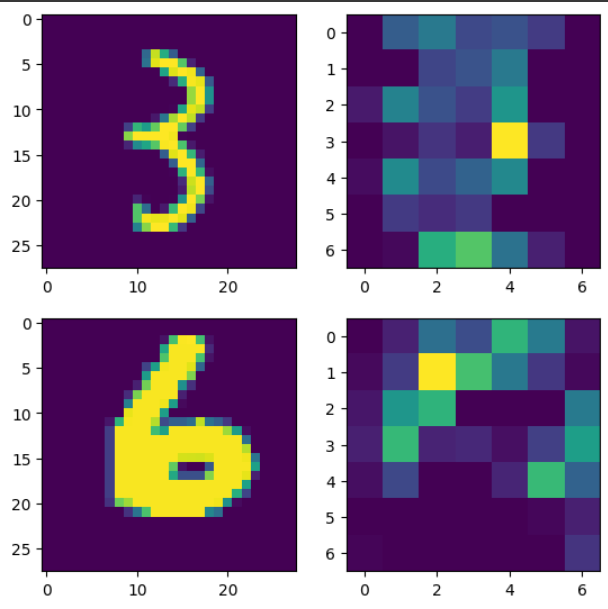


ورودیهای این تابع، عکس ورودی، مدل آموزش دیده و اسم آخرین لایه کانولوشنی در آن مدل است.

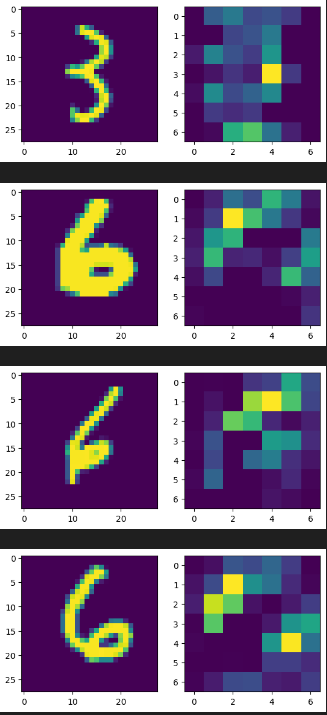
ورودیهای بالا رو توی کد زیر به تابع دادم:

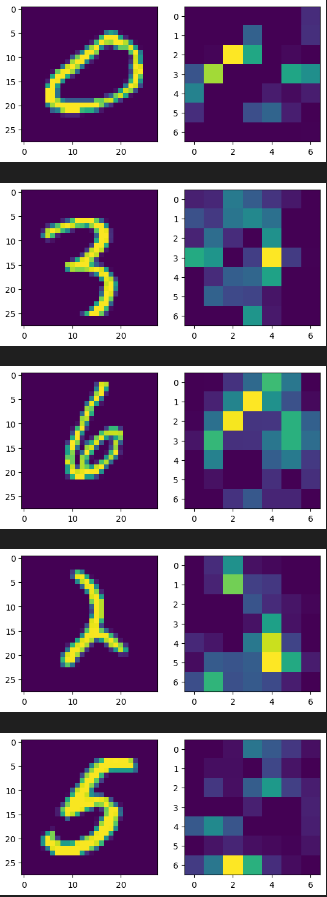


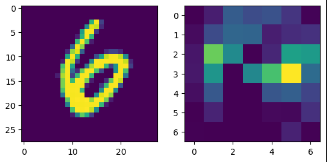
خروجی برای عکس اول و دوم:



ج)







نکاتی که درون این heat map ها میتوان به آن توجه کرد:

* + خانه هایی از نقشه که برای یک رقم، مقادیر روشنی دارند، نشان دهنده ویژگی هایی هستند که مخصوص آن رقم است. برای مثال برای رقم 6 در عکس دوم و چهارم پیکسل خانه (2,1) مقدار بزرگی دارد. یا برای رقم 3 در دو عکسی که حاوی این رقم هستند، خانه (4,3) حاوی بیشترین مقدار هستند.
  + نکته دیگر این است که ممکن است روشن بودن ترکیب چند خانه یا خاموش بودن ترکیب چند خانه، نشان دهنده یک رقم باشد، به همین دلیل بعضی خانه هایی که برای یک رقم 6 روشن بودند، لزوما برای عکسی دیگر حاوی این رقم روشن نخواهند بود.
  + نقشه حرارتی روی ارقام یکسان با فونت های مختلف نتایج متفاوتی دارد ولی نتیجه گیری ای که کلیت نقشه در لایه های خطی بعدی میشود، همان رقم را با احتمال بالا پیش بینی میکند.