بسم الله الرحمن الرحیم

محمد عرفان زارع زردینی

۹۸۴۱۱۴۳۲

تمرین سری سوم درس یادگیری عمیق

مدرس درس: استاد داوود آبادی

پاییز 1402

سوال 1)

منابع:

Chattgpt(prompt:’ Introduce types of Tuner in KerasTuner.’)

https://www.tensorflow.org/tutorials/keras/keras_tuner

https://keras.io/keras_tuner/

https://www.javatpoint.com/tensorflow-mnist-dataset-in-cnn

chatgpt(prompt: how to optimize convolution network with Keras Tuner for classifcation task.also talk about the importance and effect of using dropout and pooling in improving network performance)

الف)

ابزاری است برای بهینه سازی و تنظیم دقیق معماری های شبکه عصبی و ابرپارامترها. روشی انعطاف پذیر و کارامد برای جست وجوی بهترین ابرپارامترها برای مدل شبکه عصبی همچون نرخ آموزش ، تعداد لایه ها و اندازه لایه ها می باشد.

مراحل بهینه سازی شبکه هم گشتی با استفاده از این ابزار و کتابخانه به شرح زیر است:

ابتدا کتابخانه های keras ,keras tuner را اضافه نموده و داده های طبقه بندی شده را پیش پردازش(در صورت نیاز) و آماده می سازیم و به مجموعه داده آموزش و اعتبارسنجی قسمت می کنیم. سپس مدل هم گشتی مدنظر را با پارامتر های قابل تنظیم ایجاد نماید. برخی ابرپارامترها قابل تنطیم با ابزار مورد استفاده تنظیم می شوند. سپس نوع تیونری که میخواهیم استفاده نماییم را تعیین می نماییم که به طور مثال می تواند RandomSearch، Hyperband، BayesianOptimization و ابرپارامتر هارا برای جستوجو و فضاهای جستجوی مربوطه آنها را تعریف می کنیم. تیونر را براساس تابع ساخت مدل، فضای جست وجو ابرپارامتر، تعداد آزمایش ها و سایر پارامترها اجرا می کنیم. تیونر در فضای ابرپارامترها جست وجو می کند و مدل های مختلف را با ترکیب های مختلف ابرپارامترها آموزش می دهد و پس از جست وجو، بهترین مدل را براساس معیارهای تعریفی از نتایج تیونر، بازیابی می کند. من از تیونر Hyperband به دلیل توانایی در جست وجوی کارآمد پیکربندی هایپرپارامتر، به ویژه با تمرکز بر به حداقل رساندن خطا در طبقه بندی مجموعه داده MNIST استفاده کردم. تخصیص

منابع تطبیقی و مکانیسم توقف زودهنگام Hyperband، آن را برای جست وجو در طیف وسیعی از ابرپارامترها، با هدف یافتن بهترین پیکربندی ها در تکرارهای کمتر، مناسب می کند.

همانطور که گفتیم اتیونر ها در این ابزار انواع مختلفی دارند که در پایین توضیح داده شده است.

1. RandomSearch: این تیونر به طور تصادفی ترکیب های ابرپارامتر را در فضای جستجوی تعریف شده انتخاب می کند. برای جستجوی اولیه در محدوده وسیعی از ابرپارامترها کاربردی است.

2. Hyperband: بر اساس مفهوم نصف کردن متوالی، Hyperband تعداد زیادی مدل را برای چند دوره آموزش می دهد و مدل هایی با عملکرد بهتر را برای آموزش بیشتر انتخاب می کند. با متوقف کردن زودهنگام مدل‌های کمتر سودمند، منابع را به طور مفیدتری تخصیص می‌دهد.

3. بهینه سازی Bayesian: با استفاده از استنتاج بیزین، این تیونر یک مدل احتمالی از تابع هدف می سازد (به عنوان مثال، دقت اعتبارسنجی) و ابرپارامترهایی را انتخاب می کند که احتمالاً عملکرد مدل را بهبود می بخشد.

4. Sklearn: تیونر Keras را با استفاده از scikit-learn های RandomizedSearchCV وGridSearchCV ادغام می کند، که امکان تنظیم در یک پایپ لاین scikit-learn را فراهم می نماید.

انتخاب براساس معیارهای متفاوتی می تواند باشد که شامل اندازه فضای جست و جو(اگر فضای جست وجو بزرگ و متنوع باشد، بهینه سازی کارآمدتر است.) ، محدودیت منابع(hyperband ممکن است زمانی ارجح باشد که منابع محاسباتی محدود باشد، چون می تواند همانطور که گفتم منابع را مدیریت کند.) ، exploration & exploitation (بهینه سازی بیزی میان اکتشاف و آزمایش پیکربندی های جدید و بهره برداری و تمرکز بر بخش های امیدوارکننده تر ، تعادل برقرار نموده و این درحالی است که جست وجوی تصادفی فضارا به طور یکنواخت انجام می دهد.

ب)

مجموعه داده MNIST یک مجموعه داده مستند کلاسیک در یادگیری ماشین و بینایی کامپیوتر است. شامل مجموعه ای از تصاویر 28*28 خاکستری از ارقام دست نویس (0-9) است. MNIST شامل 60000 تصویر آموزشی و 10000 تصویر تست است که برای آموزش و ارزیابی مدل ها، به ویژه برای کارهای طبقه بندی رقمی کاربرد دارد.

برای استفاده از ابزارمان روی این دیتاست ابتدا مجموعه داده موردنظر را با استفاده از کتابخانه keras یا tensorflow لود نموده ونرمال سازی می نماییم. (در محدوده 0تا1) سپس تابعی درجهت پیاده سازی مدل مان با ابرپارامترهای قابل تنظیم می نویسیم(می تواند شامل تعداد لایه های کانولوشن، اندازه فیلتر، اندازه هسته، نرخ یادگیری، نرخ ترک تحصیل و غیره باشد.). حال تیونر مناسب را پیکربندی کرده و ابرپارامترهای مناسب برای آن را تعریف می کنیم. حال تیونر را با مشخص کردن تابع ساخت

مدل، فضای جستجوی ابرپارامتر و سایر پارامترهای مرتبط اجرا می نماییم. تیونر ترکیب های مختلف ابرپارامترها را بررسی می کند، چندین مدل را آموزش می دهد و عملکرد آنها را ارزیابی می کند و درنهایت بهترین را بر می گزیند. در واقع در بهینه‌سازی یک شبکه برای مجموعه داده‌های MNIST با استفاده از Keras Tuner، آزمایش با ترکیب‌های مختلف dropout rates، استراتژی‌های pooling و سایر ابرپارامترها می‌تواند به طور قابل‌توجهی بر عملکرد و قابلیت‌های تعمیم مدل تأثیر بگذارد. تنظیم این مؤلفه ها امکان تنظیم دقیق معماری شبکه را فراهم می کند تا با پیچیدگی های مجموعه داده مطابقت داشته باشد.

اهمیت و تاثیر استفاده از دو مورد ذکر شده در بهبود عملکرد شبکه را در زیر توضیح دادم:

dropout تکنیک منظم سازی برای جلوگیری از بیش برازش بیش از حد شبکه است و به طور تصادفی مقدار مشخصی از نورون ها را در آموزش حذف می کند. با این حذف، شبکه نسبت به وزن های خاص هر نورون حساستر شده و قابلیت تعمیم می یابد و سبب می شود تا ویژگی های قویتری را بیاموزد. همچنین از تکیه بیش از حد شبکه به نورن های خاص جلوگیری می کند و تعمیم به داده های دیده نشده را بهبود می دهد. در واقع با کاهش بیش از حد بیش برازش و بهبود توانایی آن برای تعمیم ، به داده های دیده نشده بر بهبود عملکرد شبکه کمک می کند.

در pooling برای نمونه برداری از ابعاد در فضای ورودی، کاهش پیچیدگی محاسباتی وتعداد پارامترهای شبکه کاربرد دارد. این روش به نگه داری مهمترین ویژگیها کمک نموده و در عین حال اطلاعات کمتر مرتبط را دور می ریزد. همچنین حساسیت به نویز در دادهای ورودی را کم می نماید که سبب بهبود کارایی و توانایی شبکه برای استخراج ویژگی های ضروری از داده می شود. و درنهایت توانایی شبکه را بر تمرکز روی مهمترین ویژگی ها و تعمیم بهتر به تغییرات تصاویر ورودی به همراه دارد.

ج)

ابتدا 6 پارامتر داده شده را مانند زیر در کد تنظیم نموده، و با استفاده از hyperband، مدل را بررسی می نماییم. باتوجه به زمان زیادی که نیاز دارد(به علت زیاد بودن حالات) آن را بر روی تعداد دفعات محدودتری پیش برده و نتایج را بررسی نمودیم.

```
def build_model(hp):
    model = keras.Sequential()
    for i in range(hp.Int('conv_layers', 2, 5)):
        model.add(keras.layers.Conv2D(
            filters=hp.Int(f'filters_{i}', 32, 256),
            kernel_size=hp.Choice(f'kernel_{i}', [3, 5]),
            activation='relu'
        ))
        model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))

    model.add(keras.layers.Flatten())
    for i in range(hp.Int('dense_layers', 2, 5)):
        model.add(keras.layers.Dense(
            units=hp.Int(f'units_{i}', 32, 256),
            activation='relu'
        ))
        model.add(keras.layers.Dropout(rate=hp.Float(f'dropout_{i}', 0.0, 0.5, step=0.1)))

    model.add(keras.layers.Dense(10, activation='softmax'))

    model.compile(optimizer=keras.optimizers.Adam(learning_rate=hp.Choice('learning_rate', [1e-4, 5e-4, 1e-3])),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

```
tuner = Hyperband(
    build_model,
    objective='val_accuracy',
    max_epochs=7,
    factor=3,
    directory='my_dir',
    project_name='mnist_classification',
    max_consecutive_failed_trials=5
)
```

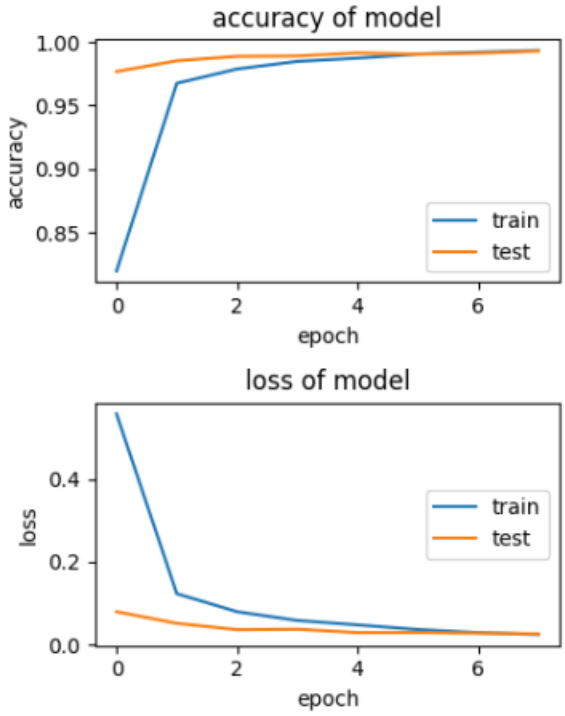
```
Trial 8 Complete [00h 00m 01s]

Best val_accuracy So Far: 0.9886666536331177
Total elapsed time: 01h 23m 33s
```

```
{'conv_layers': 3,
 'filters_0': 138,
 'kernel_0': 5,
 'filters_1': 138,
 'kernel_1': 5,
 'dense_layers': 4,
 'units_0': 188,
 'dropout_0': 0.2,
 'units_1': 151,
 'dropout_1': 0.0,
 'learning_rate': 0.0001,
 'filters_2': 167,
 'kernel_2': 3,
 'filters_3': 62,
 'kernel_3': 5,
 'filters_4': 42,
 'kernel_4': 5,
 'units_2': 142,
 'dropout_2': 0.4,
 'units_3': 139,
 'dropout_3': 0.4,
 'units_4': 101,
 'dropout_4': 0.4,
 'tuner/epochs': 8,
 'tuner/initial_epoch': 0,
 'tuner/bracket': 0,
 'tuner/round': 0}
```

ما انتظار داریم تا تعداد لایه هم گشتی بیشتر از لایه های کاملا متصل باشد. همچنین تعداد لایه های هم گشتی و تعداد فیلترها به گونه ای نباشند که سبب پیچیدگی مدل و کاهش تعمیم پذیری آن شوند. همه موارد پوشش داده می شوند به جز آن که تعداد لایه های کاملا متصل و تعداد نورون های آنها بیشتر از لایه هم گشتی است که درصورت بیشتر بودن تعداد و ادامه یافتن جست وجو بدان می رسیدیم.(ممکن است مقادیر تعداد لایه هم گشتی وفیلترها بهترین مقدار ممکن نباشد و در ادامه تغییر می کرد.)

من برای بررسی و نمایش دقت و خطا ، بهترین هایپرپارامترها را نگه داشته و مدل را اموزش دادم .

نتایج مدل نیز در نمودار به شکل زیر می شود:



اطلاعات دقیق دقت و خطا در فایل موجود است. همچنین همانطور که مشاهده می شود مدل به وخوبی آموزش داده شده است. و دقت روی داده های تست بالا است.

انتخاب اندازه فیلترها در لایه های کانولوشن:

اندازه فیلترها در لایه های کانولوشنال به طور قابل توجهی بر توانایی شبکه برای استخراج ویژگی ها تأثیر می گذارد. اندازه های رایج فیلتر شامل3x3 و 5x5 است. انتخاب بستگی به این دارد:

پیچیدگی و جزئیات در داده ها: فیلترهای کوچکتر (به عنوان مثال، 3x3) جزئیات دقیق تری را ثبت می کنند، مناسب برای ویژگی های پیچیده.(وجه تمایز در واقع در ویژگی های کوچک و محلی است)

کارایی محاسباتی: فیلترهای کوچکتر پارامترها و بار محاسباتی را کاهش می دهند.

سلسله مراتب ویژگی ها: فیلترهای بزرگتر (به عنوان مثال، 5x5) ممکن است ویژگی های سلسله مراتبی پیچیده تری را ثبت کنند.به طور کلی، شروع با فیلترهای کوچکتر و افزایش تدریجی اندازه با پیشرفت شبکه ممکن است هر دو ویژگی ریز و درشت را به طور موثری ثبت کند.

کاهش ابعاد: اندازه کرنل 1*1 برای کاهش ابعاد استفاده و هدف آن کاهش تعداد کانال ها است. بدین گونه که مشارکت های کانال های ورودی را تنها در یک پیکسل از نقشه ویژگی ثبت می کند.

◦ field Receptive: اگر تمایل داریم که مدل مان میدان دید بیشتری داشته باشد، اندازه فیلترها را زیاد می کنیم.

در این دیتاست به علت نسبتا ساده بودن، فیلتر با سایز کوچک استفاده کردم.

برای قسمت بعد هم ,همانطور که بخشی از موارد را در قسمت قبل گفتم، در اینجا به ادامه موارد اشاره می نمایم:
لایه ادغام نقشه های ویژگی را نمونه برداری کرده وو ابعاد را کم می کند. این فرایند دارای چندین اثر همچون کاهش پیچیدگی مدل، تعمیم ویژگی ها، کاهش حساسیت به مکان دقیق ویژگیها (بدون توجه به موقعیت دقیق عناصر در ورودی ، ویژگی ها را تشخیص دهد و آموخته اش را قوی نماید.)

Dropout سبب می شود تا بازنمایی اضافی از ویژگی ها را بیاموزد، و آن را قویتر و کمتر به نورون های خاص وابسته می کند.همچنین انطباق شدن نورون ها با داده های آموزشی را کاهش می دهد وهمچنین تعمیم را بهبود می دهد. در cnn لی ن روش ها بعد از لایه کانولوشن استفاده می شود.
آزمایش دقیق با قرارگیری و پارامترهای لایه‌های ادغام و dropout، همراه با تنظیم مناسب سایر ابرپارامترها، می‌تواند به طور قابل توجهی توانایی CNN را برای تعمیم و بهبود دقت با کاهش گرایش‌های بیش‌برازش افزایش دهد. (منبع این بخش همان منابع قسمت قبل اند که بسط داده شده اند)

(2

فایل مد در زیپ موجود است.(تی ای مربوطه گفت نیاز به توضیح کد نیست.)

(3

از مشکلاتی که می تواند سبب منفی شدن امتیاز r2 شود، می توان به موارد زیر اشاره کرد: (که البته با توجه به توضیحات تی ای موارد اولیه نیست و در آخر به موردی که مدنظر هست اشاره کردم.

پیش پردازش داده ها : ممکن است داده ها به درستی پیش پردازش نشده باشند و ویژگی ها و مقادیر هدف استاندارد نباشند. همچنین ممکن است داده هایی داشته باشیم که متناقض باشند و بر مدل اثر منفی گذارند.

شکل ورودی: بعد سوم برای شکل ورودی باید تعداد ویژگی ها باشد نه 1.

لایه خروجی: استفاده از یک لایه Flatten قبل از لایه نهایی Dense ضروری نیست و به نظر می شود اصلاح شود.

موارد اصلاحی هم داخل توضیحات اشاره نمودم. باتوجه به اینکه گفته شده ساختار آن درست است به نظر تنها اصلاح این موارد وهمچنین مقیاس بندی داده هایمان با استفاده از MinMaxScaler ، و تقسیم فیچر ها و تغییر شکل ورودی ها می تواند اثر گذار باشد.

(4

Refrences: chatgpt{prompt: what is cnn,rnn and applications. Also compare them}{prompt: compare each of them about number of paraameters and parallelization}

شبکه هم گشتی (کانولوشن)(CNN):

شبکه ای تخصصی برای پردازش داده های شبکه مانند ساختاریافته طراحی شده و عمدتا برای تجزیه و تحلیل تصویر و تسک های تشخیص کاربرد دارد. بدان گونه است که با اعمال فیلتر ها برروی داده های ورودی و لایه pooling برای کم کردن ابعاد، ویژگی استخراج را انجام میدهد وهمچنین ازلایه های ورودی و لایه های کاملا متصل تشکیل شده است .بیشتر کاربرد در بینایی کامپیوتر هست.

کاربردها{

1- تشخیص تصویر: شناسایی اشیا، رخداد یا الگوهای درون تصویر

2- طبقه بندی اشیا درون تصویر

3- تولید تصویر: ایجاد تصویرجدید مانند انتقال سبک یا مدل های تولیدی

4-تصویربرداری پزشکی: تجزیه تحلیل تصاویر پزشکی ، تشخیص بیماری ها}

ویژگی ها {

1-یادگیری سلسله مراتبی: یادگیری سلسله مراتبی ویژگی ها، آموزش الگوهای محلی و ترکیب آنها به شکل تدریجی تا نمایش سطح بالاتری را تشکیل دهند. 2- تغییرناپذیری در برابر تغییرات در موقعیت یا جهت گیری ویژگی ها در یک تصویر 3- اشتراک گذاری پارامترها: وزن ها از طریق این لایه به اشتراک گذاشته شده و تعداد پارامترها کاهش می یابد.(بدین سان برای ورودی های بزرگ همچون تصویر مناسبه) }

شبکه عصبی بازگشتی(RNN):

برای کار با داده های متوالی با پردازش ورودی ها به صورت متوالی و حفظ حافظه داخلی طراحی شده است. از لوپ فیدبک برای پردازش اطلاعات در طول زمان استفاده می شود به آن اجازه می دهد توالی های با طول مختلف را مدیریت کنند. در این شبکه عصبی ، خروجی مرحله قبل به عنوان ورودی مرحله فعلی داده می شود.

کاربردها:{

1- پردازش زبان طبیعی: مانند مدلسازی زبانی، ترجمه ماشینی ، تجزیه وتحلیل احساسات و تشخیص گفتار
2- تحلیل سری زمانی: پیش بینی و تشخیص ناهنجاری و الگو در داده های متوالی
3- ایجاد توالی هایی همچون متن و موسیقی
4- درک و تجزیه و تحلیل داده های ویدئویی فریم به فریم}

ویژگی ها:{

1-مدل سازی وابستگی های متوالی و به شکل مفیدی زمینه و اطلاعات زمانی را ضبط کند.

2- دنباله های ورودی با طول های مختلف را کنترل می کند و آنها را برای داده های متوالی مناسب سازی می کند.

3-یادگیری وضعیتی : حفظ زمینه ها در طول مراحل زمانی (مناسب برای درک زبان)

تفاوت CNN,RNN :

1- توانایی پردازش اطلاعات زمانی
2- CNN ورودی با اندازه ثابت می گیرند و خروجی هایی با اندازه ثابت تولید می کند و برای کارهایی مورد استفاده است که ورودی و خروجی مستقل از هم اند. درحالی که RNN برای طول های ورودی و خروجی را مدیریت می کنند و برای کارهایی کاربرد دارند که خروجی به ورودی قبلی یا آینده وابسته است.
3- CNN برای پردازش تصویر و ویدئو استفاده می شود. درحالی که از RNN برای تجزیه و تحلیل متن و گفتار استفاده می شود.

ب)

تعداد پارامتر ها:

Cnn : تعداد پارامترهای آن بسته به اندازه فیلترها در لایه های کانولوشن و تعداد فیلترها تعیین می شود. لایه های fully connected نیز بر تعداد پارامترها تاثیر می گذارد. البته به دلیل اشتراک پارمتر ها و تغییرناپذیری فضایی در لایه کانولوشن، اغلب پارامترهای کمتری نسبت به شبکه کاملا متصل دارد.

RNN : تعداد پارامتر ها در آن بسته به تعداد واحد چنهان و اندازه ورودی است. هر اتصال بین واحد ها دارای یک وزن مرتبط است و هر واحد دارای یک بایاس است . پس همه پارامترها میتواند برای شبکه پیچیده ، بسیار زیاد باشد.

پیرامون قابلیت موازی سازی:

RNN : چون ذات توالی دارندد ، برای موازی سازی چالش دارند. همانطور هم که در قسمت قبل اشاره شد، خروجی یک مرحله به زمان مرحله قبل وابسته است. بدین سان توالی از وابستگی ایجاد می شود که عملیات موازززی سازی را محدود می کند. برای حل این موارد از روش هایی همچون انتشار پس انداز کوتاه استفاده می شود که در حل این مشکل و کارآمدی آموزش ،موثر است. به دلیل توانایی در داشتن پارامتر زیاد ، موازی نمودن آن سخت تر است.

CNN : CNN های به دلیل ماهیتشان، قابلیت موازی سازی دارند چون وزن های یکسانی در قسمت ورودی اعمال شده و سبب می شود که این عملیات را میتوان به صورت موازی انجام داد. همچنین به طور معمول پارامترهای کمتری دارد و موازی سازی آسان تری دارد.

سوال5)

Conv(filters=c , padding=same) -> input:(w,h,c) output: (w,h,c)

Dilated-Conv(filters=c , kernel=k, strid=s, dilation_rate=a, padding=valid) -> if input:(w,h,c)

output:

$$\left(\frac{w-[(k-1)(a-1)+k-1]}{s},\frac{h-[(k-1)(a-1)+k-1]}{s},c'\right)=(\frac{w-(k-1).a}{s},\frac{h-(k-1).a}{s},c')$$

Max-Pooling(window=(k, k), strid=s): if input:(w,h,c) output: → (w//s, h//s, c)

Layer	Output size	Params
Layer0: Input	(256, 256, 3)	0
Layer1: Conv(64, (3,3), 1, same)	(256, 256,64)	64*(3*3*3+1)
Layer2: Dilated-Conv(32, (5,5), 2, dr=2, valid)	(126, 126, 32)	32*(5*5*64+1)
Layer3: Max-pool (size=(2,2), stride=2)	(63, 63, 32)	0
Layer4: Conv(128, (3,3), stride=1, padding='same')	(63, 63, 128)	128*(3*3*32+1)
Layer5: Dilated-Conv(64, (5,5), stride=2, dilation rate=4, padding='valid')	(28, 28, 64)	64*(5*5*128+1)
Layer6: Max-pool (size=(2,2), stride=2)	(14, 14, 64)	0
Layer7: Conv(256, (3,3), stride=1, padding='same')	(14, 14, 256)	256*(3*3*64+1)
Layer8: Dilated-Conv(128, (5,5), stride=2, dilation rate=8, padding='valid')	((4, 4, 128)	128*(5*5*256+1)
Layer9: Max-pool (size=(2,2), stride=2)	(2, 2, 128)	0
all		1261920

ب)

با فرض اینکه ورودی به فرمت W*H باشد و سایز کرنل ما x*y باشد و عملیات conv با stride مقدار s انجام شود، داریم :

Padding(for weigh)=(W-1)*s(for weigh) +x-W

Padding(for height)=(H-1) * s(for height) +y-H

همانطور که میدانیم پدینگ بدین صورت است که نصف آن برای یک سمت(راست، بالا) و نصف دیگر برای سمت دیگر(چپ، پایین) می باشد. در فرضی ساده اگر پدینگ نباشد، به اضافه سایز کرنل m، به اندازه m//2 از (راست، بالا) و m//2 از (چپ، پایین) ایگنور می شود. پس نیاز پدینگ با سایز m-1 داریم. می توان آنها را به فرمت) 2/(m-1) هم نوشت.(با فرض stride برابر1) (اگر m فرد باشه)

(6

Refrences: chatgpt{prompt: what is epsilon in batch normalization}{ What is the problem or problems of using batch normalization with size one}

الف)

جمله دوم درست است.(نرمال سازي دسته اي توزيع خروجي را نرمال مي كند تا در ابعاد يكنواخت تر باشد.){هدف نرمال سازی همین است. یکنواخت نمودن ورودی با توزیع مناسب}

نرمال‌سازی دسته‌ای تکنیکی است که در شبکه‌های عصبی برای عادی‌سازی ورودی‌های هر لایه، کمک به کاهش تغییر متغیر داخلی و تسریع فرآیند آموزش استفاده می‌شود. این تضمین می‌کند که خروجی‌های یک لایه در طول آموزش با عادی‌سازی توزیع خروجی نرمال می‌مانند، که به تثبیت و سرعت بخشیدن به آموزش شبکه‌های عصبی عمیق کمک می‌کند.

دلیل غلط بودن عبارت اول: در این روش هنگام forward pass یک سری عملیات بیشتر انجام می شود و با افزودن این لایه به شبکه ، تعداد پارامترهای مدل زیاد می شود. بدین سان هنگام backpropation پارامترهای بیشتری آپدیت می شوند. پس زمان آموزش کم نمی شود و فقط زمان همگرا شدن مدل را بهبود می دهد.

دلیل غلط بودن عبارت سوم: این روش، اثر مقداردهی اولیه وزن ها را کم نموده ولی آنرا به سمت صفر نمیبرد لزوما.

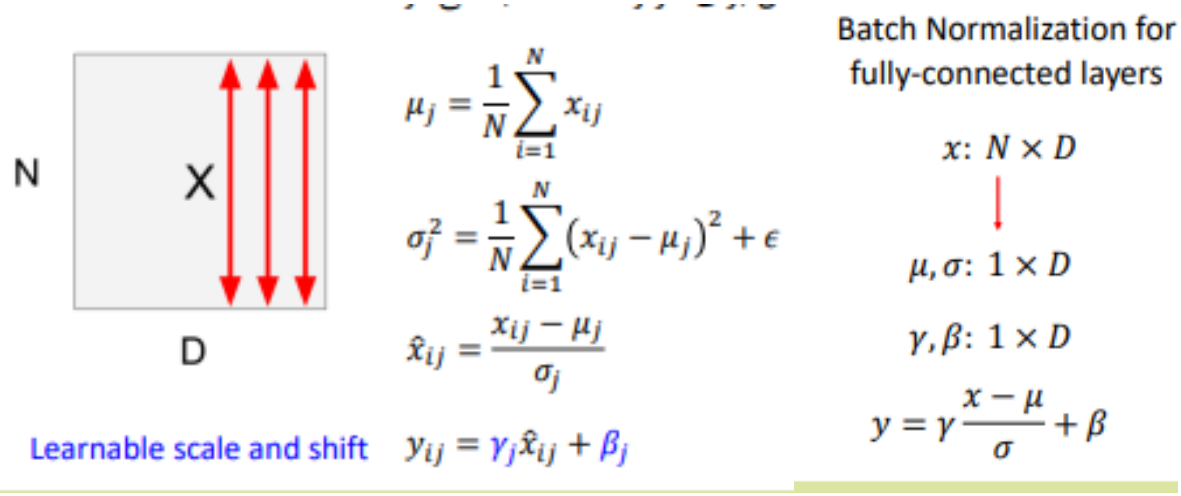
ب)

با توجه به اسلایدهای درس داریم:

در train: در هر کانال ، میانگین و واریانس را یافته و با moving avg,moving var مقدار میانگین و واریانس را اپدیت می کنیم.

در test : مقدار میانگین و واریانس را برابر آخرین مقدار آپدیت شده می گذاریم.

حال با استفاده از فرمول نرمال سازی ، مقدار ورودی رانرمال کرده و پارامتر های gamma ,beta را اعمال می کنیم.



ج)

نقش اساسی در لایه normalization batch برای حفظ ثبات عددی در طول فرآیند نرمال سازی داره.

در نرمال سازی دسته ای ، با کم کردن mean دسته ای و تقسیم بر رادیکال واریانس(انحراف معیار)، لایه یک لایه را برای هردسته نرمال انجام می دهیم. ولی در صورت کوچک بودن انحراف معیار، خروجی تقسیم عددی بزرگ است و سبب بی ثباتی عددی شده و آموختن اثر منفی دارد. پس درجهت جلوگیری از آن ، قبل از رادیکال گیری، مقدار اپسیلون به واریانس اضافه می شود تا سبب می شود تا تقسیم بر عدد صفر یا کوچک رخ ندهد .

د)

مشکلات این روش:

1- بیش برازش در مدل های کوچک: استفاده از آن برای مدل های کوچک، می تواند سبب بیش برازش شود. زیررا نرمال سازی توانایی مدل را در تطبیق با داده های آموزشی کم می کند.
2- وابسته بودن به اندازه دسته: عملکرد نرمال سازی دسته ای به اندازه دسته آموزشی وابسته باشد. در برخی حالات، اندازه دسته ممکن است تاثیر مستقیمی بر عملکرد و کارایی این نرمال سازی داشته باشد.
3- تاثیر بر ورودی های کوچک: وقتی دسته‌های آموزشی کوچک هستند، میانگین و واریانس محاسبه شده ممکن است ناپایدار شود و بهبود عملکرد نرمال‌سازی دسته‌ای را کاهش دهد.
4- پیچیدگی افزوده شده: استفاده از نرمال‌سازی دسته‌ای ممکن است به معماری شبکه افزوده شده و محاسبات را پیچیده‌تر کند.

5- محدودیت استفاده در برخی معماری ها: در برخی از معماری‌های شبکه عصبی مانند برخی از مدل‌های بازگشتی، استفاده از نرمال‌سازی دسته‌ای ممکن است به‌طور کامل کارایی خود را از دست بدهد یا پیچیدگی بیشتری را به دست آورد.

ه)

10 input -> fc-20 -> BatchNorm

Layer	output	Params
Input	N*10	0
FC-20	N*20	20*(10+1)
BatchNorm	N*20	20*2

7)

این روش تکنیک برای درک شبکه عصبی کانولوشنال است، با برجسته کردن منطقه تصویری که مدل هنگام انجام پیش بینی روی آنها تمرکز می نماید، استفاده می شود. به درک اینکه کدام بخش، بیشترین کمک را به پیش بینی نهایی مدل می نماید، می کند.با روش گفته شده در مسئله(grad-cam) تصویر از پیش پردازش را از مدل عبور می دهیم تا نقشه ویژگی کانولوشن نهایی را بیابیم. سپس روی آن backpropagation میزند و با ترکیب نمودن گرادیان ها و دادن وزن بدان ، اهیمت آن نقشه ویژگی را می یابیم.سپس نقشه ویژگی ها را به روی تصویر اصلی قرار داده تا مناطقی که بیتشرین ارتباط را برای پیش بینی دارند تجسم کند . من در این جا تعدادی از آن هارا قراردادم.(کامل آن ها داخل کد هست)

پارامترها و روش‌های مورد استفاده برای Grad-CAM اغلب شامل انتخاب کلاس هدفی است که می‌خواهید توجه را برای آن تجسم کنید، انتخاب آخرین لایه کانولوشن قبل از لایه ادغام میانگین جهانی، و تنظیم وزن‌ها بر اساس آن.

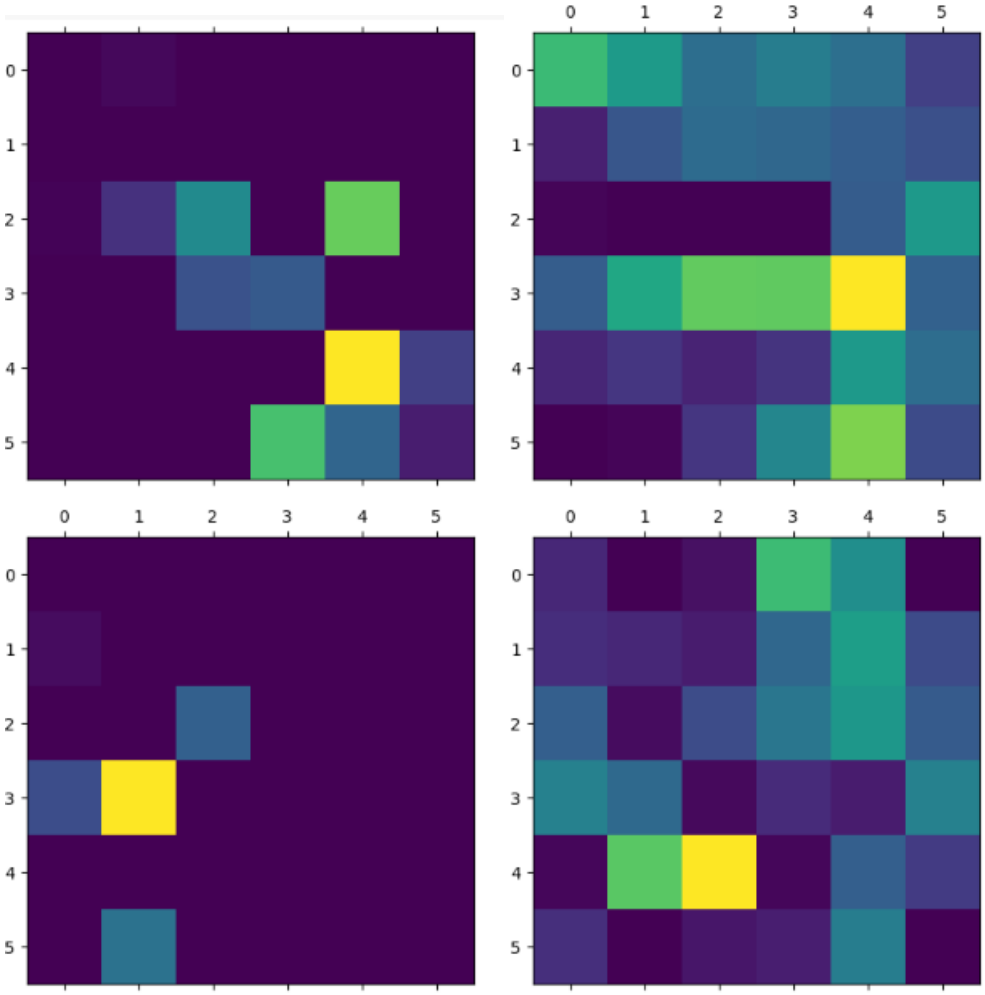
کد نیز براساس مراحل توضیح داده شده در سوال می باشد.

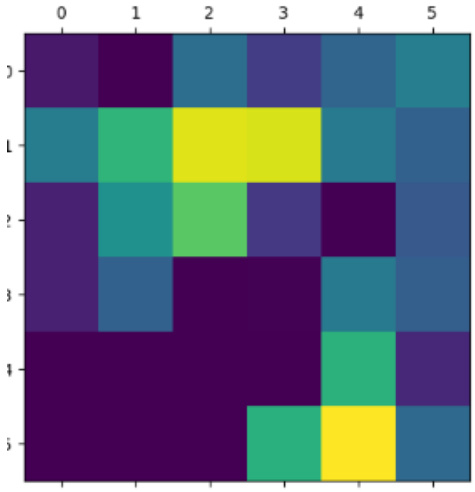
راجب کد مربوط به grad_cam ، ابتدا مدلی طراحی نموده که ورودی را به فعالسازها دهد. در لایه آخر کانولوشن، خروجی پیش بینی می باشد.سپس گرادیان کلاس بالاتر پیش بینی شده برای ورودی تصویر را با توجه به فعالساز لایه کانولوشن آخر محاسبه می کنیم. همچنین گرادیان نورن خروجی را با توجه فیچر مپ خروجی لایه آخر محاسبه میکنیم.همچنین برخی کار های دیگر همچون ضرب کانال در آرایه فیچر مپ را انجام میدهیم. و درنهایت برای visualize نرمال سازی انجام میدهیم میان 0و1. همچنین لایه سافت مکس اخر را حذف نمودم. و درنهایت هیت مپ کلاس فعالساز را هم رسم می کنیم.

```
Training data dimensions: (60000, 28, 28)
```


```
Epoch 1/15
938/938 [==============================] - 72s 75ms/step - loss: 0.1645 - accuracy: 0.9496
Epoch 2/15
938/938 [==============================] - 69s 73ms/step - loss: 0.0472 - accuracy: 0.9854
Epoch 3/15
938/938 [==============================] - 70s 75ms/step - loss: 0.0321 - accuracy: 0.9896
Epoch 4/15
938/938 [==============================] - 68s 73ms/step - loss: 0.0252 - accuracy: 0.9917
Epoch 5/15
938/938 [==============================] - 68s 72ms/step - loss: 0.0203 - accuracy: 0.9936
Epoch 6/15
938/938 [==============================] - 68s 73ms/step - loss: 0.0156 - accuracy: 0.9951
Epoch 7/15
938/938 [==============================] - 68s 73ms/step - loss: 0.0126 - accuracy: 0.9959
Epoch 8/15
938/938 [==============================] - 68s 73ms/step - loss: 0.0106 - accuracy: 0.9967
Epoch 9/15
938/938 [==============================] - 68s 73ms/step - loss: 0.0100 - accuracy: 0.9967
Epoch 10/15
938/938 [==============================] - 70s 75ms/step - loss: 0.0095 - accuracy: 0.9969
Epoch 11/15
938/938 [==============================] - 65s 70ms/step - loss: 0.0072 - accuracy: 0.9976
Epoch 12/15
938/938 [==============================] - 65s 70ms/step - loss: 0.0072 - accuracy: 0.9974
Epoch 13/15
938/938 [==============================] - 65s 69ms/step - loss: 0.0055 - accuracy: 0.9980
Epoch 14/15
938/938 [==============================] - 66s 70ms/step - loss: 0.0062 - accuracy: 0.9979
Epoch 15/15
938/938 [==============================] - 66s 70ms/step - loss: 0.0041 - accuracy: 0.9987
<keras.src.callbacks.History at 0x7b0479686b60>
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_6 (MaxPoolin  (None, 13, 13, 32)        0
 g2D)

 conv2d_7 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_7 (MaxPoolin  (None, 6, 6, 64)          0
 g2D)

 conv2d_8 (Conv2D)           (None, 6, 6, 64)          36928

 max_pooling2d_8 (MaxPoolin  (None, 3, 3, 64)          0
 g2D)

 flatten_2 (Flatten)         (None, 576)               0

 dense_4 (Dense)             (None, 128)               73856

 dense_5 (Dense)             (None, 10)                1290

=================================================================
Total params: 130890 (511.29 KB)
Trainable params: 130890 (511.29 KB)
Non-trainable params: 0 (0.00 Byte)
```




0
1
2
3
4
5