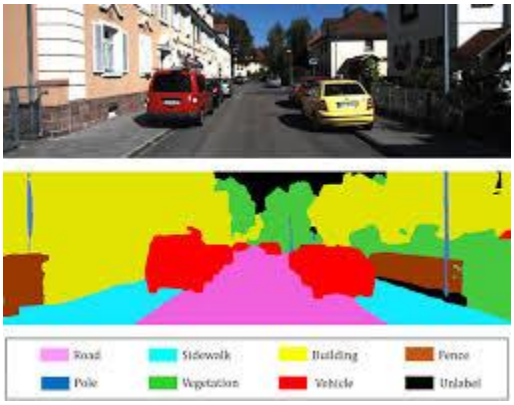
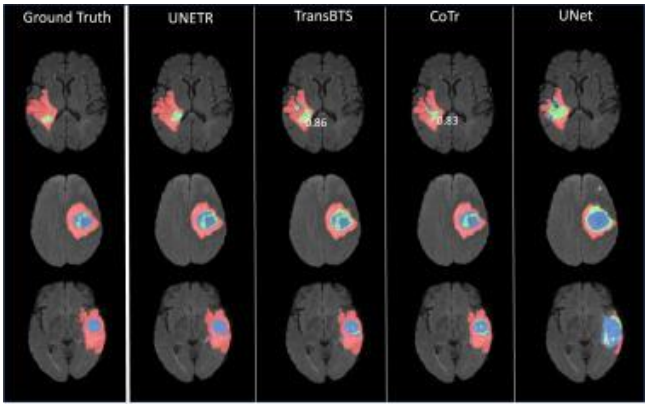
تعریف مسئله

پردازش تصویر و بینایی کامپیوتر یکی از شاخه‌های هوش‌مصنوعی است که به درک تصاویر توسط کامپیوتر با استفاده از الگوریتم‌های یادگیری عمیق می‌پردازد. یکی از موارد مورد استفاده‌ی بینایی کامپیوتر، ناحیه‌بندی معنایی (sematic segmentation) تصاویر است. در این تسک ما به دنبال این هستیم که تمام پیکسل‌های موجود درون تصویر ورودی را میان کلاس‌های موجود دسته‌بندی (classification) کنیم. در این حالت می‌توان مرز دقیق هر شی درون تصویر را تشخیص داد.

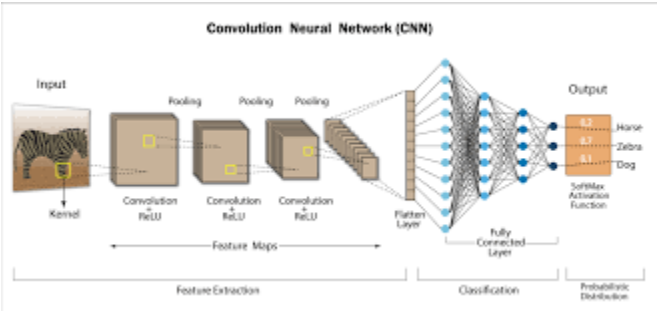


یکی از کاربردهای این تسک در پزشکی است که با تشخیص مرز دقیق تومورها در تصاویر پزشکی بافت‌های مختلف بدن، به دانشمندان و محققان حوزه‌ی پزشکی در راستای تصمیم‌گیری‌های بهتر کمک شایانی می‌کند.

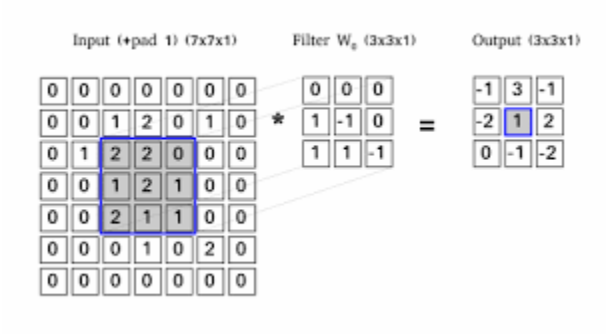


عمده‌ی کارهای انجام شده در زمینه‌ی بینایی کامپیوتر، با استفاده از شبکه‌های عصبی کانولوشنی (convolutional neural networks) انجام می‌شوند. این شبکه‌ها با اعمال فیلترهای مختلف بر روی تصویر ورودی، ویژگی‌های مختلفی را از آن استخراج کرده و سعی می‌کنند ماهیت آن را یاد بگیرند. شبکه‌های عصبی

با اعمال فیلترهای مختلف در هر لایه، به ازای هر فیلتر ویژگی متفاوتی را یاد می‌گیرند. سپس خروجی لایه‌ی جاری را به لایه‌ی بعدی داده و فیلترهای دیگری را برروی آن‌ها اعمال می‌کنند تا با ترکیب ویژگی‌های لایه‌های قبلی، ویژگی‌های سطح بالاتری را یاد بگیرند.



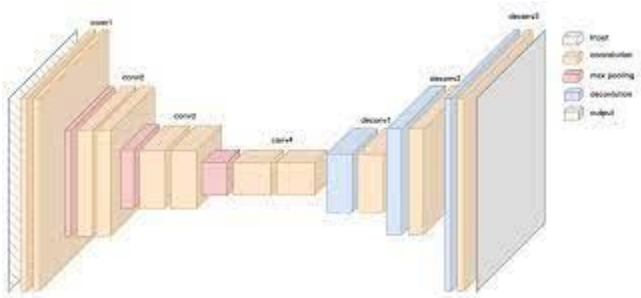
فیلترهای کانولوشنی، ماتریس‌های دو یا سه‌بعدی هستند که در هر اعمال، بخشی از تصویر ورودی را پوشش داده و بر روی آن بخش اعمال می‌شوند و خروجی می‌دهند. سپس با گامی مشخص به جلو لغزیده و بخش دیگری را پوشش می‌دهند. این عمل باعث می‌شود تا ابعاد فضایی (طول و ارتفاع) تصویر ورودی در هر لایه، بسته به گام و ناحیه‌ی تحت پوشش فیلتر دچار تغییراتی شود. برای اینکه بتوان عملیات استخراج ویژگی از تصویر را به‌صورت بهینه انجام داد، نیاز است تا ابعاد فضایی تصویر کاهش یابد اما برای استخراج ویژگی‌های غنی‌تر از تصویر، تعداد فیلترها زیاد می‌شوند.



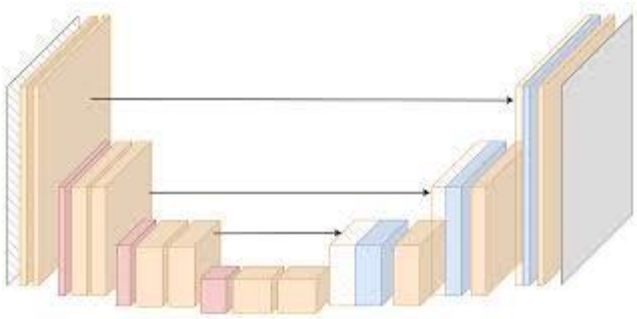
برای تسک ناحیه‌بندی معنایی، یادگیری ویژگی‌های سطح بالا برای اینکه بتوان تشخیص داد یک پیکسل متعلق بهخ چه کلاسی است نکته‌ای کلیدی است که کاربرد شبکه‌های عصبی کانولوشنی را بیش از پیش می کند. اما کاهش ابعاد فضایی (spatial resolution) یکی از مواردی است که در نتیجه‌ی استفاده از این شبکه‌ها، برای این تسک مورد دلخواه نیست. برای اینکه بتوان از ابعاد فضایی کاهش یافته، در اثر اعمال فیلترهای کانولوشنی به فضای پیکسلی اولیه رسید، بعد از استخراج ویژگی، وکتور نهایی به ابعاد اولیه بازگردانده می‌شود. سپس بر روی هر پیکسل یک دسته‌بند اعمال می‌شود تا بتوان کلاس هر کدام را تشخیص داد. محققین دریافتند که انجام این

کار به صورت پله‌ای می‌تواند تاثیرات بهتری بر روی نتیجه‌ی تسک ناحیه‌بندی معنایی داشته باشد. به این منظور که ابتدا تصویر اولیه وارد شبکه می‌شود. سپس بر روی آن فیلترهایی اعمال می‌شوند که ابعاد آن را کاهش نمی‌دهد. بعد از استخراج ویژگی‌های اولیه، با اعمال فیلترهای مناسب، ابعاد فضایی تصویر نصف می‌شوند. دوباره در این ابعاد، ویژگی‌هایی با سطح متوسط استخراج می‌شوند. سپس ابعاد آن دوباره نصف می‌شوند. همین روند ادامه می‌یابد تا به بعد خاصی برسیم (به این عمل اصطلاحا encoding گفته می‌شود). سپس از این بُعد ابتدا ابعاد تصویر را دو برابر می‌کنیم (به جای آنکه به یک باره به فضای پیکسلی اولیه برسیم). برای افزایش ابعاد هم می‌توان از فیلترهای خاص استفاده کرد و یا از عملیات‌های ریاضیاتی مانند interpolation بهره برد. سپس در این مرحله فیلترهایی که بعد را تغییر نمی‌دهند اعمال می‌شوند تا ویژگی‌های بهتری از بردار ورودی یاد بگیرند و سپس دوباره ابعاد آن دوبرابر می‌شوند. و این روند تا جایی ادامه می‌یابد که به ابعاد اولیه برسیم (به این عمل اصطلاحا decoding گفته می‌شود).

حال می‌توانیم بر روی این ویژگی‌های استخراج شده از تصویر که هم‌اندازه‌ی تصویر ورودی هستند، دسته‌بند خود را اعمال کنیم.



یکی از معماری‌های معروف در این حوزه، معماری Unet می‌باشد. توسعه‌دهندگان این شبکه دریافتند که به دلیل اینکه فیلترهای کانولوشن نگاه محلی به تصویر دارند (تنها بخشی از تصویر ورودی را پوشش می‌دهند) نمی‌توانند کلیت تصویر را به‌خوبی درک کنند و در عملیات decoding نمی‌توانند به‌خوبی ماهیت جسم درون تصویر را درک کنند زیرا بسیاری از ویژگی‌های مرحله‌ی encoding را فراموش کرده‌اند. بنابراین با یک راه ارتباطی (اصطلاحا skip connection) ویژگی‌های هر مرحله قبل از نصف شدن ابعاد را به سمت decoder بردند و با ویژگی‌های هم‌اندازه‌ی موجود در این مرحله، concat کردند که جهش بزرگی در زمینه‌ی ناحیه‌بندی معنایی بود.



برای مطالعه‌ی بیشتر در خصوص شبکه‌های عصبی کانولوشنی، تسک ناحیه‌بندی معنایی و شبکه‌ی Unet می توانید به منابع زیر مراجعه کنید:

Deep learning course

Digital image processing

Fundamentals of computer vision

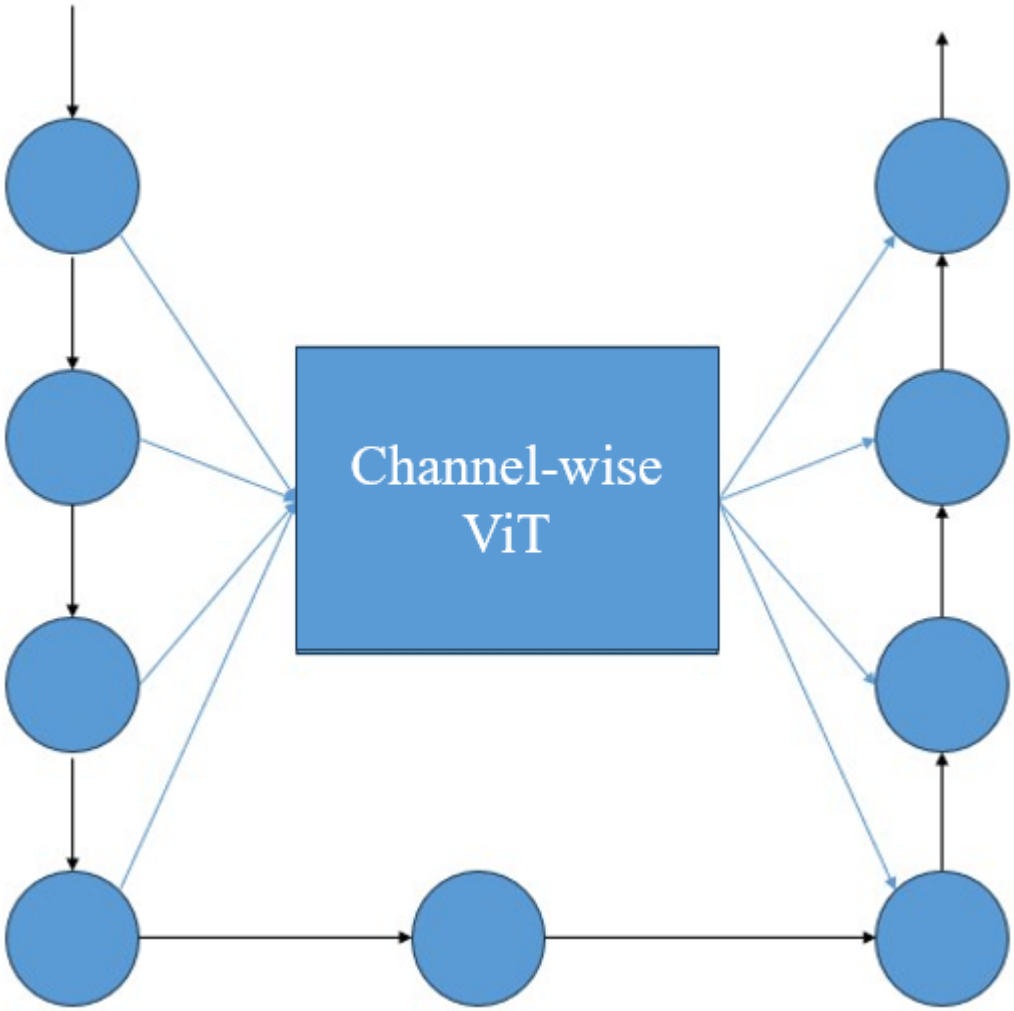
Convolutional Neural Networks

Unet architecture explained

اما مدل Unet با چندین مشکل مواجه است

1. Skip connection های ساده، اطلاعات مفیدی را حمل نمی‌کنند
2. میان ویژگی‌های Encoder و Decoder اختلاف معنایی وجود دارد
3. میان Stageهای مختلف اطلاعات به اشتراک گذاشته نمی‌شوند

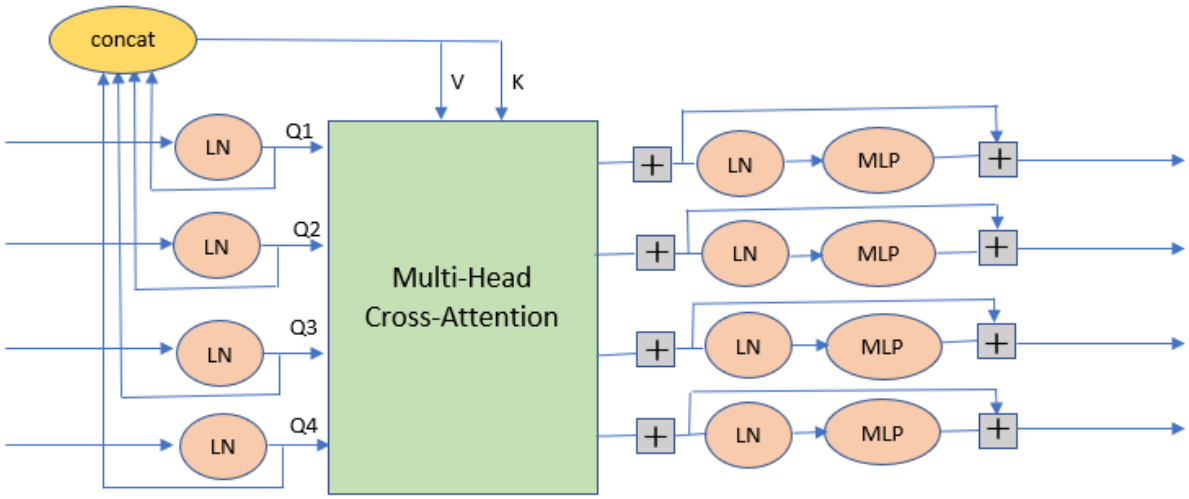
مشکلات بالا ریشه در ماهیت عملگرد کانولوشن دارند. راه‌های بسیاری برای غلبه بر این مشکلات ارائه شده‌اند، ما در این تسک می‌خواهیم یک راه مبتنی بر Transformer و مکانیزم Attention برای غنی کردن skip connection های این مدل را بررسی کنیم.



مدل پیشنهادی

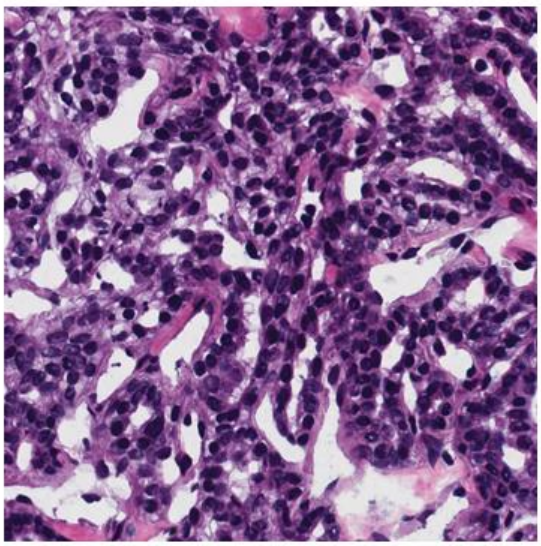
در این تسک قصد داریم connectionهای خروجی هر stage را به صورت مناسب‌تر باهم ادغام کنیم. مناسب از این منظر که هم ویژگی‌های مناسبتری یاد بگیریم و هم از نظر محاسباتی بهینه باشیم.

بهترین کار برای رسیدن به این هدف، پیاده‌سازی یک معماری Transformer می‌باشد که برروی بُعد channel های بردارهای ورودی اعمال می‌شود. در این صورت، هم از نظر محاسباتی پیچیدگی کمتری داریم و همچنین با استفاده از Key و Value های مشترک، فضای ویژگی مناسبی که می‌تواند چالش‌های ذکر شده را نیز یاد می‌گیریم. در شکل زیر نمایی از معماری پیشنهادی آورده شده است.

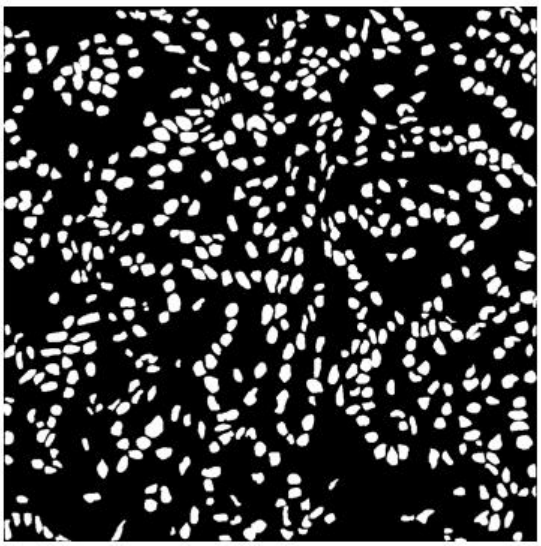


در این تسک، می‌خواهیم مسئله‌ی ناحیه‌بندی معنایی را برای ناحیه‌بندی تصاویر مولکولی به کار ببریم

نمونه‌هایی از تصاویر ورودی و برچسب‌های آن‌ها در ادامه آورده شده است:



تصویر ورودی



برچسب

در این تسک که دسته‌بندی پیکسل‌های تصویر بین دو کلاس 0 و 1 است. تصاویر به مدل Unet داده می‌شوند و در انتهای مدل، پیش‌بینی مدل آماده می‌شود. به ازای هر Stage یک بردار به ابعاد

(batch_size, channel, heught, width) از قسمت encoder مدل گرفته می‌شود و وارد transformer می‌شوند. بعد از اعمال transfomrer، خروجی 4 بردار به ابعاد (batch_size, channel, height, width)

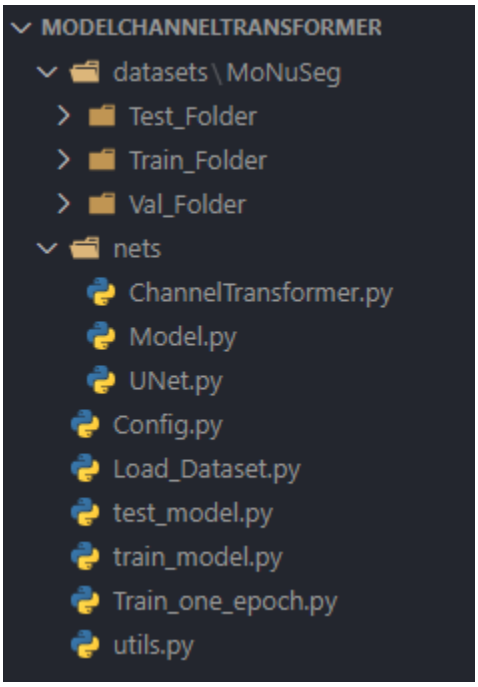
هم‌اندازه‌ی بردارهای ورودی خواهند بود که هر کدام از بردارها به stageهای متناظر در قسمت decoder وارد خواهند شد.

نحوه‌ی حل مسئله

ابتدا فایل zip را از آدرس زیر دانلود کرده و از حالت فشرده خارج کنید. سپس مسیر پوشه‌ی جاری را به درون پوشه‌ی modelChennelTransformer تغییر دهید.

project

در این پوشه، شما با ساختار زیر روبرو هستید:



در این ساختار، مدل channel-wise ViT پیشنهادی، درون فایل nets/ChannelTransformer.py پیاده‌سازی شده است و شما قرار است این فایل را تکمیل نمایید. در فایل nets/UNet.py ستون‌فقرات شبکه‌ی عصبی Unet پیاده‌سازی شده است. درون فایل net/Model.py کل مدل سرهم شده و ما از هر stage مدل Unet خروجی گرفته و آن‌ها را به مدل transformer پیاده‌سازی شده توسط شما پاس خواهیم داد. سپس خروجی آن را گرفته و به قسمت decoder مدل خواهیم داد.

درون فایل train_model.py حلقه‌ی اصلی آموزش مدل پیاده‌سازی شده است. براساس تنظیماتی از مدل که درون فایل Config.py تنظیم شده‌اند، مدل درون فایل train_model.py آموزش داده خواهد شد.

آموزش مدل شامل چندین epoch و هر epoch شامل چندین iteration است. هر epoch عبارت است از یکبار دیدن کل dataset توسط مدل. به دلیل محدودیت‌های سخت‌افزاری، نمی‌توان تمام دیتاست را به یکباره به مدل داد. برای همین دیتاست را به چندین batch شکسته و درون iterationهای مختلف batchهای دیتاست را توسط مدل می‌بینیم. بنابراین بعد از دیدن تمام batchهای موجود درون iterationهای تعریف شده، یک epoch تکمیل می‌شود. انجام هر epoch (انجام تمام iterationها) درون فایل Train_one_epoch.py انجام می‌شود. این فایل درون فایل train_model.py فراخوانی می‌شود تا epochهای آموزشی مدل تکمیل شوند.

(برای اطلاعات بیشتر از پیاده‌سازی حلقه‌ی آموزشی در چارچوب نرم‌افزاری pytorch، نحوه‌ی پیاده‌سازی Unet درون pytorch و نیز مدل نهایی از Unet و channel-wise transformer درون فایل Model.py به فایل‌های مربوطه رجوع کنید و محتویات آن‌ها را مطالعه نمایید.)

نحوه‌ی پیاده‌سازی Channel-wise ViT

برای اینکار تمامی راهنمایی‌های ممکن درون فایل nets/ChannelTransformer.py آورده شده است. اما در ادامه کلیت کار نیز اشاره می‌شود.

کلاس ChannelTransformer مدلی را تعریف می‌کند که ترکیبی از patch based embeddings، یک transformer encoder، و فرآیند reconstruction را برای مدیریت feature mapها در spatial resolution مختلف ارائه می‌دهد. این معماری برای پردازش ورودی‌های multiscale (برای مثال، feature mapها از stage مختلف یک شبکه عصبی کانولوشنی) طراحی شده است، به این صورت که feature map ها را به patch ها تقسیم می‌کند، آن‌ها را embedded می‌کند و از transformer encoder برای گرفتن وابستگی‌های بلندمدت بین کانال‌ها استفاده می‌کند. در نهایت، هر feature map پردازش‌شده را به spatial resolution اولیه خود بازسازی می‌کند. در هنگام ساخت شی‌ای از کلاس Channel_Embedding و Reconstruc لطفا به تنظیم درست ابعاد ورودی‌های مختلف آن توجه کنید.

کلاس Encoder یک ساختار مبتنی بر ترانسفورمر را تعریف می‌کند که multi-scale feature embedding را در سطوح مختلف با استفاده از یک stack از بلوک‌های ترانسفورمر پردازش می‌کند. این معماری برای مدیریت

ورودی‌های multi-scale (مانند feature map ها از لایه‌های مختلف یک شبکه عصبی کانولوشنی) طراحی شده است، به این صورت که هر سطح را از یک سری بلوک‌های ترانسفورمر (Block_ViT) عبور می‌دهد. هر بلوک، وابستگی‌ها و الگوها را درون و بین spatial patch در feature map شناسایی می‌کند.

کلاس Block_ViT یک بلوک Vision Transformer (ViT) را تعریف می‌کند که multi-scale feature embedding را با استفاده از attention در سطح کانال و لایه‌های feed-forward پردازش می‌کند. این بلوک برای بهبود feature representation طراحی شده است، به طوری که وابستگی‌ها را درون و بین کانال‌ها در spatial scale مختلف شناسایی می‌کند و از نرمال‌سازی و multi-head attention بهره می‌برد.

کلاس Block_ViT نرمال‌سازی لایه، attention در سطح کانال و لایه‌های feed-forward را در یک بلوک ترانسفورمر چندمقیاسی ترکیب می‌کند. این بلوک از اتصالات residual برای حفظ اطلاعات فضایی استفاده می‌کند و خروجی‌های پایدار و غنی را فراهم می‌کند که وابستگی‌ها را در مقیاس‌های فضایی و کانال‌های ویژگی مختلف ثبت می‌کند. این ساختار تعبیه‌های ویژگی را با در نظر گرفتن وابستگی‌های محلی و جهانی در هر سطح سلسله‌مراتبی بهبود می‌بخشد.

کلاس Attention_org با انجام multi-head attention در کانال‌های مختلف ویژگی‌ها، موارد زیر را انجام می‌دهد:

- تنظیم لایه‌های مجزا برایquery ، key، و value برای هر کانال و هر head که به آن امکان پردازش ورودی‌های چندمقیاسی را می‌دهد.
- محاسبه attention score با مقایسه کوئری‌ها با کلیدها، سپس وزن‌دهی به مقادیر (values) بر اساس این امتیازات برای تولید نقشه‌های ویژگی با آگاهی از concept.
- تبدیل خروجی‌های وزن‌دهی‌شده توجه برای هر کانال به قالب اصلی‌شان، و آماده‌سازی آن‌ها برای پردازش بیشتر.

این رویکرد به مدل امکان می‌دهد تا وابستگی‌های پیچیده را درون و بین کانال‌ها شناسایی کند و conceptual and spatial representation را در هر سطح ویژگی بهبود بخشد.

برای اجرا این کد بر روی گوگل کولب، ابتدا با دستور gdown فایل پروژه را دانلود و آن را از حالت zip خارج کنید.

سپس پوشه‌ی جاری را به درون پوشه‌ی پروژه تغییر داده و وابستگی‌های نرم‌افزاری لازم مانند tensorboardX

و ml_collections در نهایت تنها کافی است فایل train_model.py را اجرا کنید که در این صورت با خروجی‌ای مانند زیر روبرو خواهید شد:

```
/content/modelChannelTransformer/Train_one_epoch.py:84: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if epoch % config.vis_frequency == 0 and logging_mode is 'Val':
scratch
transformer head num: 4
transformer layers num: 4
transformer expand ratio: 4
log dir:

========= Epoch [1/2001] =========
Test_session_11.14_09h18
Training with batch size : 4
   [Train] Epoch: [1][1/6]  Loss:0.610 (Avg 0.6097) Dice:0.3702 (Avg 0.3702) LR 1.00e-03   (AvgTime 3.8)
   [Train] Epoch: [1][2/6]  Loss:0.565 (Avg 0.5871) Dice:0.4776 (Avg 0.4239) LR 1.00e-03   (AvgTime 2.1)
   [Train] Epoch: [1][3/6]  Loss:0.459 (Avg 0.5445) Dice:0.5977 (Avg 0.4818) LR 1.00e-03   (AvgTime 1.6)
   [Train] Epoch: [1][4/6]  Loss:0.428 (Avg 0.5154) Dice:0.5731 (Avg 0.5047) LR 1.00e-03   (AvgTime 1.3)
   [Train] Epoch: [1][5/6]  Loss:0.417 (Avg 0.4956) Dice:0.6238 (Avg 0.5285) LR 1.00e-03   (AvgTime 1.2)
   [Train] Epoch: [1][6/6]  Loss:0.468 (Avg 0.4911) Dice:0.5523 (Avg 0.5325) LR 1.00e-03   (AvgTime 1.1)
Validation
   [Val] Epoch: [1][1/2]  Loss:0.583 (Avg 0.5825) Dice:0.0155 (Avg 0.0155) (AvgTime 1.0)
   [Val] Epoch: [1][2/2]  Loss:0.606 (Avg 0.5902) Dice:0.0040 (Avg 0.0117) (AvgTime 0.7)
         early_stopping_count: 0/50
```

اجازه دهید مدل با تنظیمات پیش‌فرض آموزش ببیند. در این صورت مدل خواهد توانست به معیار DICE در حدود 79.08 ± 0.67 درصد خواهد رسید. این معیار یک معیار تخصصی ارزیابی عملکرد مدل برای تسک ناحیه‌بندی معنایی می‌باشد.