



دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی برق

گزارش تمرین دوم

YOLO و VAE، UNET

نگارش

محمد رضا شهرستانی ۰۳۱۱۵۰۸۳

حسین خموشی ۰۳۱۱۵۰۸۹

استاد

دکتر شریفیان

خرداد ۱۴۰۴

چکیده

در این تمرین، سه مدل یادگیری عمیق شامل YOLO، U-Net و VAE برای تحلیل داده‌های تصویری MRI به کار گرفته شدند. مدل U-Net برای تشخیص ناحیه تومور مغزی مغزی مورد استفاده قرار گرفت. مدل VAE به منظور فشرده‌سازی و بازسازی تصاویر MRI و استخراج ویژگی‌های نهان به کار رفت. مدل YOLO نیز جهت شناسایی و موقعیت‌یابی نواحی غیرعادی یا ضایعات در تصاویر آموزش داده شد. نتایج حاصل نشان‌دهنده عملکرد مطلوب هر مدل در وظیفه‌ی خاص خود و کاربرد پذیری آن‌ها در تحلیل تصاویر پزشکی بود.

لينك كدهای گزارش:

<https://colab.research.google.com/drive/1lfqK1Xb28DPT9F2pbpXyzKU3D6g9zFOf?usp=sharing>

واژه‌های کلیدی:

YOLO، VAE، UNET، قطعه‌بندی، تشخیص ناهنجاری

صفحه

فهرست مطالعه

۱	چکیده
۵	UNET ۱
۵	۱-۱ بررسی مقاله
۱۳	۱-۲ پایگاه داده
۱۸	۱-۳ ملاک ارزیابی
۱۹	۱-۴ معماری شبکه
۲۰	۱-۵ آموزش مدل
۲۱	۱-۶ ارزیابی مدل
۲۲	VAE ۲
۲۳	۲-۱ مقاله
۲۶	۲-۲ پایگاه داده
۲۹	۲-۳ متغیرهای آموزش
۲۹	۲-۴ معماری شبکه
۳۱	۲-۴-۱ Encoder تعریف
۳۲	۲-۴-۲ Reparameterization Trick
۳۲	۲-۴-۳ Decoder تعریف
۳۲	۲-۵ آموزش مدل
۳۶	۲-۶ ارزیابی مدل
۴۴	۲-۷ تشخیص ناهمجاري
۴۷	YOLO ۳
۴۷	۳-۱ معرفی ULTRALYTICS
۴۹	۳-۲ پایگاه داده
۴۹	۳-۳ آموزش
۵۱	منابع و مراجع
۵۲	Abstract

صفحه

فهرست اشکال و جداول

۷	شکل ۱ معماری UNET
۸	شکل ۲ تقسیم‌بندی داده
۹	شکل ۳ معماری UNET-VGG16 با Transfer Learning
۹	شکل ۴ فرایند segmentation
۱۰	شکل ۵ نمودار هزینه و دقت در چهار سناریو UNET در مقایسه با UNET-VGG16
۱۱	شکل ۶ نمایش CCR و مقایسه کارایی در هر مدل
۱۲	شکل ۷ نتایج segmentation روی ۱۶ تا دیتای آزمون
۱۴	شکل ۸ نمایش اطلاعات اولیه بیمار
۱۴	شکل ۹ خروجی اصلی و mask
۱۶	شکل ۱۰ تصاویر قبل و بعد از resize
۱۷	شکل ۱۱ دیتای augment شده
۱۹	شکل ۱۲ معماری شبکه
۲۰	شکل ۱۳ نمودار مربوط به معیارها
۲۱	شکل ۱۴ خروجی برای ارزیابی مدل
۲۲	شکل ۱۵ خروجی دوم برای ارزیابی مدل
۲۵	شکل ۱۶ آزمایش‌ها
۲۵	شکل ۱۷ آزمایش‌ها
۲۷	شکل ۱۸ چند نمونه از تصاویر دیتاست
۲۸	شکل ۱۹ چند نمونه از تصاویر دیتاست بعد از resize
۳۰	شکل ۲۰ جدول معماری شبکه
۳۱	شکل ۲۱ شماتیک معماری شبکه
۳۳	شکل ۲۲ نمودار KL Divergence
۳۴	شکل ۲۳ نمودار Reconstruction Loss
۳۵	شکل ۲۴ نمودار Total Loss
۳۵	شکل ۲۵ جدول بررسی توابع هزینه
۳۶	شکل ۲۶ نمونه‌ها
۳۸	جدول ۱ معیار MSE
۳۹	جدول ۲ معیار MAE
۴۱	جدول ۳ معیار PSNR
۴۳	جدول ۴ معیار SSIM

۵۰	جدول ۵ دقیق YOLO
۴۵	تصویر ۱ تشخیص ناهنجاری‌ها ۱
۴۵	تصویر ۲ تشخیص ناهنجاری‌ها ۲
۵۰	تصویر ۳ خروجی YOLO

UNET ۱

۱- بررسی مقاله

مقاله به این می‌پردازد که تومور مغزی یکی از بیماری‌های گشته است. در اندونزی پانزدهمین بیماری گشته در مقایسه با دیگر انواع سرطان است. تشخیص تومور مغزی توسط عکس‌های MRI انجام می‌شود و دستگاه‌های مختلفی از شرکت Tesla برای گرفتن عکس‌های MRI وجود دارد. هرچه از دستگاه‌های به روزتری استفاده شود تبعاً کیفیت تصاویر بالاتر است اما غالباً هزینه دستگاه‌های جدید بسیار زیاد است. حالا روش‌های یادگیری ماشین به کمک آمده تا حتی روی دستگاه‌های نه چندان بروز نیز بتوان قسمت‌های سالم مغز و تومور را با دقت بالا از هم تفکیک کرد و به پزشکان و جراحان این امکان را داد که بتوانند با دقت بیشتری تومور مغزی شخصی را عمل کنند. این مقاله سعی می‌کند با تکنیک Image Segmentation، روی تصاویر MRI یک boundary (مرز) ترسیم کند؛ به نحوی که قسمت‌های سالم مغز از قسمت‌های ناسالم (تومور مغزی) قابل تفکیک دقیق باشد. ناحیه‌ای از مغز که تومور آنجاست را ROI (Region of interest) می‌نامند؛ یعنی ناحیه‌ای که ما روی آن تمرکز داریم. این مقاله با یک مدل ترکیبی از VGG16 و UNET که معماری Transfer Learning در آن ساده‌کرده‌اند و از UNET را در آن استفاده کرده‌اند کار را پیش ببرده است. در نهایت با قرار دادن شبکه Fully Connected در مدل ROI و non-ROI را از هم تفکیک می‌کنند. این مدل دقت نسبتاً بالایی (حدود ۹۶,۱) روی داده‌های آموزش بدست آورده است. همچنین اعتبار سنجی مدل نیز با معیار CCR (Correct Classification Ratio) مقایسه شده است.

معیار CCR نشان می‌دهد که مدل UNet-VGG16 می‌تواند تومور مغزی را با دقت میانگین ۹۵,۶۹ تشخیص دهد. مقالات دیگر در زمینه image segmentation clustering نوشته شده بود که از روش‌های به عنوان مدل پایه‌شان استفاده کرده بودند اما در اینجا نیاز به دقت بیشتری بود. یکی از تفاوت‌های اصلی این مقاله استفاده از شبکه عصبی (NN) برای classification بود. البته در مقالات دیگری نیز از CNN و Deep

FCN (Fully image segmentation استفاده شده بود اما در این مقاله از Learning semantic segmentation استفاده شده است و خروجی بسیار خوبی برای Convolutional Network) داشته است. دیتاست مورد استفاده در این مقاله RSUD است که شامل داده‌های واقعی است. در مدلی که در مقاله ارائه شده است برای کاهش زمان معماری U-Net را ساده‌تر کنند تا از لحاظ زمانی عملکرد بهینه‌تری داشته باشیم و همچنین از VGG16 به عنوان لایه encoder استفاده شده است که در مجموع ترکیب این دو با هم به علاوه Transfer learning توانسته بر مشکل classification روی داده‌های کم غلبه کند. در ادامه مقاله به روش‌های تحقیق (Research Method) پرداخته و در مورد FCN توضیحاتی ارائه داده است. در مقاله اشاره شده است که FCN یکی از روش‌های deep learning است که برای استفاده شده می‌شود. در واقع FCN توسعه یافته CNN است. CNN شامل لایه‌های fully connected و pooling، convolution و mapping است. همچنین FCN را به صورت convolution layer از fully connected پیکسل به پیکسل انجام می‌دهد و از ground truth برای تعیین کلاس هر پیکسل استفاده می‌کند.

دیتای ورودی هم به صورت image است که سه مشخصه دارد: w (عرض)، h (طول) و d (عمق). در واقع منظور از d تعداد کانال‌های تصویر است. برای مثال اگر تصویر grayscale باشد $d=1$. اگر تصویر RGB باشد $d=3$ خواهد بود. در ادامه مقاله به بررسی kernal size، بحث max pooling و ... پرداخته است. یکی از موضوعات مهم دیگری که به آن اشاره شده است بحث انتخاب معماری U-Net است که یک معماری تحت FCN است. شکل ۱ نمایی از معماری U-Net را نشان می‌دهد:

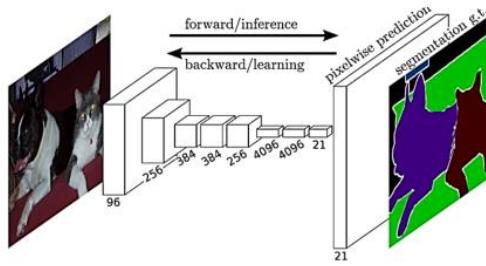
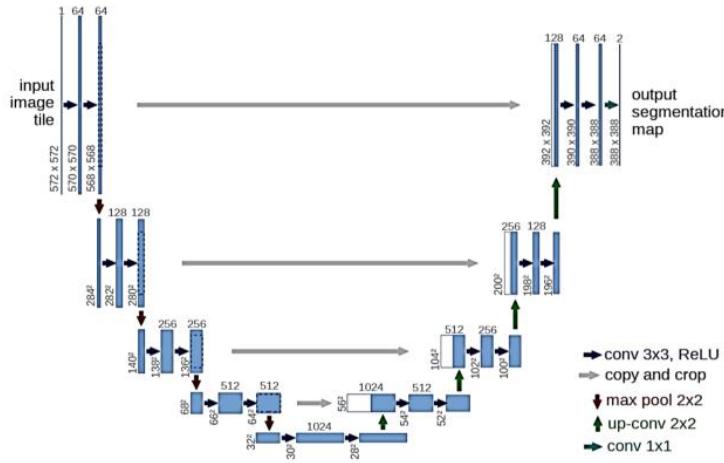


Figure 1. Structure of fully convolutional network (sourced from [14])

Figure 2. The U-Net architecture
(example for 32x32 pixels in the lowest resolution, sourced from [12])

شکل ۱ معماری UNET

به دلیل اینکه شکل آن شبیه U است نام آن را U-Net گذاشتند. در واقع سمت چپ آن به نوعی یک encoder است. داده ورودی را می‌گیرد و خروجی آن feature map/vector است. در واقع فرایند به این صورت است که encoder سایز ماتریس ورودی را کاهش و تعداد feature map ها را زیاد می‌کند در حالی که decoder سعی می‌کند ماتریس با اندازه اولیه (اصلی) را بازسازی کند و تعداد feature map ها را کاهش دهد. در نهایت خروجی پیکسل به پیکسل با ground truth مقایسه می‌شود. مقاله همچنین به مسئله Transfer learning اشاره داشته است که یک رو شی است که از مدل های از پیش آموزش دیده به عنوان نقطه شروع در مسائلی مانند language processing, computer vision استفاده می شود چرا که به دلیل نیازمند بودن به منابع محاسباتی و منابع زمانی گسترده بسیار سودمند است. علاوه بر اینها برای ساده کردن

AlexNet، شبکه‌های دیگری را نیز در نظر گرفته‌اند و ترکیب کرده‌اند. معماری‌هایی مانند LeNet و U-Net، شبکه‌هایی دیگری را نیز در نظر گرفته‌اند و ترکیب کرده‌اند. VGG-Net و ZFNET مدل را بهبود ببخشد. در این مقاله VGG-Net encoder را در بخش loss محاسبه می‌شود. در واقع مقدار خطای بین مقدار accuracy (Performance) و loss توسط مقدار واقعی است.

مقاله به مباحث مربوط به Optimization و Training پرداخته و اشاره کرده که از بهینه‌ساز adam در این مدل استفاده شده است. در واقع adam از گرادیان نزولی تصادفی استفاده می‌کند و برای بهروز سانی تمام وزن‌ها از نرخ یادگیری (Learning rate) ثابتی استفاده می‌کند. این تصمیم می‌کند که یادگیری در طول فرایند تغییر نمی‌کند. همچنین برای loss function از binary cross-entropy loss function برای طبقه‌بندی باینری (binary classification) استفاده می‌کند. در واقع binary cross-entropy loss ترکیب تابع فعال‌ساز sigmoid است که در تصویر زیر می‌بینید:



Figure 4. Division of the data

شكل ۲ تقسیم‌بندی داده

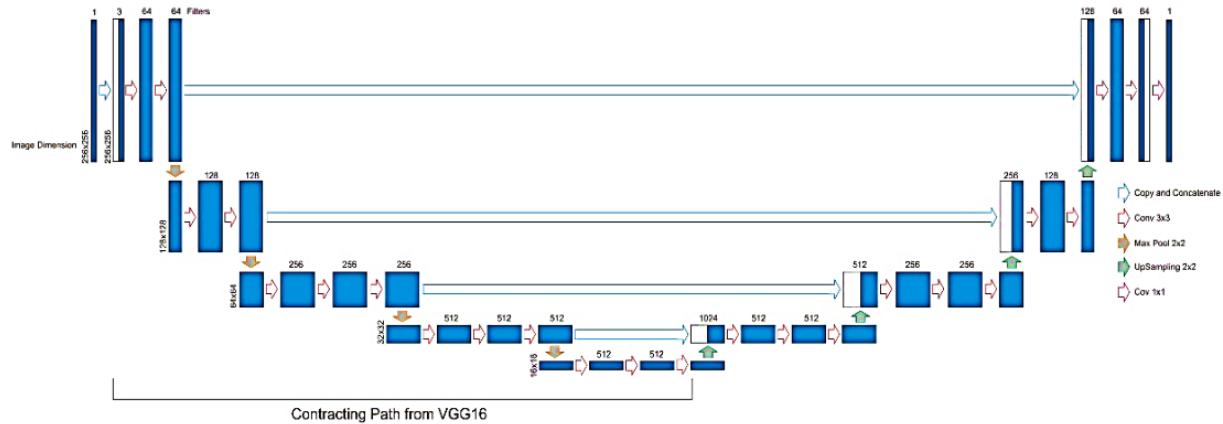


Figure 5. The architecture of UNet-VGG16 with transfer learning

شكل ۳ معماری UNET-VGG16 با Transfer Learning

در ادامه به مدل‌های U-Net و VGG16 پرداخته و مروری بر آنها داشته است و توضیح داده است که از استفاده شده است و در نهایت با ساده کردن معماری‌ها و ترکیب آنها با هم مدل UNet-VGG16 with Transfer Learning به وجود آمده است که در شکل زیر می‌بینید:

همچنین فرایندی که در مدل طی می‌شود به صورت زیر است:

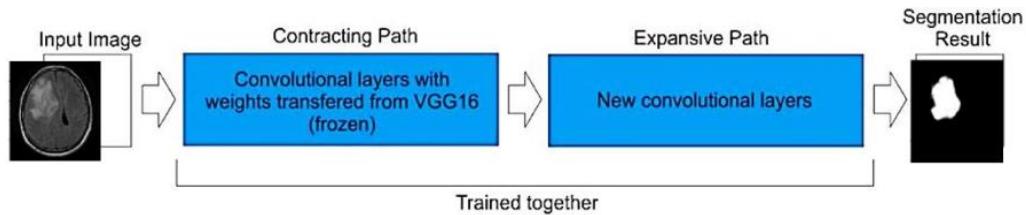


Figure 6. The segmentation process under UNet-VGG16 with transfer learning

شكل ۴ فرایند segmentation

در ادامه مقاله به بررسی مدل‌های مختلف پرداخته و سعی کرده مدلی که ارائه داده را نسبت به بقیه بسنجد و بهترین را انتخاب کند. در پیاده‌سازی تمام مدل‌های مختلفی که ارائه داده از Python با کتابخانه‌های

استفاده کرده است. در فرایند آموزش از کامپیوتر با کانفیگ زیر استفاده کرده و روی آن اجرا کرده است:

Intel Core i7, 32GB RAM, 128 GB SSD without GPU and VRAM

هر epoch به مدت ۸۰ دقیقه زمان برده است. همچنین learning curve را در دو نمودار نشان داده و مقایسه کرده است:

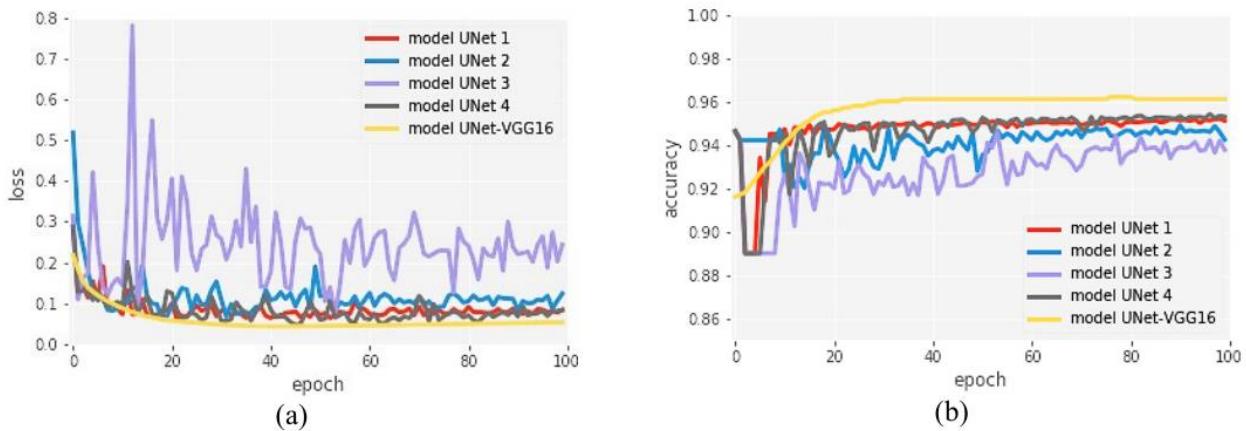


Figure 7. (a) Loss and (b) accuracy from the four U-Net scenarios compare to UNet-VGG16 with transfer learning

شکل ۵ نمودار هزینه و دقت در چهار سناریو UNET-VGG16 در مقایسه با

سپس مقاله به بررسی نتایج segmentation روی بهترین مدل پرداخته است و از پارامتر CCR که در ابتداء تعریف کردیم استفاده کرده و دو جدول ارائه داده است:

Table 3. The performance comparison of each model

Model comparison	Loss	Accuracy
U-Net :	Model 1	0.124
	Model 2	0.083
	Model 3	0.085
	Model 4	0.244
UNet-VGG16 with TL	0.054	0.961

Table 4. The summary of CCR for testing data

Testing number	CCR	Testing number	CCR
1	0.957184	9	0.957489
2	0.970047	10	0.965225
3	0.916245	11	0.961166
4	0.935806	12	0.905029
5	0.982773	13	0.933807
6	0.981598	14	0.928635
7	0.979904	15	0.985748
8	0.974136	16	0.975601

شکل ۶ نمایش CCR و مقایسه کارایی در هر مدل

این جداول همراه با Figure 8 که در ادامه آمده است نشان می‌دهد که توانسته ناحیه تومور (ROI) و اندازه آن را در نیمکرهای چپ و راست مغز تشخیص دهد. برای داده‌های تست معیار CCR را در نظر گرفته که در CCR می‌بینید. در مجموع نتایج segmentation ground truth به بسیار نزدیک شده و معیار CCR بالای ۹۰٪ است. میانگین CCR برای همه داده‌های تست ۹۵,۶۹٪ است.

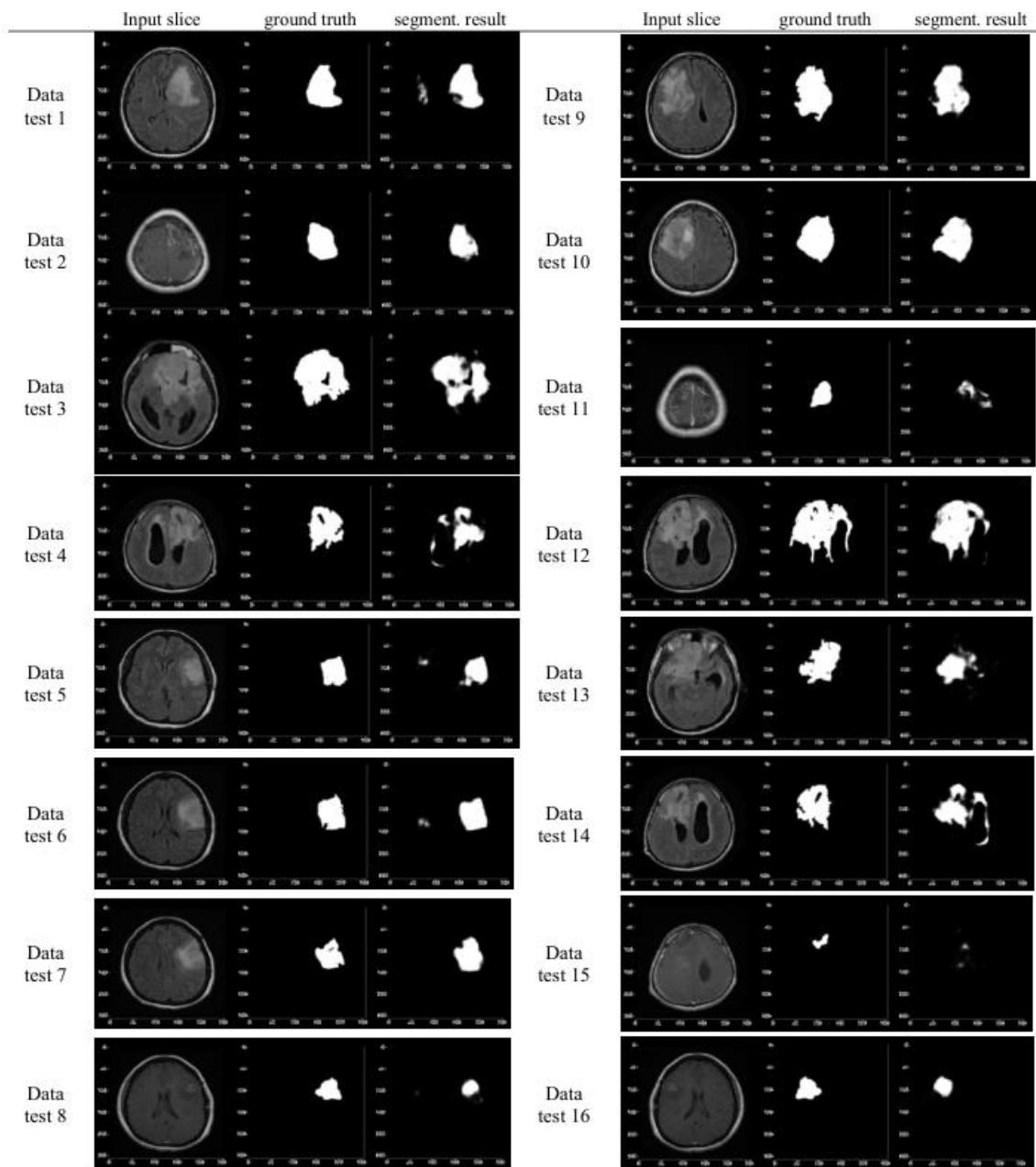


Figure 8. The segmentation results of 16 testing data

شكل ۷ نتایج segmentation روی ۱۶ تا دیتای آزمون

در ادامه مقاله به نتیجه‌گیری پرداخته است و استدلال کرده که با توجه به نتایج و آنالیزهای انجام شده می‌توانیم نتیجه بگیریم که مدل ارائه شده با نام UNet-VGG16 with Transfer Learning توانسته با اجرای روی کامپیوتری با کانفیگی که قبل اشاره شد نتایج خوبی را بدست آورد. مدل ارائه شده نسبت به چهار سناریو مختلف که برای مدل UNet در نظر گرفته شده بود عملکرد بهتری داشت و کمترین مقدار loss و حداکثر مقدار accuracy را به ارمغان آورد. برای کارهای آینده نیز پیشنهاد داده است که از معماری و بلاک‌های کانولوشنی متفاوتی استفاده شود. همچنین اشاره کرده است که هنوز هم optimum epoch برای بدست آوردن زمان محاسباتی بهینه برای ساخت و train مدل آموزشی مورد نیاز است.^[1]

۱-۲ پایگاه داده

پایگاه داده داده شده با نام Brain MRI segmentation در لینک زیر قرار داده شده است:^[2]

<https://www.kaggle.com/datasets/mateusbuda/lgg-mri-segmentation/data>

ما بعد از انجام کانفیگ‌های Kaggle، دیتاست مربوطه را دانلود و unzip کردیم و داخل پوشه‌ای با نام lgg_mri_segmentation ریختیم. حالا دیگر به خود فایل zip احتیاج نداشتیم و آنها را حذف کردیم تا فضا آزادتر شود. در مجموع در این مرحله کارهای دانلود، استخراج، پاکسازی پوشه‌ها و فایل‌های اضافی و نمایش فایل‌های آماده شده را انجام دادیم. در ادامه فایل data.csv که شامل اطلاعات متادیتا در مورد بیماران است را می‌خوانیم و سپس اطلاعات اولین بیمار را برای نمونه نمایش می‌دهیم. این اطلاعات شامل مواردی مانند جنسیت، شناسه بیمار و... است. بعد از این‌ها تصاویر MRI همراه با آن بصورت جفت نمایش داده می‌شود. به خروجی زیر دقت کنید:

```

Patient          TCGA_CS_4941
RNASEqcluster    2.0
Metylationcluster 4.0
miRcluster      2
CNCcluster       2.0
RPPAcluster      NaN
Oncosigncluster  3.0
COCluster        2
histological_type 1.0
neoplasm_histologic_grade 2.0
tumor_tissue_site 1.0
laterality        3.0
tumor_location    2.0
gender            2.0
age_at_initial_pathologic 67.0
race              3.0
ethnicity         2.0
deathH0           1.0

Name: 0, dtype: object
.....  

TCGA_CS_4941_19960909_0.mask.tif TCGA_CS_4941_19960909_20.tif  

TCGA_CS_4941_19960909_10.tif TCGA_CS_4941_19960909_21.mask.tif  

TCGA_CS_4941_19960909_11.mask.tif TCGA_CS_4941_19960909_21.mask.tif  

TCGA_CS_4941_19960909_11.tif TCGA_CS_4941_19960909_22.mask.tif  

TCGA_CS_4941_19960909_12.mask.tif TCGA_CS_4941_19960909_22.tif  

TCGA_CS_4941_19960909_12.tif TCGA_CS_4941_19960909_23.mask.tif  

TCGA_CS_4941_19960909_13.mask.tif TCGA_CS_4941_19960909_23.tif  

TCGA_CS_4941_19960909_13.tif TCGA_CS_4941_19960909_2.mask.tif  

TCGA_CS_4941_19960909_14.mask.tif TCGA_CS_4941_19960909_2.tif  

TCGA_CS_4941_19960909_14.tif TCGA_CS_4941_19960909_3.mask.tif  

TCGA_CS_4941_19960909_15.mask.tif TCGA_CS_4941_19960909_3.tif  

TCGA_CS_4941_19960909_15.tif TCGA_CS_4941_19960909_4.mask.tif  

TCGA_CS_4941_19960909_16.mask.tif TCGA_CS_4941_19960909_4.tif  

TCGA_CS_4941_19960909_16.tif TCGA_CS_4941_19960909_5.mask.tif  

TCGA_CS_4941_19960909_17.mask.tif TCGA_CS_4941_19960909_5.tif  

TCGA_CS_4941_19960909_17.tif TCGA_CS_4941_19960909_6.mask.tif  

TCGA_CS_4941_19960909_18.mask.tif TCGA_CS_4941_19960909_6.tif  

TCGA_CS_4941_19960909_18.tif TCGA_CS_4941_19960909_7.mask.tif  

TCGA_CS_4941_19960909_19.mask.tif TCGA_CS_4941_19960909_7.tif  

TCGA_CS_4941_19960909_19.tif TCGA_CS_4941_19960909_8.mask.tif  

TCGA_CS_4941_19960909_1.mask.tif TCGA_CS_4941_19960909_8.tif  

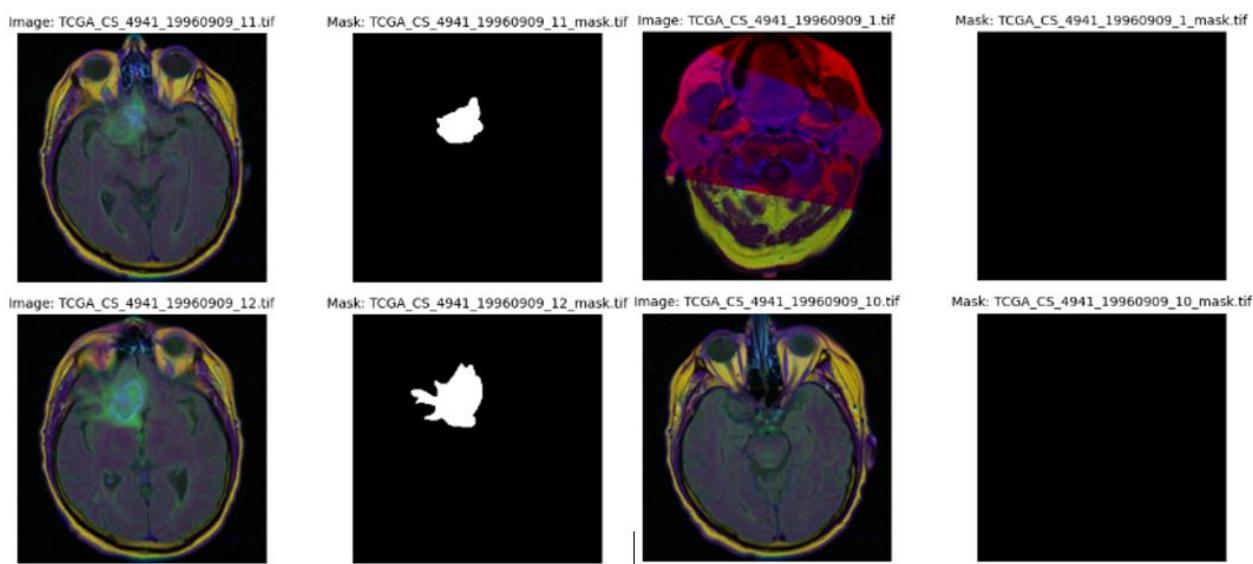
TCGA_CS_4941_19960909_1.tif TCGA_CS_4941_19960909_9.mask.tif  

TCGA_CS_4941_19960909_20.mask.tif TCGA_CS_4941_19960909_9.tif

```

شکل ۸ نمایش اطلاعات اولیه بیمار

سپس ما کدی نوشتم که ابتدا چهار تصویر MRI اول مربوط به یک بیمار خاص را از پوشه‌ای مشخص بازگذاری می‌کند. برای هر تصویر، تلاش می‌کند فایل ماسک متناظر با آن را نیز پیدا کند. سپس تصاویر و ماسک‌ها را در کنار هم به صورت زوجی (در دو ستون و چند ردیف) با استفاده از کتابخانه "No matplotlib" نمایش می‌دهد. اگر ماسک یک تصویر وجود نداشته باشد، به جای آن متن "No mask" نمایش داده می‌شود. هدف این کد بررسی کیفیت تصاویر و ماسک‌هاست، تا بتوان در مراحل بعدی مثل آموزش مدل‌های سگمنتیشن از آن‌ها استفاده کرد. خروجی بصورت زیر است:

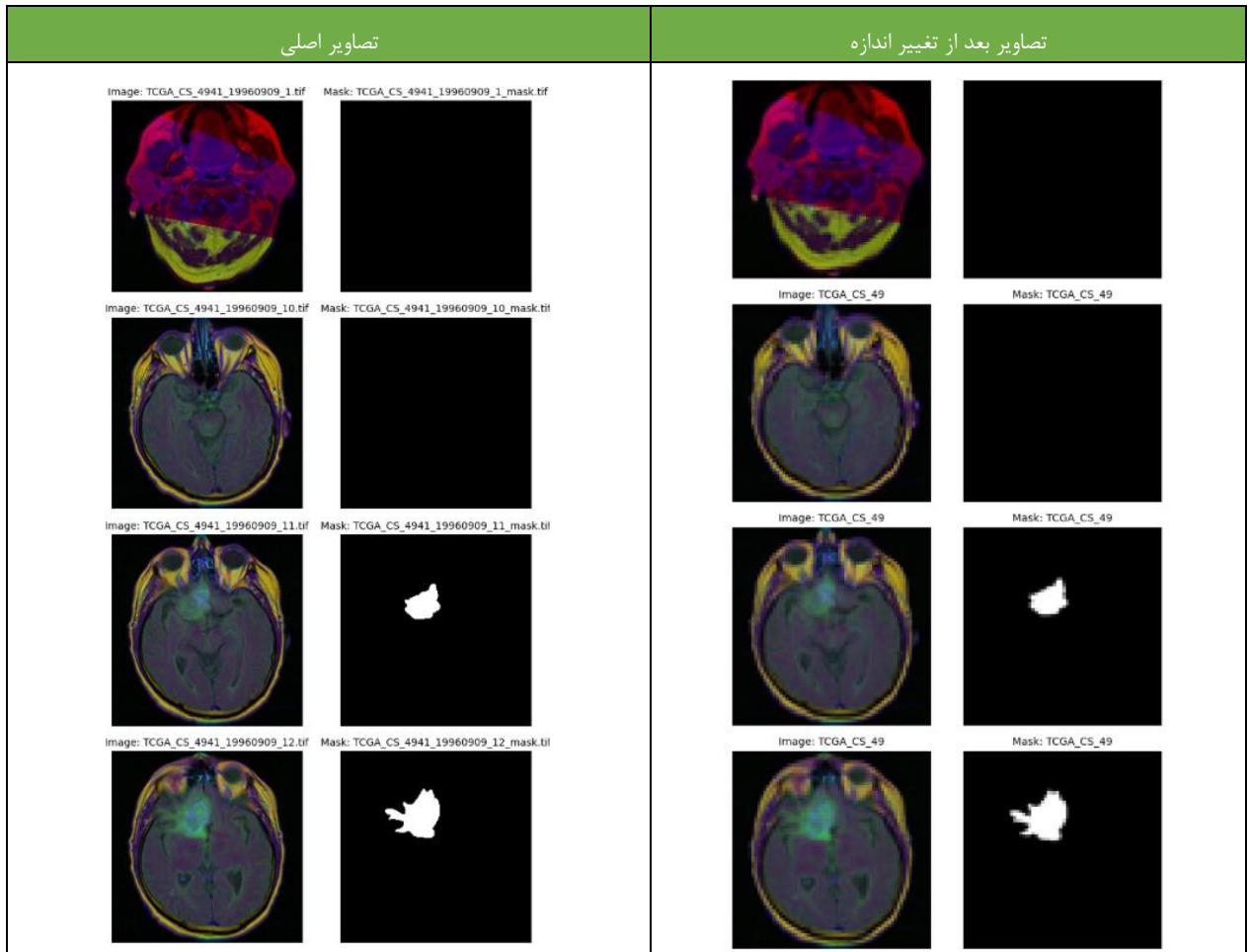


شکل ۹ خروجی اصلی و mask

تمام فایل‌های موجود در پوشه `lgg_mri_segmentation/kaggle_3m/` (و زیرپوشه‌هایش) را به صورت یکجا به یک پوشه جدید به نام `dataset` منتقل می‌کنیم. اگر فایلی با همان نام قبلاً در مقصد وجود داشته باشد، آن فایل را منتقل نمی‌کنیم و فقط پیغام "Skipped" نمایش می‌دهیم. پس از اتمام انتقال فایل‌ها، کل پوشه‌ی `lgg_mri_segmentation` را به طور کامل حذف می‌کنیم. به طور خلاصه، کارهای زیر انجام می‌شود:

- الف) همه‌ی فایل‌ها را از ساختار پوشه‌ای پیچیده خارج می‌کنیم و در یک پوشه‌ی ساده قرار می‌دهیم،
- ب) از تکراری نبودن فایل‌ها در مقصد مطمئن می‌شویم،
- ج) و در پایان داده‌های اصلی فشرده‌سازی شده را پاک می‌کنیم تا فضا آزاد شود.

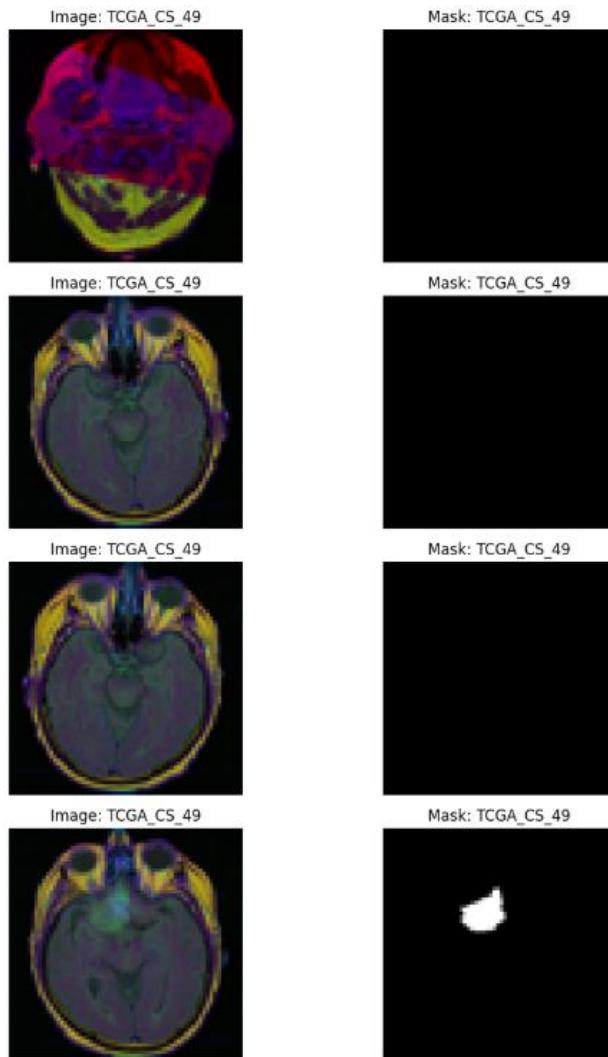
به مرحله بعد که بحث `resize` تصاویر است می‌رویم. در اینجا تمام تصاویر موجود در پوشه `dataset` را پیمایش می‌کنیم و اندازه آنها را به 64×64 پیکسل تغییر می‌دهیم. این اندازه را به این دلیل انتخاب کردیم که `trade off` ای بین زمان اجرا و کیفیت تصاویر برقرار کنیم. اگر برای مثال اندازه 128×128 را انتخاب می‌کردیم زمان زیاد و پردازش زیادی را به سیستم متحمل می‌کردیم در حالی که می‌توانیم با اندازه 64×64 تقریباً همان دقت را بدست آوریم. اگر اندازه را کوچکتر می‌کردیم کیفیت تصاویر پایین می‌آمد. بنابراین اندازه 64×64 اندازه بهینه‌ای است. بعد از `resize` تصاویر خروجی را نشان دادیم تا با اندازه اولیه مقایسه کنیم:



شکل ۱۰ تصاویر قبل و بعد از resize

بعد از انجام مراحل فوق به سراغ data augmentation می‌رویم. ما برای تصاویر MRI و mask های متناظر آنها این فرایند را انجام می‌دهیم. ابتدا یک پوشش به نام augmented_dir می‌سازیم تا دیتای Albumentations augment شده را داخل آن ببریزیم. در ادامه با استفاده از کتابخانه قدرتمند Albumentations augment شده را شامل چرخش افقی، چرخش تصادفی ۹۰ درجه، جایه‌جایی، بزرگنمایی و چرخش مختصراً، تغییر روشنایی و کنترast و افزودن نویز گاوی اعمال می‌شود. در نهایت یک نسخه augment شده از تصویر و mask تولید می‌شود. چیزی که برای ما می‌ماند پوشش augmented_data است که شامل نسخه اصلی هر تصویر و ماسک، نسخه تقویت شده هر تصویر و ماسک (تعداد آن به N_AUG بستگی دارد) است. هدف از data augmentation افزایش تنوع داده‌ها برای بهبود عملکرد مدل‌های یادگیری عمیق است.

در ادامه sample هایی از دیتای augment شده و ما سک آنها را نشان دادیم تا درک بهتری داشته باشیم:



شکل ۱۱ دیتای augment شده

در قسمت بعد آمدیم و یک مرحله بسیار مهم که همان data split است را انجام دادیم. دیتاهای را به سه مجموعه train (آموزش)، validation (اعتبارسنجی) و test (آزمون) تقسیم کردیم. نکته حائز اهمیت این است که داده‌های augment شده را فقط به مجموعه آموزش اضافه کردیم و نباید این دیتاهای را در بخش دیتاهای test بگذاریم. نسبت تقسیم داده‌ها ۷۰٪ برای آموزش، ۱۵٪ برای اعتبارسنجی و ۱۵٪ برای تست است.

۱-۳ ملاک ارزیابی

ابتدا به بررسی معیارها می‌پردازیم. معیار Dice Coefficient در واقع یک معیار شباهت است و شباهت بین دو مجموعه (در اینجا ماسک پیش‌بینی شده و ماسک واقعی) را اندازه‌گیری می‌کند. خروجی Dice Coefficient یک عدد بین صفر و یک است و هر چه به ۱ نزدیکتر باشد یعنی مدل بهتر عمل کرده است. این معیار مناسب برای داده‌های نامتوازن است (مثلًا تصاویر پزشکی که فقط بخشی از تصویر شامل تومور است). اما معیار دیگری نیز که به آن اشاره شده IoU (Intersection over Union) است. در هر یک از دو مجموعه وجود دارد) را اندازه‌گیری می‌کند. IoU نسبت به Dice سختگیرانه‌تر است. خروجی آن مانند Dice عددی بین صفر و یک است و هرچه به ۱ نزدیکتر باشد نشانه تطابق بهتری است.

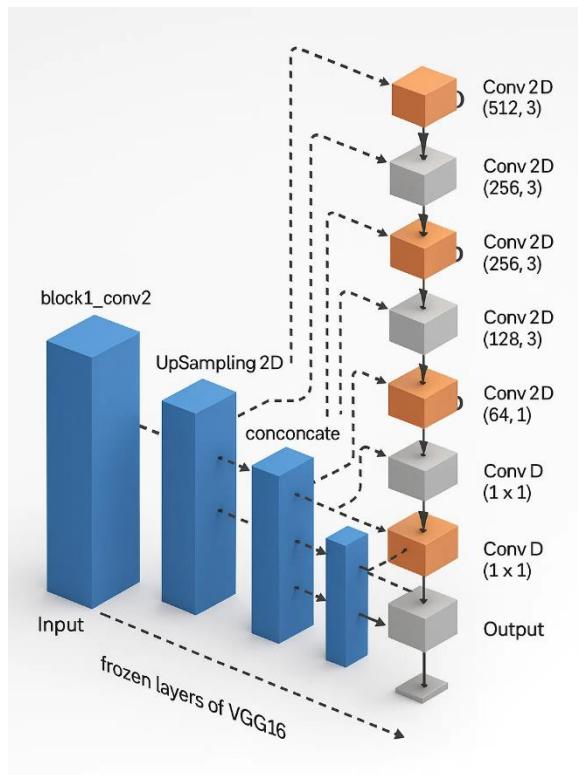
IoU	Dice	ویژگی
کمتر	بیشتر	حساسیت به داده‌های نامتوازن
$0.5 <$	$0.8 <$	مقدار معمول برای یک مدل خوب
۱	۱	مقدار، زمانی که پیش‌بینی کامل است
۰	۰	مقدار، زمانی که هیچ همپوشانی نیست

شکل ۱۲ بررسی معیارها

در راستای اشاراتی که در صورت تمرین شده معیار weighted dice coefficient را به عنوان loss در نظر گرفتیم و IoU Score را پیاده‌سازی کردیم و در کنار معیار accuracy به عنوان function شاخص‌های ارزیابی قرار دادیم. بهینه‌ساز را نیز Adam در نظر گرفتیم.

۱-۴ معماری شبکه

در این قسمت یک مدل تقسیم‌بندی تصویر (segmentation) بر اساس معماری U-Net ساختیم که در بخش encoder از مدل VGG16 با وزن‌های آموزش‌دیده روی ImageNet استفاده می‌شود. لایه‌های VGG16 را فریز کردیم تا ویژگی‌های از پیش‌آموخته شده حفظ شوند. خروجی‌های میانی VGG16 برای اتصال‌های skip در مسیر decoder استفاده شده‌اند تا جزئیات فضایی بهتر حفظ شود. سپس با استفاده از لایه‌های upsampling و convolution، مسیر decoder ساخته شده و در نهایت یک لایه خروجی باتابع فعال سازی sigmoid برای تولید ماسک باینری ارائه می‌شود. در پایان مدل کامل شده و ساختار آن را چاپ کردیم.



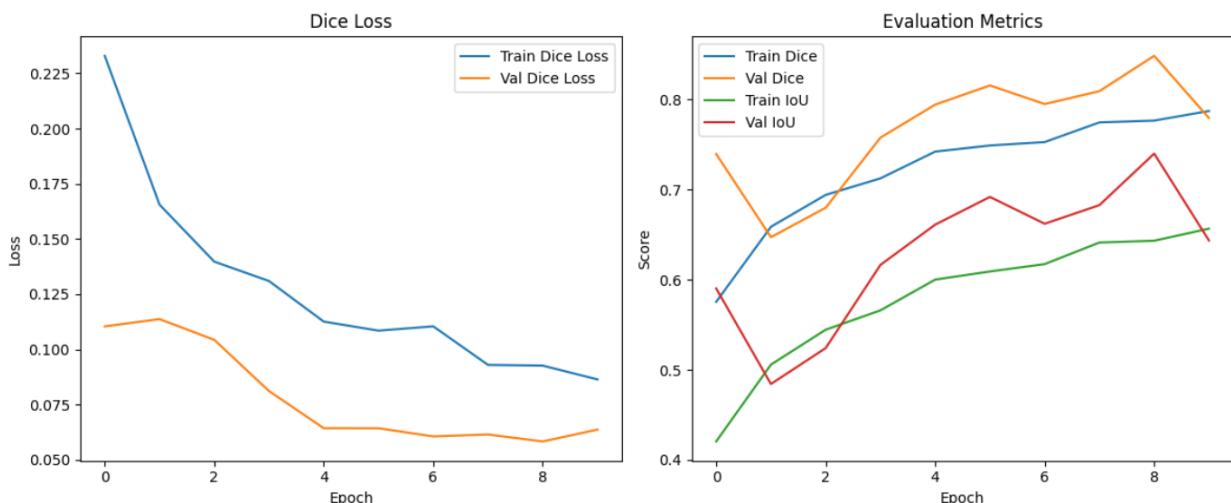
شکل ۱۲ معماری شبکه

۱-۵ آموزش مدل

آمدیم و یک Data Loader در ست کردیم. با استفاده از نام فایل تصویر (با فرمت .tif)، فایل ما سک متناظر آن (که باید پسوند _mask.tif داشته باشد) را پیدا کردیم. تصاویر و ماسکها را به ابعاد مشخص شده (64×64) تغییر اندازه دادیم. مقادیر پیکسل‌ها را نرمال‌سازی کردیم (در بازه 0 تا 1). ماسک‌ها را باینری کردیم.

(اگر مقدار پیکسل > 0.5 باشد، 1 در غیر این صورت 0) در نهایت داده‌ها را به صورت batch بارگذاری کردیم تا در حین آموزش حافظه اشغال نشود. قابلیت shuffle هم دارد تا ترتیب تصاویر در Train هر epoch تغییر کند. تعداد epoch‌ها را 10 قرار دادیم و batch size نیز 16 . سپس اقدام به مدل کردیم. بهینه‌ساز را Adam قرار دادیم.تابع هزینه را نیز مطابق صورت تمرین گذاشتیم. نمودار

خروجی روی معیارهای گفته شده به صورت زیر است:



شکل ۱۳ نمودار مربوط به معیارها

Loss (Dice Loss): 0.0839

Dice Coefficient: 0.7791

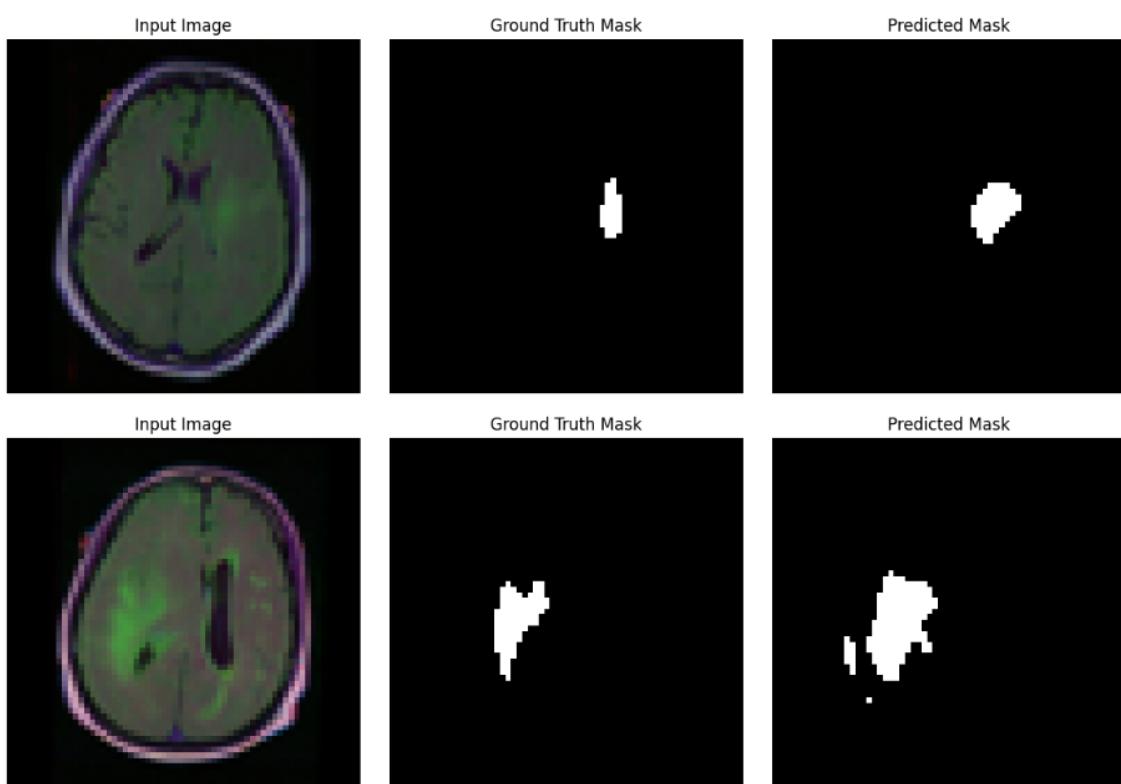
IoU Score: 0.6570

Accuracy: 0.9955

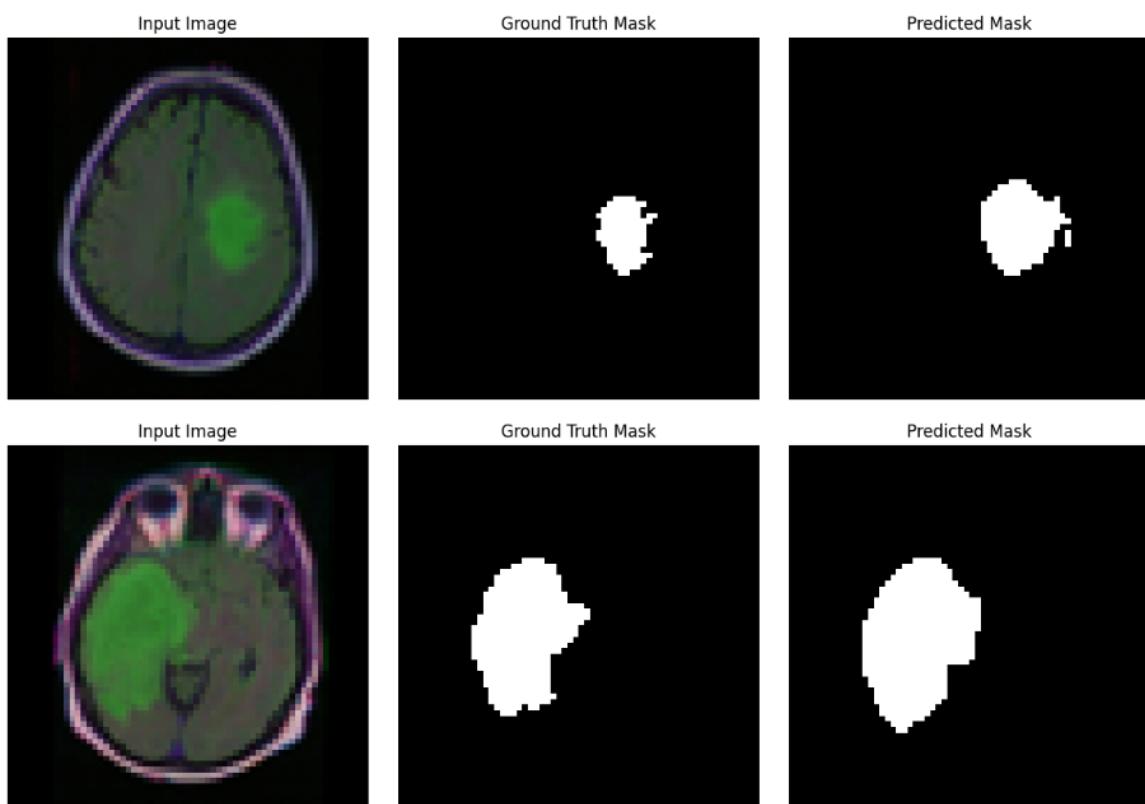
همانطور که در نمودارها مشخص است روی داده‌های Train dice loss ابتدا بالا بوده بعد به مرور کمتر شده و به 0.839 رسیده است. IoU Score و Dice coefficient نیز مطابق خواسته تمرین به بالای 0.6 رسیده است. نکته دیگر اینکه در نمودار سمت راست، Val IoU و Val Dice از لحاظ شکلی، شبیه هم هستند اما Val Dice بالاتر است و نتیجه خیلی بهتری را بدست آورده است. منظور از Train Dice و Train IoU نیز Val در نامگذاری، داده‌های اعتبارسنجی (validation) است. Val و Val IoU و Val Dice وضعیتی مشابه Val IoU و Val Dice دارند.

۱-۶ ارزیابی مدل

برای ارزیابی نتایج چند تصویر را ارائه می‌دهیم:



شکل ۱۴ خروجی برای ارزیابی مدل



شکل ۱۵ خروجی دوم برای ارزیابی مدل

همانطور که در تصاویر می‌بینید ما یک عکس ورودی، Ground truth و mask مربوط به تومور را پیش‌بینی شده را داریم. که با نگاه به تصاویر متوجه می‌شویم که مدل توانسته با دقیقیت مناسبی تومور را تشخیص دهد و خروجی مناسبی به ما بدهد.

VAE ۲

۱-۲ مقاله

مقاله‌ی معروف "Auto-Encoding Variational Bayes" است که توسط Diederik Kingma و Max Welling ، معرفی‌کننده‌ی Variational Auto-Encoder (VAE) ارائه شده است. در ادامه، به صورت خلاصه بخش‌های مختلف این مقاله را توضیح می‌دهیم.

(Introduction) مقدمه

در این بخش مقاله پرسش‌هایی را مطرح می‌کند و می‌کوشد به آنها جواب بدهد. چگونه می‌توان در مدل‌های احتمالاتی با متغیرهای پنهان پیوسته و توزیع پسین پیچیده، تخمین کارآمد و قابل مقایسه‌پذیر انجام داد؟

مقاله الگوریتمی به نام Stochastic Gradient Variational Bayes (SGVB) و نسخه‌ای بهینه‌تر به نام Auto-Encoding Variational Bayes (AEVB) ارائه می‌دهد.

(Method) روش

این بخش از مقاله خودش چهار قسمت است. به بررسی هر یک از این قسمت‌ها می‌پردازیم

الف) سناریوی مسئله

مدلی گرافی با متغیر پنهان Z و داده‌ی قابل مشاهده X در نظر گرفته می‌شود. هدف یادگیری پارامترهای مدل و تخمین پسین $p(z|x)p(x|z)$ است که معمولاً غیرقابل محاسبه است.

(Variational Lower Bound) ب) کران پایین واریاسیونی

با استفاده از کران پایین مبتنی بر KL divergence ، تابع هدف برای بهینه سازی تعریف می‌شود که شامل یک term بازسازی (reconstruction) و یک term منظم‌کننده (KL divergence) است.

ج) برآوردگر SGVB و الگوریتم AEVB

برای حل مسئله‌ی مشتق‌گیری از تابع هدف نسبت به پارامترهای φ ، از تخمینی بهینه به نام SGVB استفاده می‌شود. این تخمین به صورت مونت‌کارلو است ولی با variance پایین، قابل مشتق‌گیری و قابل آموزش با گرادیان نزولی تصادفی.

(Reparameterization Trick)

متغیر تصادفی Z از $q(z|x)$ با یک نگاشت تابعی از نویز ϵ و داده x بازنویسی می‌شود تا امکان محاسبه گرادیان فراهم شود.

مثال از (Variational Autoencoder)

در این بخش کاربرد عملی روش در قالب VAE با استفاده از شبکه‌های عصبی توضیح داده می‌شود.

Encoder • مدل $q(z|x)$ به صورت شبکه‌ای عصبی با خروجی Gaussian

Decoder • مدل $p(x|z)$ با خروجی Gaussian یا Bernoulli

Encoder • log-likelihood term به طور کامل قابل پیاده‌سازی هستند.

(Related Work)

مقایسه با الگوریتم‌هایی مثل:

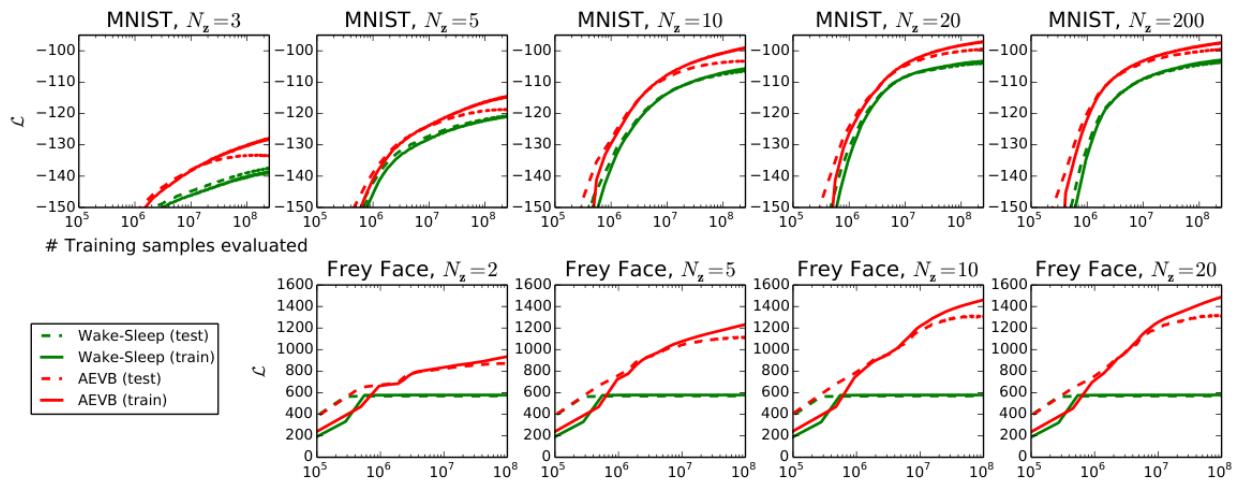
Wake-Sleep (Hinton) •

Stochastic Variational Inference •

Black-box Variational Inference •

نشان داده می‌شود که روش AEVB ساده‌تر، سریع‌تر، و قابل تعمیم به مدل‌های مختلف است.

(Experiments)

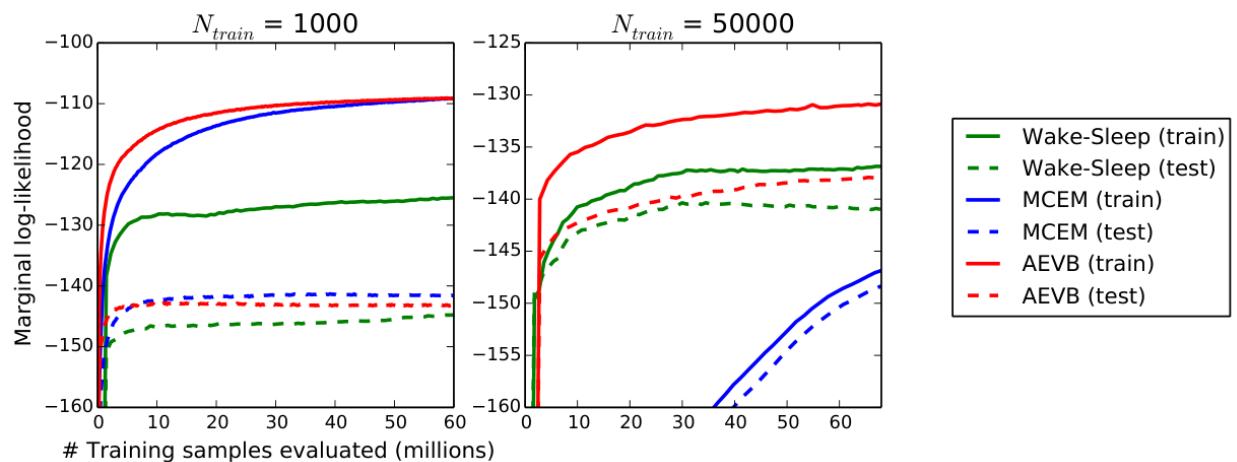


شکل ۱۶ آزمایش‌ها

روش AEVB روی دیتاستهای Frey Faces و MNIST اجرا شده است. نتایج آن به صورت زیر است.

• سریع‌تر و پایدارتر از Wake-Sleep است.

• افزایش ابعاد متغیر پنهان لزوماً منجر به overfitting نمی‌شود.



شکل ۱۷ آزمایش‌ها

در این نمودار نیز مقایسه estimated marginal likelihood (در سنتمایی حاشیه‌ای تخمینی) در سه الگوریتم Wake-Sleep و AEVB و Monte Carlo EM (MCEM) را نشان می‌دهد. به این

نکته باید توجه کرد که MCEM برخلاف دو الگوریتم دیگر یک on-line algorithm نیست و نمی‌توان به طور موثر آن را بر روی کل داده‌های دیتابس MNIST اعمال کرد. در کل اگر بخواهیم این نمودار را تحلیل کنیم متوجه می‌شویم که روش معرفی شده در مقاله یعنی AEVB دقیق‌ترین و پایدارترین الگوریتم در میان روش‌های مقایسه شده است. اگر دقت کنید فاصله بین خطاهای train و test در نمودار برای AEVB کمتر از دو روش دیگر است. البته در حجم داده کم این فاصله بیشتر است اما اگر دقت کنید با حجم داده زیاد فاصله بین دو خط کمتر است. این یعنی مدل AEVB overfit نشده و به خوبی generalize شده است. همچنین MCEM نسبت به Wake-Sleep پی‌شرفت بهتری دارد ولی نسبت به AEVB در سرعت و دقت پایینتر است. این تفاوت در دیتا است بزرگ‌تر بهتر آشکار می‌شود. همچنین Wake-Sleep ضعیفترین عملکرد را دارد. روی test و حتی train خیلی زود به plateau می‌رسد به این معنا که خیلی زود گیر می‌کند و خروجی ضعیفتری می‌دهد. در مجموع روش ارائه شده توسط مقاله یعنی AEVB نه تنها سریع‌تر یاد می‌گیرد، بلکه مدل‌های تولیدی آن نیز به طور محسوسی بهتر هستند، چه روی داده‌ی آموزش، چه روی داده‌ی آزمون.

(Conclusion)

مقاله یک روش جدید برای یادگیری مدل‌های احتمالاتی ارائه می‌دهد که:

- قابل مقیاس‌پذیر است
- بر پایه‌ی بازپارامتردهی است
- در قالب شبکه‌های عصبی (VAE) قابل پیاده‌سازی است

(Future Work)

برای کارهای آینده این مقاله پیشنهاد می‌کند که از مدل‌های سری زمانی، شبکه‌های پیچشی (CNN)، آموزش مدل‌های نیمه‌نظرارتی نیز استفاده شود.[3]

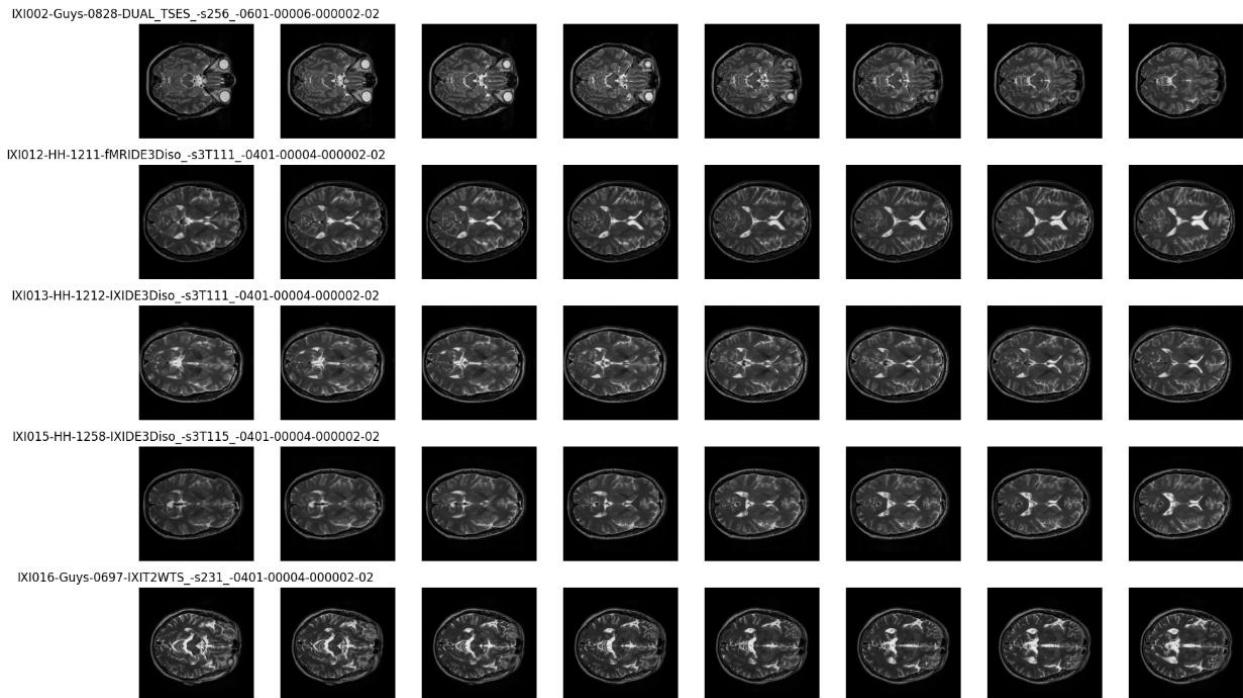
۲-۲ پایگاه داده

در این قسمت پایگاه داده با لینک زیر را دانلود کردیم:[4]

<https://www.kaggle.com/datasets/haonanzhou1/ixit2-slices>

به صورت دقیقتر ما فایل zip را از Kaggle دانلود کردیم و سپس unzip کردیم و فایل zip اصلی را حذف کردیم تا فضا آزاد شود. سپس آمدیم و چند نمونه از آن را نشان دادیم که در تصویر زیر

می‌بینید:



شکل ۱۸ چند نمونه از تصاویر دیتاست

بعد از اینها وارد مرحله پیش‌پردازش شدیم. در این مرحله ابتدا مسیر ریشه‌ی داده‌ها

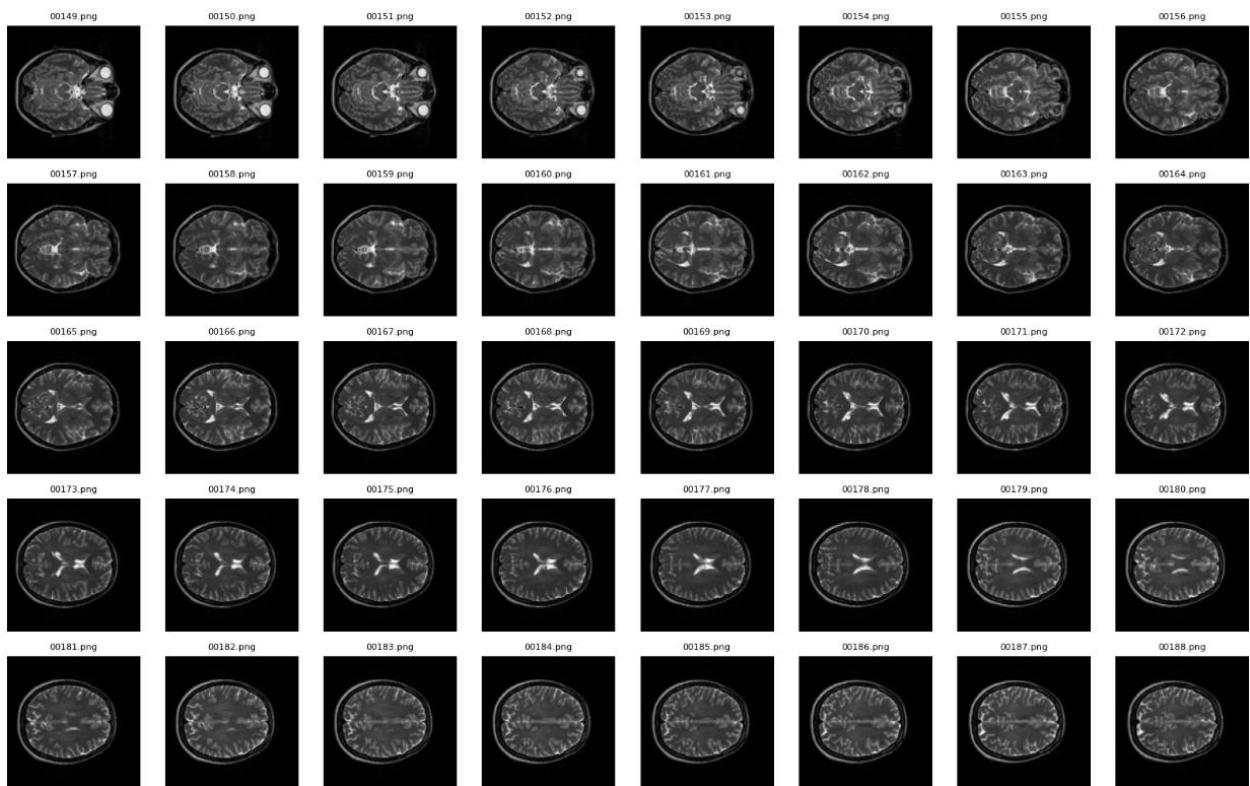
ixit2-slices/image slice-T2

که شامل پوشش‌های بیماران است، تعریف می‌شود. تمام پوشش‌های داخل آن مسیر (هر پوشش نماینده‌ی یک بیمار) شناسایی و مرتب می‌شوند. برای هر پوشش (مثلاً پوشش شماره ۵)، یک پیش‌شوند سه‌رقمی ساخته می‌شود (مثل ۰۰۵). سپس تمام تصاویر png. داخل آن پوشش بررسی می‌شوند. نام تصویر جدید با ترکیب پیش‌شوند سه‌رقمی و شماره تصویر ساخته می‌شود. مثلاً اگر نام فایل اصلی 49.png باشد در پوشش بیمار ۵، نام جدید می‌شود: 00549.png. سپس فایل از مسیر قدیمی کپی می‌شود به مسیر

جدید:

dataset/00549.png

این کار برای تمام پوشه‌ها و تصاویر تکرار می‌شود تا کل دیتا است در یک ساختار تخت (flat) مرتب و جمع‌آوری شود. در کل هدف اصلی از این کار حذف ساختار پوشه‌ای پیچیده و تبدیل آن به یک پوشه مرکزی، اطمینان از نامگذاری یکتا برای هر تصویر و مناسبسازی داده‌ها برای استفاده در شبکه‌های عصبی، data generator ها و پردازش‌های batch-based است. بعد از اتمام این کارها دیتابست اصلی را نیز پاک می‌کنیم تا فضا آزادتر شود و سپس به سراغ داده‌های استخراج شده می‌رویم. در ادامه تصاویر را نیز پاک می‌کنیم و به اندازه 128×128 می‌بریم. بعد از resize تصاویر دوباره چند sample را نمایش دادیم تا ببینیم خروجی چگونه است. در تصویر زیر می‌توانید بخشی از تصاویر resize شده را ببینید:



شکل ۱۹ چند نمونه از تصاویر دیتابست بعد از resize

فایل‌ها موجود در پوشه dataset را بر اساس شناسه‌ی بیمار (patient ID) به سه بخش train، validation (val) و test تقسیم کردیم و آن‌ها را در پوشه‌های جداگانه قرار دادیم. این تقسیم‌بندی به گونه‌ای انجام شد که تمام تصاویر مربوط به هر بیمار فقط در یکی از این سه مجموعه قرار بگیرند، نه

در چندتای آنها. در یادگیری ماشین، به ویژه در مسائل پزشکی، اگر بخشی از تصاویر یک بیمار در train و بخشی در test باشند، ارزیابی مدل غیرواقعی و خوبینانه خواهد بود. این قسمت طوری کدنویسی شده که این مشکل به وجود نماید. در مجموع ۷۰٪ داده‌ها برای آموزش، ۱۵٪ برای اعتبارسنجی و ۱۵٪ درصد نیز برای آزمون قرار داده شد. اطلاعات راجب این تقسیم‌بندی به شکل زیر است:

Total patients: 577

Train patients: 403

Validation patients: 87

Test patients: 87

۲-۳ متغیرهای آموزش

در این قسمت کدی را نوشتیم که زیرساخت اصلی تابع هزینه VAE را با استفاده از یک لایه‌ی سفارشی قابل ترکیب در مدل Keras پیاده‌سازی کرده است. به جای تعریف دستی loss هنگام compile، این لایه هنگام forward pass خودش loss را محاسبه و به مدل اضافه می‌کند. Loss بر اساس Adam نیز به عنوان تابع بهینه‌ساز استفاده کردیم.

۲-۴ معماری شبکه

برای بررسی معماری شبکه، طبق گفته صورت تمرین، جدول و شکل مناسب با معماری را اینجا ارائه می‌کنیم:

Model: "functional"

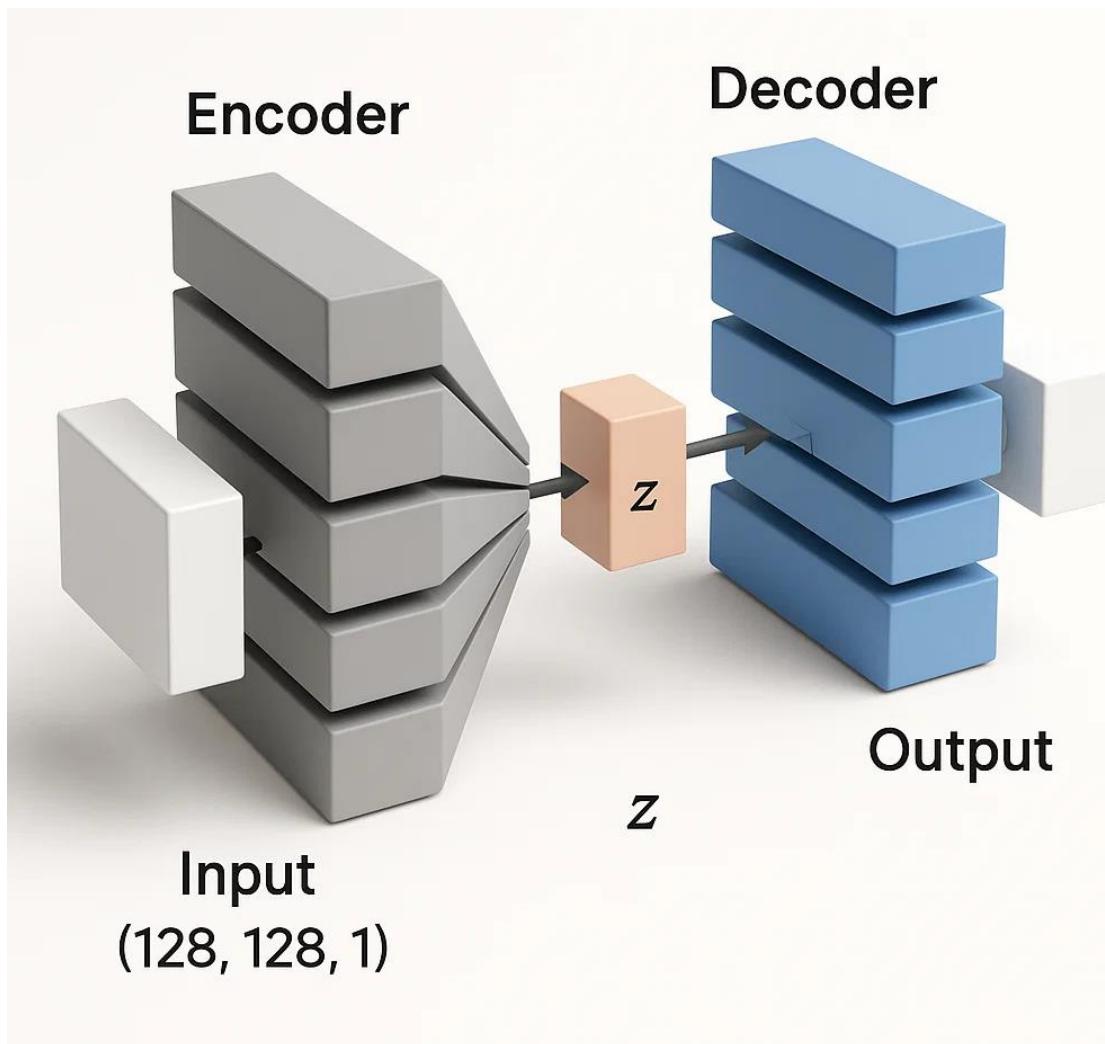
Layer (type)	Output Shape	Param #	Connected to
input_layer_20 (InputLayer)	(None, 128, 128, 1)	0	-
conv2d_30 (Conv2D)	(None, 64, 64, 32)	320	input_layer_20[0...]
conv2d_31 (Conv2D)	(None, 32, 32, 64)	18,496	conv2d_30[0][0]
flatten_10 (Flatten)	(None, 65536)	0	conv2d_31[0][0]
dense_20 (Dense)	(None, 128)	8,388,736	flatten_10[0][0]
z_mean (Dense)	(None, 10)	1,290	dense_20[0][0]
z_log_var (Dense)	(None, 10)	1,290	dense_20[0][0]
z (Lambda)	(None, 10)	0	z_mean[0][0], z_log_var[0][0]
decoder (Functional)	(None, 128, 128, 1)	776,577	z[0][0]
vae_loss_layer_10 (VAELossLayer)	(None, 128, 128, 1)	0	input_layer_20[0...] decoder[0][0], z_mean[0][0], z_log_var[0][0]

Total params: 9,186,709 (35.04 MB)

Trainable params: 9,186,709 (35.04 MB)

Non-trainable params: 0 (0.00 B)

شکل ۲۰ جدول معماری شبکه



شکل ۲۱ شماتیک معماری شبکه

Encoder ۲-۴-۱ تعریف

ورودی تصویر $(128, 128, 1)$. سپس دو لایه کانولوشنی با $\text{stride}=2$ ابعاد را نصف می‌کنند و ویژگی‌ها را استخراج می‌کنند. پس از تخت کردن(Flatten)، یک لایه Dense برای فشرده‌سازی داده‌ها به ابعاد کمتر (ویژگی‌های فشرده). سپس دو لایه جداگانه برای خروجی‌های (latent space) تعریف می‌شوند که پارامترهای توزیع نرمال پنهان $(z_{\text{mean}} \text{ و } z_{\text{log_var}})$ هستند.

Reparameterization Trick ۲-۴-۲

با استفاده از لایه Lambda و تابع sampling، از فضای پنهان نمونه‌گیری انجام می‌شود. این ترند اجازه می‌دهد گرادیان‌ها از طریق Z جریان داشته باشند و مدل آموزش‌پذیر باقی بماند.

Decoder ۲-۴-۳

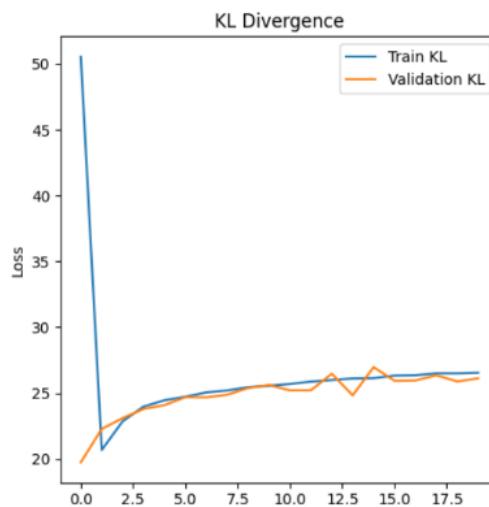
ورودی decoder، بردار Z است که از توزیع پنهان نمونه‌برداری شده. ابتدا با Dense و Reshape به شکل مناسب باز سازی می‌شود. سپس با استفاده از Conv2DTranspose ابعاد تصویر دوباره به 128×128 بازگردانده می‌شود. لایه نهایی با sigmoid خروجی را در محدوده ۰ تا ۱ می‌دهد (برای تصاویر نرمال‌شده مناسب است).

در ادامه Loss Layer را تعریف می‌کنیم. تابع loss ما از ترکیب Reconstruction loss و KL است. تابع بهینه‌ساز نیز همانطور که اشاره شد Adam گذاشتیم.

۲-۵ آموزش مدل

در این بخش آمدیم و یک data generator ساختیم. در واقع یک data loader ساختیم که تصاویر grayscale (تک کالله) را از پوشش‌هایی مانند dataset/train، dataset/val، pipeline کاملاً به صورت batch بارگذاری می‌کند و به مدل می‌دهد. در مجموع یک VAE (Variational Autoencoder) را با استفاده از داده‌های آموزش و اعتبارسنجی به مدت ۲۰ دوره (epoch) آموزش می‌دهیم. حالا به تحلیل این نمودارها می‌پردازیم.

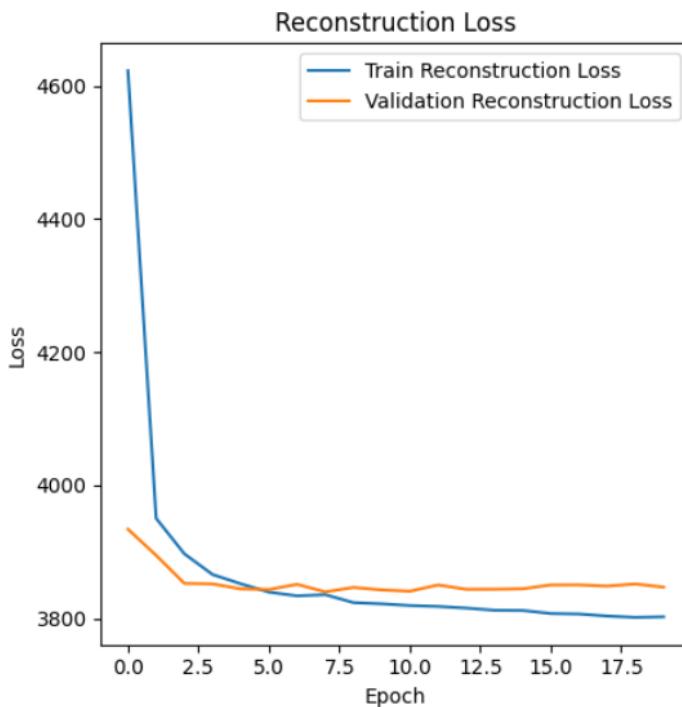
نمودار KL Divergence



شکل ۲۲ نمودار KL Divergence

در ابتدای آموزش (epoch 0)، KL به طور ناگهانی بالا است ($\text{Train KL} \approx 50$)، اما بلافاصله کاهش می‌یابد. سپس از epoch 1 به بعد، به تدریج افزایش یافته و در محدوده‌ی تقریباً ۲۵-۲۷ ثبت می‌شود. روند validation مشابه روند آموزش است، البته به جز آن قسمتی که در ابتدای آموزش می‌بینیم. در واقع validation از حدود ۲۰ شروع می‌کند سپس یک روند نرم افزایشی تا حدود ۲۵ را دارد و از آنجا به بعد تا حدودی ثابت می‌شود. این روند طبیعی است: مدل ابتدا توزیع نهان تصادفی و نامنظمی تولید می‌کند، ولی سپس تلاش می‌کند آن را به توزیع نرمال استاندارد نزدیک کند. ثبت KL نشان می‌دهد فضای پنهان به خوبی منظم شده و مدل به تعادل رسیده است.

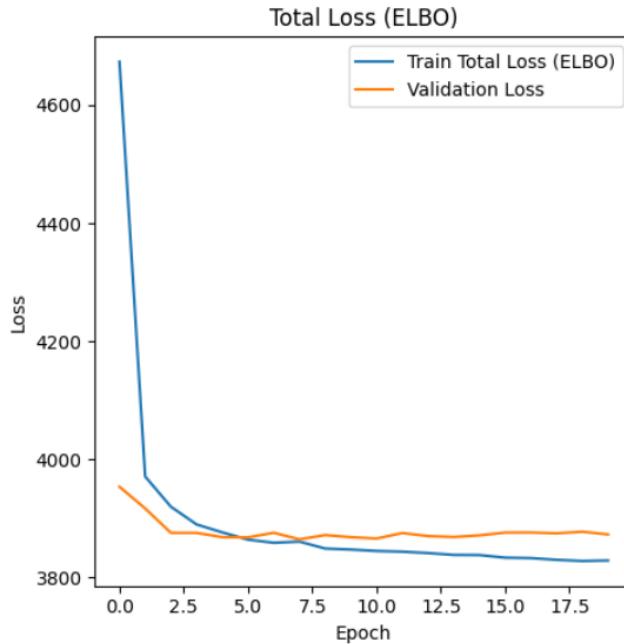
نمودار Reconstruction Loss



شکل ۲۳ نمودار Reconstruction Loss

در ابتدا (epoch 0)، مقدار بالایی دارد ($\text{Train} \approx 4600$) و به سرعت کاهش پیدا می‌کند. پس از چند epoch (حدود ۳ تا ۵)، روند کاهش متوقف می‌شود و منحنی صاف می‌شود (ثبتیت در حدود ۳۸۵۰–۳۷۸۰). روند validation تقریباً از ۳۹۵۰ شروع می‌شود و بعد از رسیدن حدودی به ۳۸۰۰–۳۷۸۰ شروع به ثابتیت شدن می‌کند. این نشان می‌دهد مدل در بازسازی تصاویر موفق عمل کرده و به خوبی یاد گرفته است. عدم افزایش مجدد یا نوسان در validation loss نشان می‌دهد overfitting رخ نداده. تفاوت بسیار کم بین Train و Val نشانه generalization خوب مدل است.

نمودار Total Loss



شکل ۲۴ نمودار Total Loss

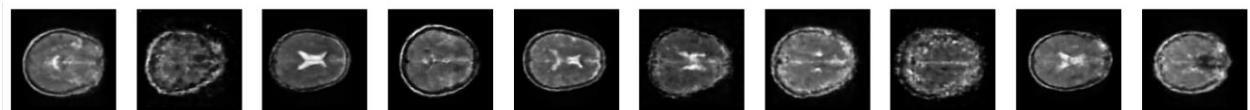
روند کلی شبیه به Reconstruction Loss است، چون سهم بیشتری در total loss دارد. کاهش اولیه سریع، سپس ثبات در حدود 3 epoch به بعد. منحنی‌های Train و Validation تقریباً نزدیک به هم حرکت می‌کنند (در ابتدا از هم فاصله دارند) و مدل به سرعت یاد گرفته و سپس در حالت پایدار قرار گرفته. نوسانات کم است و فاصله کم بین val و train بیانگر پایداری و تعادل مناسب در آموزش است. عدم کاهش شدید پس از 5 epoch نشان می‌دهد که مدل به حالت convergence (همگرایی) رسیده است.

شاخص	نتیجه
KL Divergence	به درستی از مقدار زیاد اولیه کاهش یافته و به مقدار ثابت شده‌ای رسیده است. فضای پنهان منظم شده است.
Reconstruction Loss	مدل به خوبی تصاویر را بازسازی کرده، بدون نشانه‌ای از overfitting
Total Loss (ELBO)	افت اولیه سریع و سپس پایداری – آموزش موفقیت‌آمیز با تعادل مناسب بین دو نوع loss

شکل ۲۵ جدول بررسی توابع هزینه

۲-۶ ارزیابی مدل

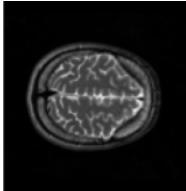
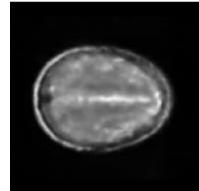
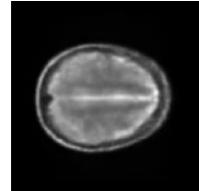
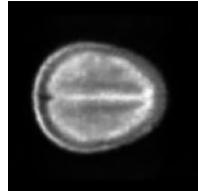
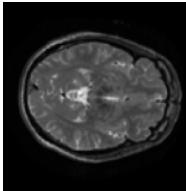
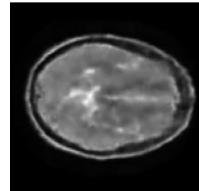
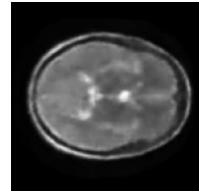
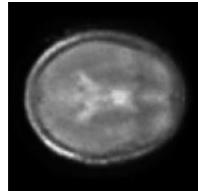
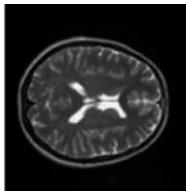
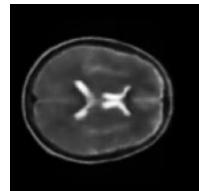
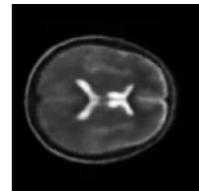
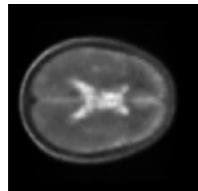
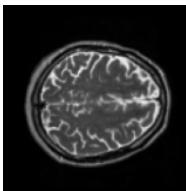
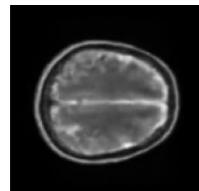
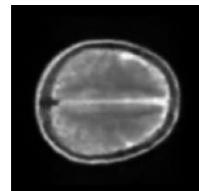
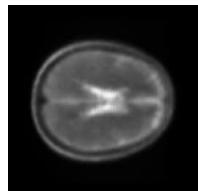
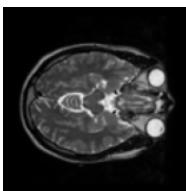
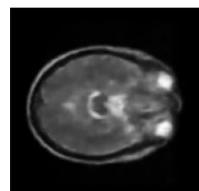
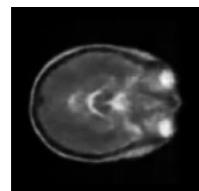
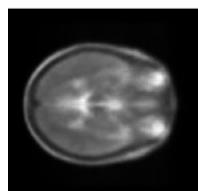
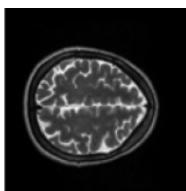
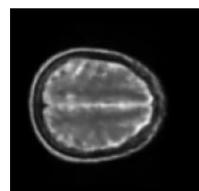
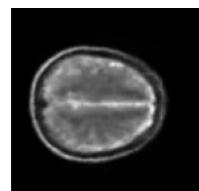
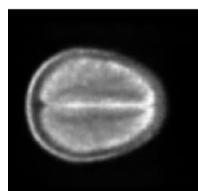
برای ارزیابی مدل، چند نمونه از نتایج را نشان می‌دهیم و از لحاظ معیارهای PSNR، MAE، MSE و SSIM بررسی می‌کنیم تا کیفیت خروجی را متوجه شویم. ما تصاویر زیر را برای بررسی انتخاب کردیم:

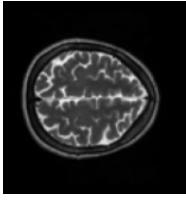
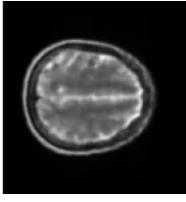
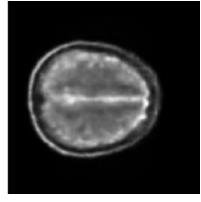
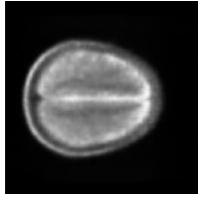


شکل ۲۶ نمونه‌ها

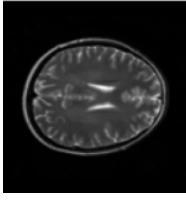
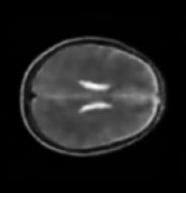
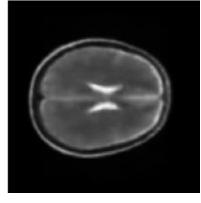
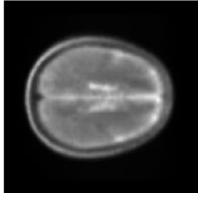
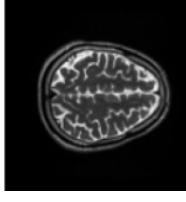
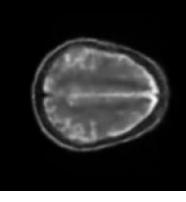
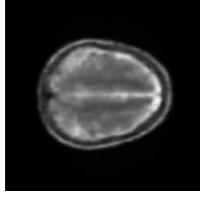
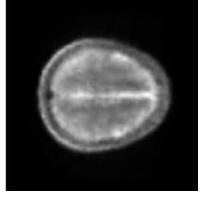
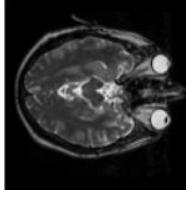
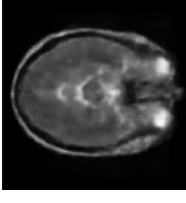
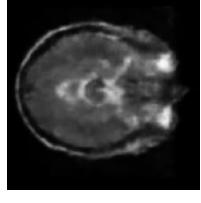
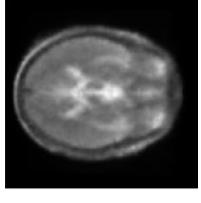
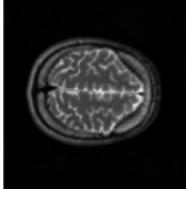
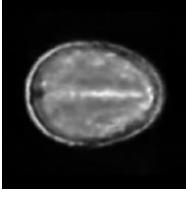
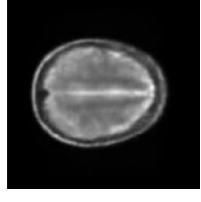
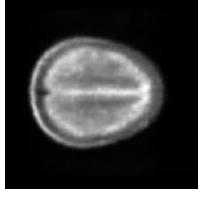
در ادامه این‌ها را در چهار جدول بررسی می‌کنیم.

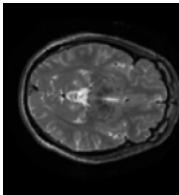
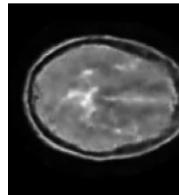
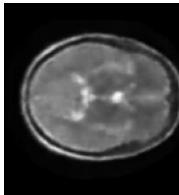
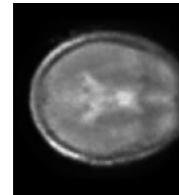
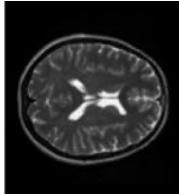
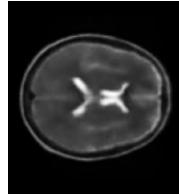
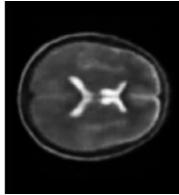
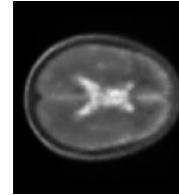
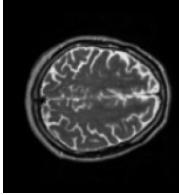
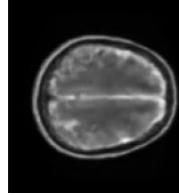
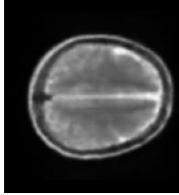
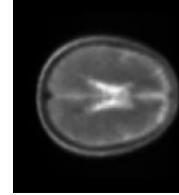
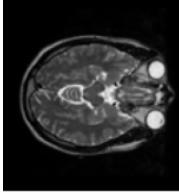
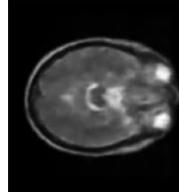
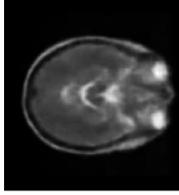
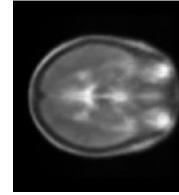
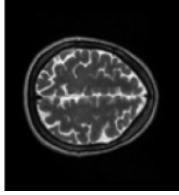
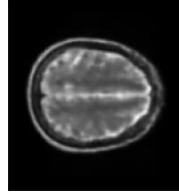
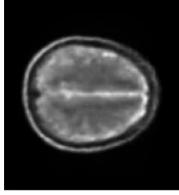
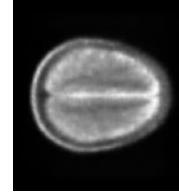
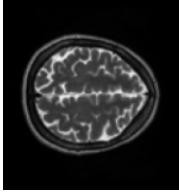
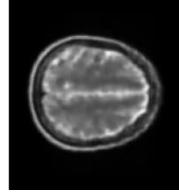
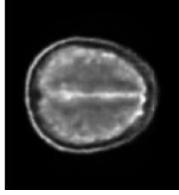
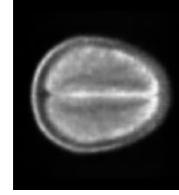
Original	Reconstructed (latent 20)	Reconstructed (latent 10)	Reconstructed (latent 2)	Best MSE
				0.0031
	0.0031	0.0034	0.0047	
				0.0084
	0.0084	0.0086	0.0091	
				0.0075
	0.0082	0.0075	0.0126	

				0.0041
0.0041	0.0045	0.0054		
				0.0044
0.0044	0.0054	0.0062		
				0.0040
0.0040	0.0041	0.0065		
				0.0085
0.0086	0.0085	0.0136		
				0.0056
0.0056	0.0058	0.0094		
				0.0071
0.0071	0.0071	0.0088		

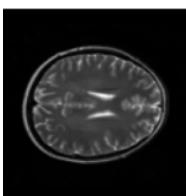
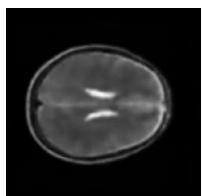
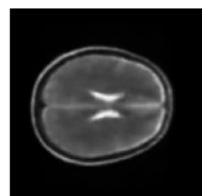
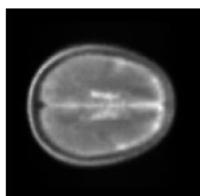
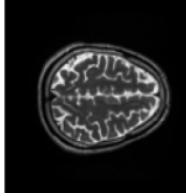
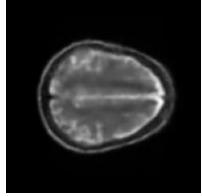
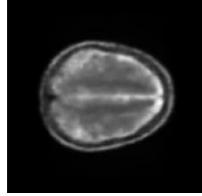
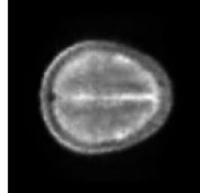
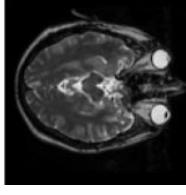
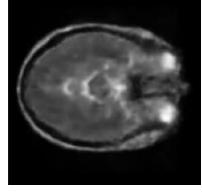
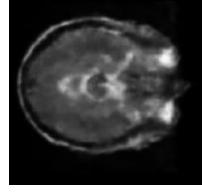
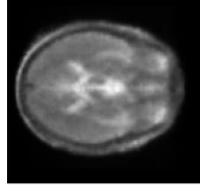
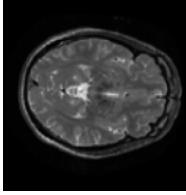
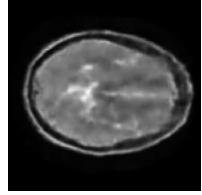
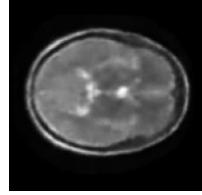
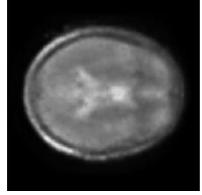
				0.0069
0.0069	0.0071	0.0088		

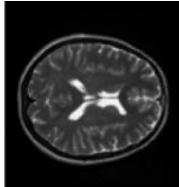
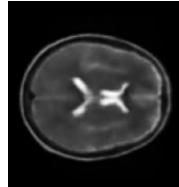
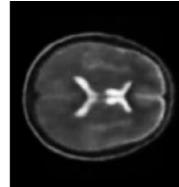
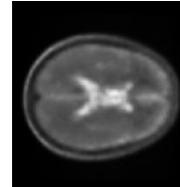
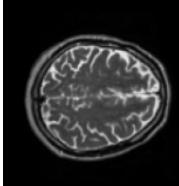
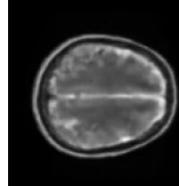
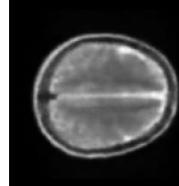
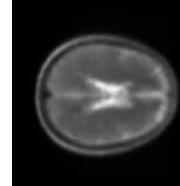
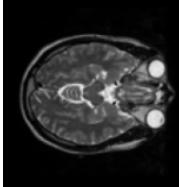
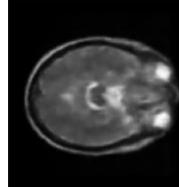
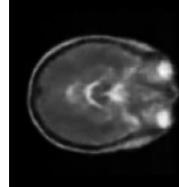
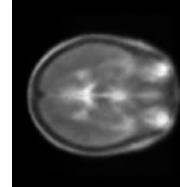
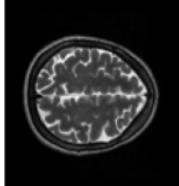
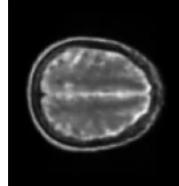
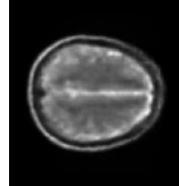
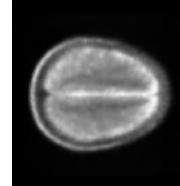
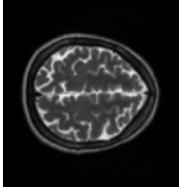
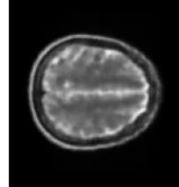
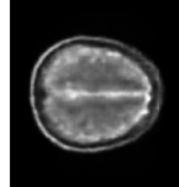
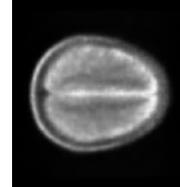
جدول ۱ معیار MSE

Original	Reconstructed (latent 20)	Reconstructed (latent 10)	Reconstructed (latent 2)	Best MAE
				0.0240
0.0240	0.0260	0.0343		
				0.0415
0.0415	0.0436	0.0429		
				0.0455
0.0490	0.0455	0.0625		
				0.0293
0.0293	0.0305	0.0357		

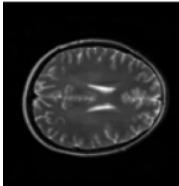
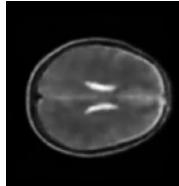
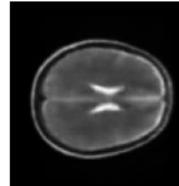
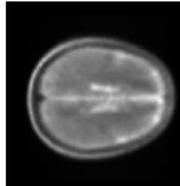
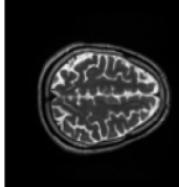
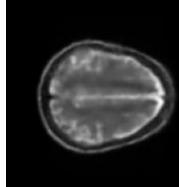
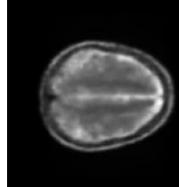
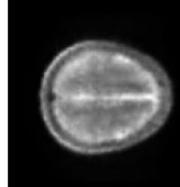
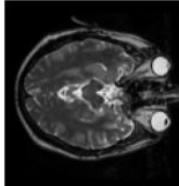
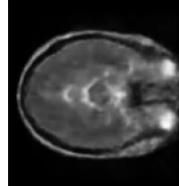
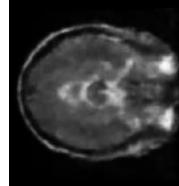
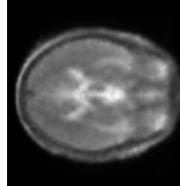
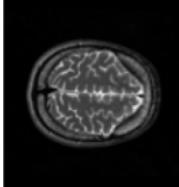
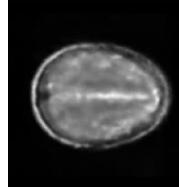
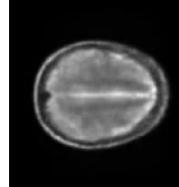
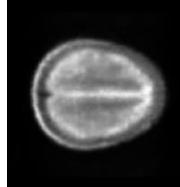
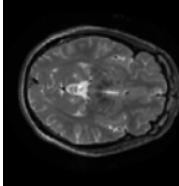
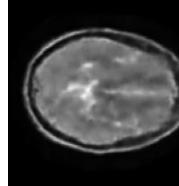
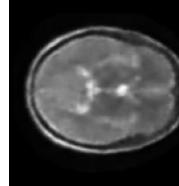
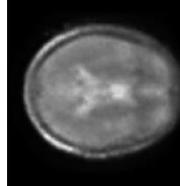
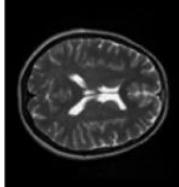
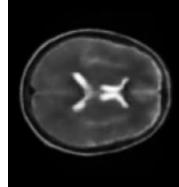
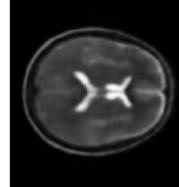
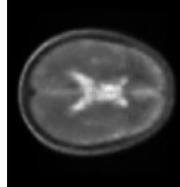
				0.0327
	0.0327	0.0370	0.0414	
				0.0281
	0.0281	0.0289	0.0426	
				0.0433
	0.0434	0.0433	0.0536	
				0.0353
	0.0353	0.0369	0.0478	
				0.0374
	0.0375	0.0374	0.0459	
				0.0368
	0.0368	0.0374	0.0456	

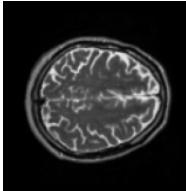
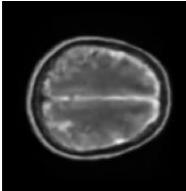
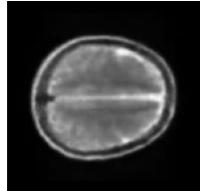
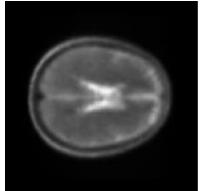
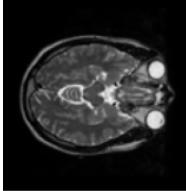
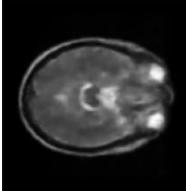
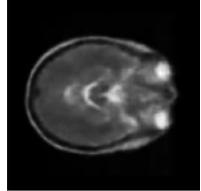
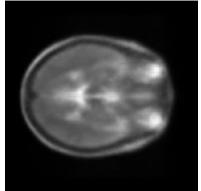
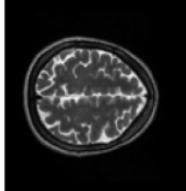
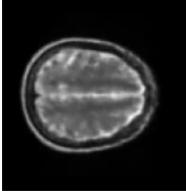
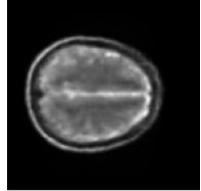
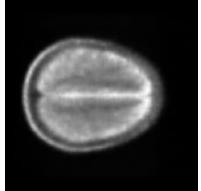
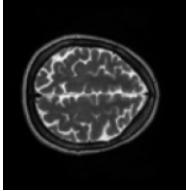
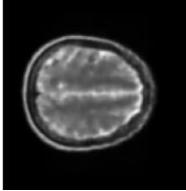
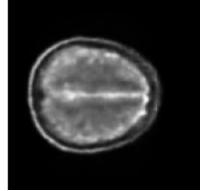
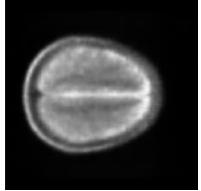
جدول ۲ معیار MAE

Original	Reconstructed (latent 20)	Reconstructed (latent 10)	Reconstructed (latent 2)	Best PSNR
				25.13dB
	25.13dB	24.72dB	23.31dB	
				20.75dB
	20.75dB	20.68dB	20.41dB	
				21.26dB
	20.85dB	21.26dB	18.99dB	
				23.92dB
	23.92dB	23.44dB	22.65dB	
				22.69dB
	23.59dB	22.69dB	22.09dB	

				24.03dB
	24.03dB	23.90dB	21.85dB	
				20.71dB
	20.65dB	20.71dB	18.67dB	
				22.54dB
	22.54dB	22.34dB	20.28dB	
				21.47dB
	21.47dB	21.46dB	20.54dB	
				21.62dB
	21.62dB	21.49dB	20.56dB	

جدول ۳ معیار PSNR

Original	Reconstructed (latent 20)	Reconstructed (latent 10)	Reconstructed (latent 2)	Best SSIM
				0.8115
	0.8115	0.7914	0.6901	
				0.7386
	0.7386	0.7226	0.6907	
				0.6337
	0.6202	0.6337	0.4842	
				0.7299
	0.7299	0.7142	0.6980	
				0.6972
	0.6972	0.6586	0.5931	
				0.7909
	0.7909	0.7849	0.6711	

				0.7224
0.7224	0.6925	0.5973		
				0.7228
0.7228	0.7166	0.6027		
				0.7689
0.7689	0.7400	0.6804		
				0.7699
0.7699	0.7347	0.6773		

جدول ۴ معیار SSIM

تحلیل

همانطور که در جداول می‌بینید هر چه latent بالاتر است کیفیت تصویر بازسازی شده در اکثریت موارد (به غیر موارد بسیار معدودی) بالاتر است. طبق جداول بالا کیفیت تصاویر بازسازی شده 20 latent بهتر از 10 latent و کیفیت تصاویر بازسازی شده 10 latent بهتر از 2 latent است. اما باید در نظر داشت که هر چقدر مقدار latent بالاتر باشد محاسبات و زمان اجرا بیشتر می‌شود. حالا ما می‌توانیم با توجه به منابع و دقیقی که نیاز داریم انتخاب کنیم. در اکثریت موارد بالا خروجی تصاویر بازسازی شده با latent 10 نزدیک است و اختلاف بسیار کمی دارد. در موارد معدودی نیز حتی بهتر از latent 20

20 عمل کرده است. با توجه به جمیع این موارد انتخاب 10 latent می‌تواند انتخاب مناسبی برای تمرین ما باشد.

۲-۷ تشخیص ناهنجاری

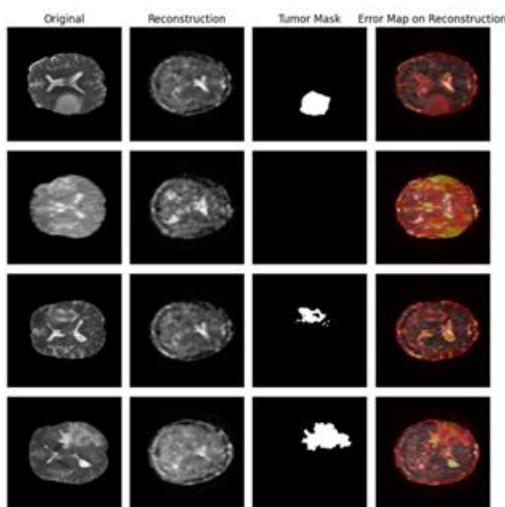
دیتابست با آدرس زیر را دانلود کردیم:[5]

<https://www.kaggle.com/datasets/awsaf49/brats20-dataset-training-validation>

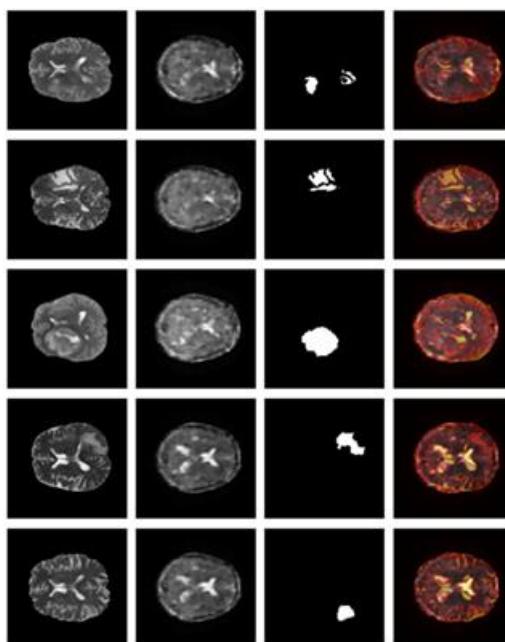
آن را استخراج کردیم و فایل zip آن را حذف کردیم تا فضای آزاد بیشتری داشته باشیم. سپس پوشه‌ها و فایل‌های درون آنها را با دستور ls بررسی کردیم تا از جزئیات دیتابست مطلع شویم. در مجموع حجم دیتابست بیشتر از ۸ گیگابایت بود. سپس کدی بسیار دقیق نوشتیم برای بررسی توانایی مدل VAE در بازسازی نواحی توموری در تصاویر MRI. اگر MSE در ناحیه تومور بالاتر از پس‌زمینه باشد (که اغلب هست)، نشان می‌دهد که مدل اطلاعات تومور را خوب یاد نگرفته است.

در مجموع در این بخش، مدل VAE آموزش‌دیده روی برش‌هایی از تصاویر MRI بیماران مبتلا به تومور مغزی از دیتابست BraTS2020 ارزیابی شد. ابتدا تصاویر T2 و ماسک تومور به صورت مرکزی از بیماران استخراج و پیش‌پردازش شدند، سپس تصاویر توسط مدل بازسازی شدند. با مقایسه تصویر اصلی و بازسازی شده، خطاهای به صورت میانگین مربعات (MSE) محاسبه شد؛ هم به صورت کلی، هم جداگانه برای نواحی توموری و غیرتوموری. نتایج نشان داد که مدل در بازسازی نواحی غیرتوموری عملکرد خوبی دارد، اما در نواحی تومور معمولاً خطای بیشتری مشاهده می‌شود. این مسئله نشان می‌دهد که مدل تمایل دارد به ساختارهای معمول مغز و فادر بماند و بخش‌های غیرمعمول (مانند تومور) را نادیده بگیرد، که می‌تواند در آینده برای شناسایی خودکار نواحی غیرطبیعی استفاده شود. تصاویر زیر مربوط به این

بخش:



تصویر ۱ تشخیص ناهنجاری‌ها ۱



تصویر ۲ تشخیص ناهنجاری‌ها ۲

همانطور که در تصاویر می‌بینید مدل VAE توانسته است ساختار کلی مغز در تصاویر MRI را نسبتاً خوب بازسازی کند؛ بافت‌های کلی، شیارها و نواحی سفید و خاکستری تا حد زیادی حفظ شده‌اند. با این حال، در مناطقی که تومور وجود دارد (مشخص شده با ماسک سفید در ستون سوم)، خطای بازسازی

بالاتری مشاهده می‌شود که در ستون آخر (error map) به صورت رنگ‌های داغ (قرمز، زرد) دیده می‌شود.

این اتفاق دو نکته مهم را نشان می‌دهد:

۱) مدل VAE با توجه به طبیعت بدون نظارت خود، تمایل دارد بیشتر به الگوهای معمول و سالم مغز وفادار بماند و نواحی غیرعادی (توموری) را بازسازی نکند یا به شکل سالم بازسازی کند. این رفتار باعث می‌شود اختلاف بین تصویر اصلی و بازسازی شده در ناحیه تومور زیاد شود.

۲) همین افزایش خطأ در ناحیه تومور به صورت طبیعی می‌تواند برای آشکارسازی نواحی غیرمعمول (anomaly detection) مورد استفاده قرار گیرد. یعنی بدون نیاز به ماسک، فقط از طریق مقایسه تصویر بازسازی شده و تصویر اصلی می‌توان نواحی مشکوک را تشخیص داد. در واقع، ناهمانگی بین بازسازی و تصویر اصلی، شاخصی از غیرطبیعی بودن ناحیه است - که این روش را برای کاربردهای anomaly detection در تصاویر پزشکی ارزشمند می‌سازد.

YOLO ۳

۱-۳ معرفی ULTRALYTICS

مدل‌های مختلف توسعه داده شده برای YOLO (You Only Look Once) در مسیر تکامل خود، با هدف بهبود دقت، سرعت، سادگی و قابلیت استفاده در پلتفرم‌های متنوع طراحی شده‌اند. در ادامه، نسخه‌های اصلی و مهم YOLO به صورت خلاصه معرفی می‌شوند:

YOLOv3 تا YOLOv1

مدل‌های اولیه توسعه یافته توسط Joseph Redmon

- YOLOv1 اولین نسخه، سریع ولی دقت نسبتاً پایین به‌ویژه در شناسایی اشیاء کوچک.

- YOLOv2 (YOLO9000) بهبود دقت با anchor boxes و قابلیت تشخیص بیش از ۹۰۰۰ کلاس.

- YOLOv3 استفاده از Darknet-53 و بهبود در تشخیص اشیاء با اندازه‌های مختلف.

YOLOv4

توسعه یافته توسط Alexey Bochkovskiy:

- ترکیبی از چند تکنیک مدرن مانند CSPNet و Mish activation ، augmentation.

- دقت و سرعت بالا برای پردازش در زمان واقعی روی GPU و حتی CPU.

YOLOv5 (Ultralytics)

مدل بازنویسی شده در PyTorch توسط Ultralytics :

- پشتیبانی از چهار سایز مختلف x, 1, m, 5s : yolov5s, yolov5m, yolov5l, yolov5x
- آموزش و استفاده بسیار ساده با دستورهای CLI و API پایتون

- مناسب برای deployment در edge devices و اپلیکیشن‌های سبک یا دقیق.

YOLOv6 (Meituan)

متمرکر بر کاربردهای صنعتی و inference سریع:

- بهینه‌شده برای deployment در edge
- سرعت بالا و latency پایین در کاربردهای real-time.

YOLOv7

یکی از دقیق‌ترین نسخه‌ها با ساختار سبک‌تر:

- پشتیبانی از multi-task تشخیص، segment و pose)

benchmark ترکیب تکنیک‌های مدرن و کارایی بالا در تست‌های

YOLOv8 (Ultralytics)

نسخه بازطراحی‌شده با ساختار anchor-free:

- پشتیبانی رسمی از detection، classification و segmentation
- estimation pose

• افزایش دقت و سادگی آموزش و استنتاج

• رابط یکپارچه و پایتونی با hub Ultralytics و CLI.

YOLOv9

جدیدترین نسل YOLO با معماری جدیدی به نام **Dy-Head** و ماژول‌های پیشرفته:

- ترکیب تطبیق‌پذیری با دقت بالا
- عملکرد بهتر در مواردی با منابع محدود (مانند موبایل).

YOLO-World

مدل مبتنی بر زبان که از Open-Vocabulary پشتیبانی می‌کند:

- می‌تواند اشیایی را شناسایی کند که در دیتابیس آموزشی نبوده‌اند
- مناسب برای سناریوهایی با کلاس‌های ناشناخته یا پویا.

در مجموع، مسیر توسعه YOLO از یک مدل سریع اما ساده، به مجموعه‌ای از مدل‌های دقیق، ماژولار و قابل توسعه رسیده که کاربردهایی از تشخیص اشیاء تا طبقه‌بندی و segmentation را پوشش می‌دهند.^[6]

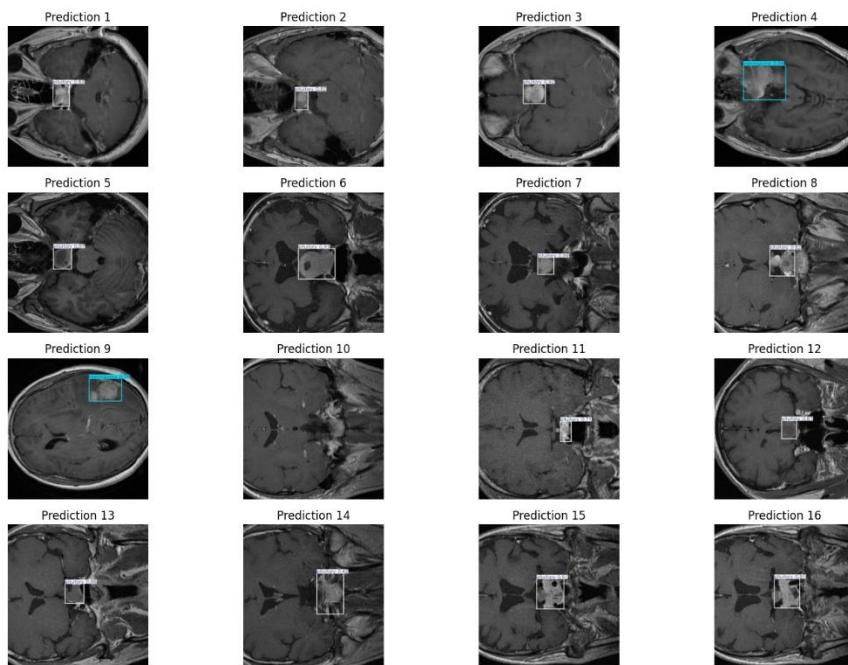
۳-۲ پایگاه داده

مانند دیتاست‌های قبلی در ابتدا دیتاست با آدرس زیر را دانلود کردیم و سپس استخراج، حذف فایل zip و نمایش پوشه‌بندی را انجام دادیم.^[7]

<https://www.kaggle.com/datasets/pkdarabi/medical-image-dataset-brain-tumor-detection/data>

۳-۳ آموزش

کدی نوشتیم که ابتدا کتابخانه Ultralytics را برای استفاده از مدل‌های YOLOv8 نصب می‌کند و سپس نسخه‌ی سبک آن یعنی YOLOv8n را که قبلاً روی دیتاست COCO آموزش دیده بارگذاری می‌کند. در مرحله بعد، مدل با استفاده از یک دیتاست سفارشی مربوط به تشخیص تومور مغزی که مسیر آن در فایل data.yaml مشخص شده، به مدت ۲۰ دوره و با اندازه تصویر ۳۲۰ × ۳۲۰ در آموزش داده می‌شود. در پایان، نتایج آموزش شامل مدل آموزش دیده و فایل‌های خروجی در پوشه‌ای به نام brain_tumor_yolov8 ذخیره می‌شوند. این کد پس از آموزش مدل YOLOv8، آن را روی تصاویر موجود در پوشه تست اجرا می‌کند. به‌طور دقیق‌تر، مدل آموزش دیده روی تمام تصاویر موجود در مسیر مشخص شده اعمال می‌شود تا اشیاء (تومورها) را شناسایی کند. تصاویر ورودی به اندازه ۳۲۰ × ۳۲۰ تغییر اندازه داده می‌شوند و آستانه اطمینان برای پیش‌بینی‌ها روی ۰/۲۵ تنظیم شده است، یعنی فقط پیش‌بینی‌هایی با confidence بیشتر از ۲۵٪ نگهداشته می‌شوند. گزینه save=True باعث می‌شود تصاویر خروجی همراه با جعبه‌های پیش‌بینی شده ذخیره شوند. این مرحله در واقع تست عملی مدل روی داده‌های دیده‌نشده است. خروجی بصورت زیر است:



تصویر ۳ خروجی YOLO

علاوه بر تصویر ۳، نتیجه YOLO را در قالب جدول ۵ می‌توانید بینید:

mAP@0.5:0.95	mAP@0.5	Recall	Precision	کلاس
0.555	0.830	0.741	0.821	glioma
0.833	0.976	0.948	0.944	meningioma
0.731	0.964	0.930	0.894	pituitary
0.706	0.923	0.873	0.886	all

جدول ۵ دقت YOLO

منابع و مراجع

- [1] <https://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/14753/8142>
- [2] <https://www.kaggle.com/datasets/mateuszbuda/lgg-mri-segmentation/data>
- [3] <https://arxiv.org/abs/1312.6114>
- [4] <https://www.kaggle.com/datasets/haonanzhou1/ixit2-slices>
- [5] <https://www.kaggle.com/datasets/awsaf49/brats20-dataset-training-validation>
- [6] <https://docs.ultralytics.com/>
- [7] <https://www.kaggle.com/datasets/pkdarabi/medical-image-dataset-brain-tumor-detection/data>

Abstract

Abstract

In this project, three deep learning models U-Net, VAE, and YOLO were applied to analyze MRI image data. The U-Net model was used for brain tumor region segmentation. The VAE was employed for compressing and reconstructing MRI images, as well as extracting latent features. The YOLO model was trained to detect and localize abnormal regions or lesions in the images. The results demonstrate that each model performs effectively in its designated task and proves useful in medical image analysis.

Code link:

<https://colab.research.google.com/drive/1lfqK1Xb28DPT9F2pbpXyzKU3D6g9zFOf?usp=sharing>

Key Words: U-Net, VAE, YOLO, Segmentation, Anomaly Detection



**Amirkabir University of Technology
(Tehran Polytechnic)**

Department of Mathematics and Computer science

Homework 2
U-Net, VAE and YOLO

By
Mohammadreza Shahrestani
Hossein Khamooshi

Supervisor
Dr. Saeed Sharifian

June 2025