دانشگاه آزاد اسلامی واحد تهران جنوب

پروژه درس بینایی ماشین

مقطع کارشناسی ارشد رشته مهندسی پزشکی گرایش بیوالکتریک

دکتر مهدی اسلامی

بخش‌بندی تصاویر سرطان سینه با استفاده از شبکه عصبی کانولوشنی

نگارنده:

نیما ولدبیگی

بهار ۱۴۰۳

فهرست مطالب

فهرست اشکال

فصل اول

فصل دوم

چکیده

سرطان سینه یکی از شایع ترین علل مرگ و میر زنان در سراسر جهان است. تشخیص زودهنگام به کاهش تعداد مرگ و میرهای زودهنگام کمک می‌کند. داده‌ها تصاویر پزشکی سرطان سینه را با استفاده از اسکن اولتراسوند بررسی می‌کند. مجموعه داده‌های سونوگرافی پستان به سه دسته تصاویر طبیعی، خوش خیم و بدخیم طبقه بندی می‌شوند. تصاویر اولتراسوند پستان می‌تواند نتایج بسیار خوبی در طبقه‌بندی، تشخیص و بخش‌بندی سرطان سینه در صورت ترکیب با یادگیری ماشینی ایجاد کند. هدف بخش‌بندی تصویر پزشکی شناسایی مناطق مهم یا مشکوک در تصاویر پزشکی است. با این حال، معمولاً هنگام توسعه شبکه‌هایی برای این نوع تحلیل، با چالش‌های زیادی مواجه می‌شویم. اولاً، حفظ وضوح تصویر اصلی برای این کار بسیار مهم است، زیرا شناسایی ویژگی‌ها یا ناهنجاری‌های ظریف می‌تواند به طور قابل‌توجهی بر دقت تشخیص تأثیر بگذارد. اخیراً، پذیرش گسترده مدل‌های مبتنی بر داده منجر به افزایش قابل توجهی در اکتشاف و پیشرفت سیستم‌های تشخیص به کمک رایانه (CAD) به کمک AI (هوش مصنوعی) شده است. رادیولوژیست‌ها، با استفاده از سیستم‌های CAD به کمک هوش مصنوعی، ممکن است بتوانند بینش‌های کامپیوتری را با تخصص خود ترکیب کنند و ارزیابی دقیق‌تر و سریع تر را امکان پذیر کنند. چنین سیستم‌های هوشمند اغلب با تکنیک‌های تصویربرداری زیست‌پزشکی، مانند اشعه ایکس، سی‌تی‌اسکن و نمونه‌های MRI کار می‌کنند. یکی از محبوب‌ترین شبکه‌های مورد استفاده در یادگیری عمیق شبکه عصبی کانولوشنی است.

فصل اول: سیستم تشخیص کامپیوتری سرطان سینه

۱-۱ مقدمه

یادگیری ماشین و یادگیری عمیق برای خودکار کردن عملیات تشخیص تومور مورد بررسی قرار گرفته است و چارچوب کلی یک سیستم تشخیص خودکار آورده شده است و هر کدام از مراحل خودکار کردن عملیات تشخیص توضیح داده شده است. در آخر انواع روش‌های ارزیابی عملکرد مدل‌ها به طور مفصل توضیح داده شده است. یکی از محبوب‌ترین شبکه‌های مورد استفاده در یادگیری عمیق شبکه عصبی کانولوشنی است. مهم‌ترین مزیت این نمونه از شبکه‌ها، استفاده از لایه کانولوشنی به منظور استخراج ویژگی‌های پیچیده است. از کاربردهای شبکه عصبی کانولوشنی عملیات دسته‌بندی و بخش‌بندی می‌باشد، که انتخاب لایه‌های مناسب، منجر به دقت مطلوب می‌شود.

۱-۲ شبکه‌های عصبی کانولوشنی

شبکه‌های عصبی کانولوشنی، نوعی شبکه عصبی برای پردازش اطلاعات هستند که ساختار شبکه شناخته شده‌ای دارند. نمونه‌هایی از این شبکه‌ها شامل داده‌های سری زمانی هستند که می‌توان آن‌ها را به عنوان یک شبکه یک بعدی در نظر گرفت که نمونه‌ها را در فواصل زمانی منظم بکار می‌برد. نمونه‌های دیگر شامل داده‌های تصویری هستند که می‌توان آن‌ها را به عنوان یک شبکه دو بعدی از پیکسل‌های تصویر به‌کار برد. شبکه‌های کانولوشنی در کاربردهای عملی بسیار موفق بوده‌اند. نام شبکه عصبی کانولوشنی نشان می‌دهد که این شبکه از یک عملیات ریاضی به نام کانولوشن استفاده می‌کند. کانولوشن یک عملگر خطی است، که از کانولوشن در مکان ضرب ماتریس‌ها در حداقل یکی از لایه‌ها استفاده می‌کنند.

شبکه‌های عصبی کانولوشنی نسبت به بقیه رویکردهای دیگر دسته‌بندی و بخش‌بندی به میزان کمتری از پیش‌پردازش استفاده می‌کنند. به این صورت که شبکه ویژگی‌هایی را استخراج می‌کند که در روش‌های یادگیری سنتی به صورت دستی ویژگی‌ها استخراج می‌شدند. پس شبکه‌های عصبی کانولوشنی نیازمند مرحله استخراج ویژگی نیستند و شبکه به صورت خودکار ویژگی‌های مربوط به مجموعه داده را استخراج می‌کند.

۱-۳ عملیات کانولوشن

لایه‌های شبکه عصبی سنتی از ضرب ماتریس در ماتریسی از پارامترها استفاده می‌کنند که پارامتر جداگانه‌ای تعامل بین هر واحد ورودی و هر واحد خروجی را توصیف می‌کند. به این صورت که هر واحد خروجی با هر واحد ورودی تعامل دارد. با وجود این، شبکه‌های کانولوشنی معمولاً تعاملات پراکنده دارند (به اتصال پراکنده یا وزن‌های پراکنده اشاره می‌کند). این کار با ایجاد کرنل[۱] کوچک‌تر از ورودی انجام می‌شود. به عنوان مثال، هنگام پردازش یک تصویر، تصویر ورودی ممکن است هزاران یا میلیون‌ها پیکسل داشته باشد، اما می‌توان ویژگی‌های کوچک و معنادار از قبیل لبه‌های تصویر را که تنها ده‌ها یا صدها نقطه اشغال می‌کنند، با استفاده از کرنل‌ها آشکار کرد. این به این معنی است که باید پارامترهای کمتری ذخیره شوند، که هر دو نیاز به حافظه را کاهش داده و کارایی آماری آن را بهبود می‌بخشد. همچنین به این معنی است که محاسبه خروجی به عملیات کمتری نیاز دارد، این بهبودها در کارایی شبکه‌ها بسیار مفید هستند.

۱-۳-۱ لایه ادغام

در یک لایه کانولوشن بعد از اعمال کانولوشن از تابع ادغام استفاده شده است، به منظور بهبود عملکرد شبکه هنگامی که ورودی از تابع ادغام عبور می‌کند اندازه ورودی مقدار قابل توجهی کوچک شده است. لایه ادغام از میان پیکسل‌های یک پنجره، یک پیکسل را به عنوان نماینده چندین پیکسل انتخاب می‌کند.

۱-۳-۲ توابع فعال ساز

شبکه عصبی کانولوشنی از توابع فعال‌ساز غیرخطی بعد از لایه کانولوشن استفاده می‌کند، استفاده از تابع غیرخطی باعث عمیق‌تر شدن شبکه شده است. در بین توابع فعال‌ساز غیرخطی، تابع ReLU از جایگاه ویژه‌ای برخوردار است، که از خانواده ReLU توابع فعال‌ساز دیگری هم وجود دارند. همچنین محاسبه‌های تابع ReLU ساده هستند، به همین خاطر محاسبه‌ها در بخش تابع غیرخطی نسبت به سایر توابع غیرخطی با سرعت بیشتری انجام شده است. پس سرعت فرایند آموزش با استفاده از تابع ReLU نسبت به توابع غیرخطی دیگر بیشتر است. همچنین از مشکلاتی که توابع غیرخطی دیگر همچون، تابع سیگموید[۲] و تابع Tanh دارند، اشباع شدن این نوع از توابع در

[۱] Kernel
[۲] Sigmoid

مقادیر خاص است. این توابع در مقادیر خیلی بزرگ و مقادیر خیلی کوچک به اشباع می‌رسند، و همین موضوع باعث شده گرادیان این توابع به سمت صفر میل کند، و بنابراین فرآیند بهینه‌سازی با سرعت کمتری انجام شود.

۱-۴ آموزش شبکه

معماری یک شبکه کانولوشنی مشابه الگوی اتصال نورون‌ها در مغز انسان است و از سازوکار بینایی[۱]الهام‌گرفته شده است. نورون‌های فردی فقط در یک منطقه محدود از میدان دیداری فرد که به نام میدان دریافتی[۲]شناخته می‌شود، به محرک‌ها پاسخ می‌دهند. مجموعه‌ای از این مناطق با هم، هم‌پوشانی دارند تا کل منطقه دیداری را پوشش دهند.

یادگیری عمیق نوعی خاص از یادگیری ماشین می‌باشد، برای درک بهتر یادگیری عمیق، ابتدا باید درک محکمی از اصول اولیه یادگیری ماشین وجود داشته باشد. الگوریتم یادگیری ماشین الگوریتمی است که قادر به یادگیری از طریق داده‌ها است. یادگیری عمیق یک زیرمجموعه از شاخه کلی‌تر هوش مصنوعی به نام یادگیری ماشین است. در یادگیری ماشین، به جای آموزش یک کامپیوتر با استفاده از یک لیست بزرگ از قوانین برای حل مشکل، شبکه‌ای ارائه شده است که در آن می‌توان نمونه‌ها را ارزیابی کرد، و یک مجموعه کوچک از دستورالعمل‌ها برای اصلاح شبکه در زمانی که یک اشتباه رخ داده را ایجاد کرد. انتظار می‌رود در طول زمان، یک شبکه مناسب بتواند مشکل را با دقت خوبی حل کند.

۱-۵ تابع اتلاف

برای حل مسأله یادگیری ماشین، یک متخصص داده باید راهی برای ایجاد عملکردی پیدا کند که کمترین میزان خطا میان عملکرد شبکه و مسأله دنیای واقعی در دسترس قرار گیرد. تابع اتلاف[۳]عملکردی است که با استفاده از آن پروژه‌های مربوط به علوم داده به ریاضیات تبدیل شده است. بیشتر عملکردهای مربوط به یادگیری ماشین، و مقدار زیادی از هوش مصنوعی به تعریف تابع اتلاف درست برای حل مسأله در دست منتهی شده است. رابطه ریاضی مربوط به تابع اتلاف به صورت رابطه

---

[۱] Visual Cortex
[۲] Receptive Field
[۳] Loss Function

$$L(x,y)=\sum_{i=1}^{N} L_i(x_i ,y_i)$$

تعریف می‌شود. در این رابطه L_i تابع اتلاف به ازای نمونه i ام، x_i ورودی i ام و y_i خروجی یا هدف i ام، L تابع اتلاف به ازای تمامی داده‌ها و N تعداد کل داده‌ها است. سپس به ازای تمامی داده‌ها از تابع اتلاف به صورت رابطه

$$\nabla L(x,y)= \sum_{i=1}^{N} \nabla L_i(x_i ,y_i$$

گرادیان گرفته می‌شود تا مقدار تابع اتلاف به کمترین حالت ممکن برسد. خطای آنتروپی متقاطع[1] یکی از توابعی است که به عنوان تابع اتلاف در شبکه‌های یادگیری عمیق زیاد استفاده می‌شود. این معیار نسبت به توابع خطای دیگر سرعت همگرایی بالاتری دارد. خطای آنتروپی متقاطع یک روش ریاضی برای اندازه‌گیری فاصله بین دو توزیع احتمال است، که به صورت رابطه

$$H(p,q)=y \log y_{pred}+(1-y) \log (1-y_{pred})$$

$$p=(y,1-y)$$

$$q=(y_{pred},1-y_{pred})$$

است. در این رابطه p توزیع داده‌های واقعی برای یک سیستم گسسته با دو پیامد و q توزیعی است، که با سیستم یادگیری ماشین پیش‌بینی شده است. تابع خطای آنتروپی متقاطع در سیستم‌های یادگیری ماشین به طور گسترده برای آموزش دسته‌بندی کننده‌ها استفاده شده است.

۱-۶ بهینه‌سازی شبکه‌های عمیق

الگوریتم‌های یادگیری عمیق شامل بهینه‌سازی در بسیاری از زمینه‌ها هستند. از بین همه‌ی بهینه‌سازی‌ها در یادگیری عمیق، دشوارترین و حساس‌ترین بهینه‌سازی مربوط به آموزش شبکه عصبی است. معمولاً بسیار رایج است که گاهی هزاران ماشین روزها و ماه‌ها برای حل یک نمونه از مشکل‌های آموزش‌های شبکه عصبی به کار گرفته شده است. از آنجا که حل این مشکل‌ها بسیار مهم و هزینه‌بر است، یک مجموعه تخصصی از تکنیک‌های بهینه‌سازی برای حل آن‌ها طراحی شده است.

[1] Cross-entropy loss

بسیاری از تکنیک‌های بهینه‌سازی، الگوریتم‌های کلی نیستند بلکه الگوهای کلی هستند که می‌توانند برای بازدهی بیشتر شبکه یا کابردهای خاص در الگوریتم‌های کلی به کار گرفته شوند. نرمال‌سازی دسته[1] یکی از جدیدترین نوآوری‌های اخیر در بهینه‌سازی شبکه‌های عصبی عمیق است. نرمال‌سازی دسته روشی است که به منظور تطبیق‌پذیری بیشتر پارامترها انجام می‌شود و در بسیاری از موارد به دلیل دشواری آموزش شبکه‌های عمیق انجام می‌شود. استفاده از روش نرمال‌سازی دسته باعث شده است که سرعت همگرایی بیشتر شود و همچنین حساسیت شبکه نسبت به پارامترها کمتر شود.

۱-۷ بیش‌برازش و کم‌برازش

هنگامی که خطای آموزشی شبکه زیاد باشد، حاکی از مشکل کم‌برازش است بدین صورت که شبکه به خوبی آموزش ندیده است. زمانی که دقت آموزش شبکه کم باشد، شبکه بر روی داده‌های تست هم نتایج مطلوبی نخواهد داشت. با زیاد کردن داده‌های آموزش یا استفاده از شبکه‌های دیگر به منظور آموزش شبکه می‌توان بر مشکل کم‌برازش غلبه کرد.

مشکل بیش‌برازش[2] به این حقیقت اشاره دارد که متناسب کردن یک شبکه برای یک مجموعه داده آموزشی خاص، تضمین نمی‌کند که عملکرد پیش‌بینی خوبی را بر روی داده‌های تست داشته باشد. حتی اگر این شبکه اهداف را بر روی داده‌های آموزشی کاملاً پیش‌بینی کند بدین صورت که شبکه با دقت بالایی آموزش ببیند. به عبارت دیگر همیشه شکاف بین آموزش و عملکرد داده آزمایشی وجود دارد، مشکل بیش‌برازش به خصوص زمانی که شبکه‌ها پیچیده هستند و مجموعه داده‌ها کوچک است وجود دارد.

مفهوم بیش‌برازش و کم‌برازش در یادگیری ماشین اغلب با استفاده از بده و بستان[3] بین واریانس و بایاس شناخته می‌شود. اگر داده‌های آموزشی به مجموعه‌ای متفاوت از داده‌ها تغییر پیدا کنند، به احتمال زیاد شبکه یک مجموعه کاملاً متفاوت از پارامترها را یاد می‌گیرد. این شبکه جدید احتمالاً یک پیش‌بینی کاملاً متفاوت در مقایسه با پیش‌بینی‌های مربوط به استفاده از اولین مجموعه داده آموزشی خواهد داشت. این نوع از تغییرات در پیش‌بینی یک نمونه آزمایشی با استفاده از مجموعه‌های داده آموزشی مختلف، تجلی واریانس شبکه است، که به خطای شبکه می‌افزاید. شبکه‌های پیچیده‌تر، این عیب را دارند که الگوهای نادرست را در نکات متفاوتی مشاهده کنند،

[1] Batch Normalization (BN)
[2] Overfitting
[3] Trade-off

بدین صورت که شبکه ویژگی‌هایی از داده‌ها را یاد می‌گیرد که منجر به خطای آموزش می‌شود، به خصوص زمانی که داده‌های آموزشی کافی نیستند.

یکی از راه‌حل‌ها برای غلبه بر مشکل بیش‌برازش زیاد کردن داده‌های آموزشی و راه‌حل دیگر کم کردن تعداد ویژگی‌هایی است که شبکه بر اساس آن ویژگی‌ها عملیات پیش‌بینی را انجام می‌دهد. راه‌حل دیگر برای غلبه بر مشکل بیش‌برازش کاهش پیچیدگی شبکه، یعنی کاهش تعداد لایه‌ها و یا تعداد نورون‌ها در هر لایه است. اما این راه‌حل بسیار زمان‌بر است، زیرا باید چندین معماری شبکه را امتحان کرد و همچنین رفتار خطای آموزشی و خطای تست برای هر شبکه مجزا مورد بررسی قرار گیرد. پس باید تا جای ممکن از شبکه‌های ساده که خطای کمی روی داده‌های آموزشی دارند استفاده کرد.

۱-۸ بخش‌بندی

بخش‌بندی تصویر یکی از مسائل مهم چالش برانگیز می‌باشد. به عبارت دیگر بخش‌بندی تصویر یکی از وظایف مهم در بینایی ماشین و کاربردهای پردازش تصویر می‌باشد. بخش‌بندی براین اساس است که تصویر به مناطق مختلف برای انجام عملیات پردازش بر روی تصویر آماده می‌شود. بخش‌بندی نقش اساسی در شناسایی وضعیت غیرطبیعی و برنامه‌ریزی جراحی برعهده دارد. بخش‌بندی تصویر برای جدا کردن بافت ناسالم از سالم مورد استفاده قرار می‌گیرد. در بعضی موارد بخش‌بندی تصویر با استفاده از شبکه‌های عصبی انجام می‌شود. روش دیگر بخش‌بندی تصویر روش آستانه‌گذاری است که بر اساس شدت روشنایی رنگ تصویر می‌توان قسمت‌های مختلف تصویر را از هم تفکیک کرد؛ که در این روش پیکسل‌های سیاه معرف پس‌زمینه و پیکسل‌های سفید معرف پیش‌زمینه تصویر می‌باشند.

چندین روش برای قسمت بخش‌بندی استفاده می‌شود که بسیاری از محققان از شبکه عصبی چندلایه استفاده کرده‌اند و یا یک ساختار مشابه به کارگرفته‌اند و الگوریتم یادگیری را حفظ کرده‌اند. روش‌های مختلف دیگری از جمله، شبکه عصبی پیش‌رونده پس‌انتشار[۱] شبکه عصبی پس‌انتشار[۲] پرسپترون چند لایه[۳] الگوریتم‌های مبتنی

[۱] Feedforward backpropagation neural network (FFBPNN)
[۲] Backpropagation neural network
[۳] Multilayer perceptron

برلبه[1] تکنیک‌های مبتنی برمنطقه[2] الگوریتم‌های دسته‌بند[3] نیز استفاده شده است، که در میان همه‌ی روش‌ها می‌توان بخش‌بندی تصویر را با استفاده از الگوریتم‌های یادگیری عمیق با دقت بهتری انجام داد.

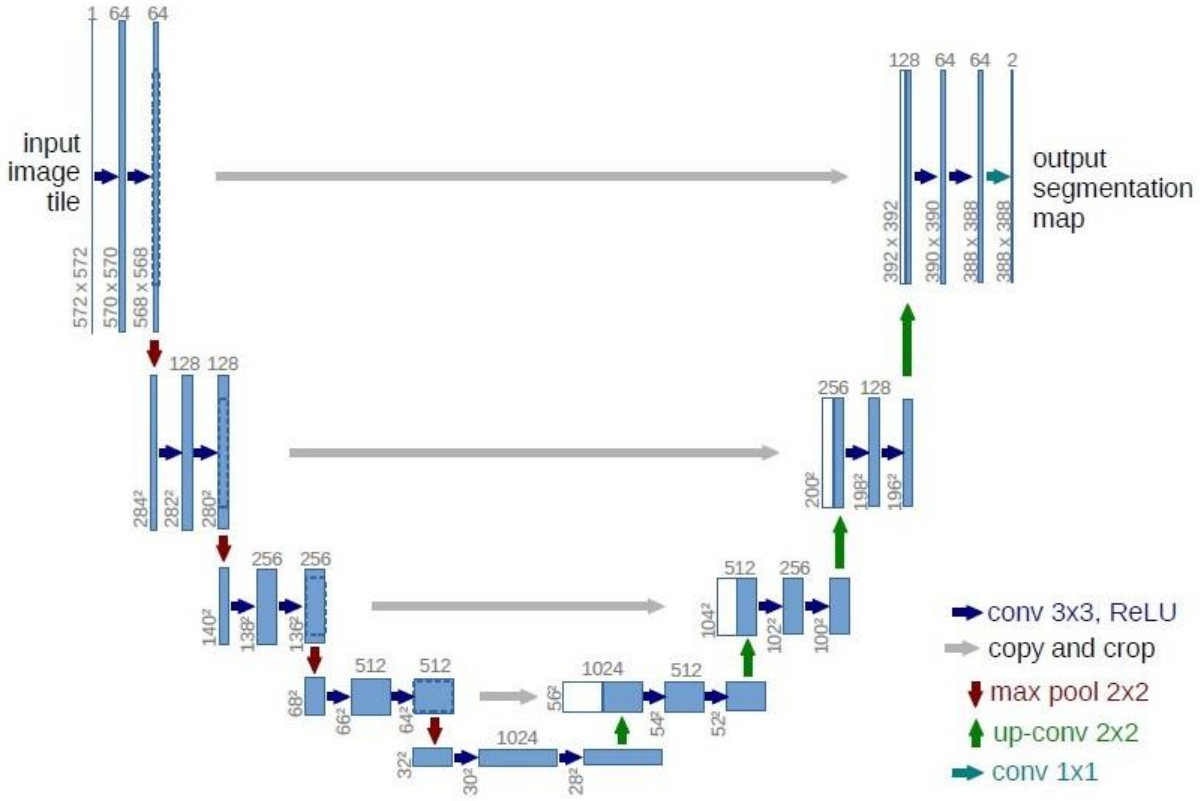
۱-۹ شبکه U-Net

استفاده معمول از شبکه‌های کانولوشن در کاربردهای دسته‌بندی است، که در آن خروجی یک تصویر دارای یک برچسب واحد است. اما در بسیاری از وظایف بصری[4] به خصوص در پردازش تصویر پزشکی، خروجی مطلوب باید شامل مکان‌یابی باشد، بدین صورت که یک برچسب کلاس باید به هر پیکسل اختصاص داده شود.

در شکل ۱-۱ شبکه U-Net نشان داده شده است. در این شکل هر بلوک آبی نمایان‌گر یک لایه کانولوشنی است، که به منظور استخراج ویژگی اعمال شده است. تعداد کانال‌های هر لایه در قسمت بالای هر بلوک نشان داده شده است. همچنین خروجی تصویر بعد از اعمال لایه کانولوشن در قسمت چپ هر بلوک نشان داده شده است. در شبکه U-Net لایه‌ها به صورت متقابل با هم اتصال[5] دارند. شبکه U-Net شامل دو مسیر است، مسیر فشرده‌سازی[6] (مسیر سمت چپ شکل (۱-۱)) و مسیر گسترده‌سازی[7](مسیر سمت راست شکل(۱-۱)) که ابتدا تصویر ورودی به مسیر فشرده‌سازی وارد شده است. این مسیر شامل کانولوشن‌های ۳×۳ همراه با تابع فعال‌ساز ReLU و لایه ادغام بیشینه ۲×۲ با گام ۲ است. مسیر گسترده‌سازی شامل کانولوشن بالا۲×۲[8] و کانولوشن ۳×۳ است. همچنین این شبکه از طریق چهار مسیر، لایه‌های مسیر فشرده‌سازی و مسیر گسترده‌سازی به هم وصل شده‌اند. اتصال‌های مسیرها به منظور جبران بخشی از ویژگی‌های از دست رفته تصاویر در عملیات فشرده‌سازی صورت گرفته است. در لایه آخر از یک لایه کانولوشن ۱×۱ به منظور دسته‌بندی هر پیکسل استفاده شده است. در این شبکه از ۲۳ لایه کانولوشن استفاده شده است. هدف از مسیر فشرده‌سازی استخراج ویژگی‌های مربوط به تصویر حاوی تومور است، و هدف از مسیر گسترده‌سازی کمک کردن به فرایند دقیق‌تر مکان‌یابی است.

---

[1] Edge-based algorithms
[2] Region-based techniques
[3] Clustering algorithms
[4] Visual tasks
[5] Concatenation
[6] Contracting path
[7] Expansive path
[8] Up convolution



شکل ۱- ۱ شبکه U-Net.

شبکه U-Net عملکرد بسیار خوبی در بخش‌بندی تصاویر مختلف پزشکی دارد، همچنین در بسیاری از موارد با استفاده از افزایش مصنوعی داده عملکرد این شبکه بهبود پیدا کرده است. پس در مواردی که مجموعه داده‌ها کم است می‌توان قبل از استفاده از شبکه U-Net داده‌ها را به طور مصنوعی افزایش داد، سپس عملیات آموزش شبکه را انجام داد.

۱-۱۰ ارزیابی عملکرد شبکه

برای محاسبه عملکرد سیستم‌های تشخیص از روش‌های مختلفی استفاده شده که هر کدام فرمول مختص خود را دارند. دراین بخش ابتدا نشانه‌های اختصاری مربوط به هر کدام از فرمول‌ها تعریف شده، سپس فرمول مربوطه آورده شده است.

صحیح مثبت[۱]: نمونه عضو دسته مثبت است و عضو همین دسته تشخیص داده شده است.

[۱] True Positives (TP)

صحیح منفی:[۱] نمونه عضو دسته منفی است و عضو دسته منفی تشخیص داده شده است.

اشتباه مثبت:[۲] نمونه عضو دسته منفی است و عضو دسته مثبت تشخیص داده شده است.

اشتباه منفی:[۳] نمونه عضو دسته مثبت است و عضو دسته منفی تشخیص داده شده است.

حساسیت:[۴] در واقع احتمال این است که یک آزمایش شناسایی مثبت باشد، با توجه به اینکه فرد مبتلا به بیماری است، و به نسبت موارد صحیح مثبت به کل داده‌ها گفته می‌شود و زمانی استفاده می‌شود که شناسایی موارد مثبت بسیار حیاتی است. به عنوان نمونه برای شناسایی یک بیماری کشنده و شناسایی موارد مثبت از این نمونه بیماری استفاده شده است. حساسیت به صورت رابطه

$$\text{Sensitivity(Se)} = \frac{TP}{TP+FN}$$

بیان شده است، همچنین حساسیت به اسم Recall هم شناخته شده است.

ویژه بودن:[۵] در واقع احتمال این است که یک آزمایش شناسایی منفی باشد، با توجه به اینکه فرد مبتلا به این بیماری نیست و به نسبت صحیح منفی به کل موارد منفی در داده‌ها گفته می‌شود. به منظور پوشش کلیه‌ی موارد منفی از این ارزیابی‌کننده استفاده می‌شود. ویژه بودن به صورت رابطه

$$\text{Specificity(Sp)} = \frac{TN}{TN+FP}$$

بیان شده است.

دقت:[۶] در واقع احتمال این است که یک آزمایش شناسایی به درستی انجام شود و نسبت پیش‌بینی‌های صحیح به کل پیش‌بینی‌های صورت گرفته است. و به صورت رابطه

$$\text{Accuracy (Ac)} = \frac{TP+TN}{TN+FP+TP+FN}$$

بیان شده است.

---

[۱] True Negative (TN)
[۲] False Positives (FP)
[۳] False Negative (FN)
[۴] Sensitivity (Se)
[۵] Specificity (Sp)
[۶] Accuracy (Ac)

ضریب تشابه[1] به صورت رابطه

$$Dsc = \frac{2\times TP}{(FP+TP)+(TP+FN)}$$

تعریف شده است. همچنین می‌توان Dsc را به صورت رابطه

$$Dsc(A,B) = \frac{2\|A\cap B\|}{\|A\|+\|B\|}$$

تعریف کرد.

رابطه (۲-۵) برای تعریف Dsc کلی‌تر است، A و B دو ماسک برای بخش بندی هستند، و ∩ عملیات اشتراک بر روی A و B است. $\|A\|$ نرم A است، که در تصاویر منظور پیکسل‌های تصاویر است.

اشتراک روی اجتماع[2] به صورت رابطه

$$IOU = \frac{TP}{TP+FP+FN}$$

تعریف شده است همچنین می‌توان IOU را به صورت رابطه

$$IOU(A,B) = \frac{\|A\cap B\|}{\|A\cup B\|}$$

تعریف کرد.

صحت[3] به نسبت موارد صحیح مثبت به کل موارد مثبت پیش‌بینی شده صحت گفته می‌شود. به صورت رابطه

$$Precision = \frac{TP}{TP+FP}$$

تعریف می‌شود. از کاربردهای این ارزیابی کننده اطمینان از موارد صحیح مثبت پیش‌بینی شده است. به منظور شناسایی موارد اشتباه منفی استفاده می‌شود.

[1] Dice similarity coefficient (Dsc)

[2] Intersection over Union (IOU)

[3] Precision

فصل دوم: سیستم بخش‌بندی خودکار تصاویر سرطان سینه با استفاده از شبکه عصبی کانولوشنی.

۲-۱ مقدمه

بخش‌بندی و دسته‌بندی خودکار سرطان سینه نقش مهمی در تشخیص، تحلیل و تفسیر بهتر تصاویر مربوط به سرطان ایفا می‌کند. بخش‌بندی سرطان سینهع ملیاتی است که به تفکیک بافتی که تومور در آن ناحیه قرار دارد نسبت به بافت سالم مغز می‌پردازد. همچنین عملیات بخش‌بندی باعث می‌شود جراح قبل از عمل بتواند تحلیل دقیقی از محل و مرز ناحیه تومور داشته باشد. دسته‌بندی عملیاتی است که سیستم به صورت خودکار نوع تومور را تشخیص می‌دهد و به دلیل اینکه نوع سرطان سینه در روند درمان اهمیت بالایی دارد، پس عملیات دسته‌بندی از اهمیت ویژه‌ای برخوردار است. مهم‌ترین عامل برای استفاده از سیستم‌های تشخیص خودکار سرطان سینه به جای استفاده از تشخیص‌های سنتی، دقت بالای این سیستم‌ها است، پس دقت شبکه‌های طراحی شده از اهمیت بالایی برخوردار است. در این فصل با استفاده از شبکه کاهش‌یافته U-Net عملیات بخش‌بندی انجام شده است سپس به منظور عملکرد بهتر این شبکه بر روی مجموعه داده‌های تست شبکه‌ی بهبود یافته U-Net پیشنهاد شده است.

۲-۲ ابزارهای تحقیق

معمولاً داده‌های پزشکی حجم بالایی دارند، به همین دلیل استفاده از ابزار مناسب برای آموزش شبکه‌های یادگیری عمیق بسیار مهم است. در صورتی که از پردازنده ضعیف برای آموزش شبکه استفاده شود ممکن است، آموزش شبکه ساعت‌ها و حتی چندین روز طول بکشد. پس انتخاب پردازنده با توجه به حجم داده و نوع شبکه مهم است.

۲-۲-۱ گوگل کولب

سرویس گوگل کولب[۱] یک سخت افزار رایگان برای انجام پروژه‌های مربوط به پایتون[۲] است. با استفاده از کولب می‌توان به صورت آنلاین و در مرورگر کدهای پایتون نوشته و اجرا شود. در کولب امکان پردازش با استفاده از

[۱] Google Colaboratory
[۲] python

GPU و TPU[1]فراهم است که معمولاً نسبت به سیستم‌های معمولی سرعت بیشتری دارد. از ویژگی‌های گوگل کولب امکان فراخوانی فایل‌ها از گوگل درایو[2]یا گیت هاب[3]یا فراخوانی مجموعه داده‌ها از طریق لینک مربوط به مجموعه داده موجود در سایت است، همچنین ویژگی دیگر گوگل کولب امکان استفاده از کتابخانه‌هایی مانند کراس[4]و تنسورفلو[5]است.

که ما در این پروژه با استفاده از قطعه کد زیر ابتدا گوگل کولب و گوگل درایو را به هم متصل میکنیم.

```
from google.colab import drive
drive.mount('/content/drive')
```

۲-۲-۳ کتابخانه کراس و تنسورفلو

کراس رابط برنامه‌نویسی کاربردی[6]برای یادگیری عمیق است که در پایتون نوشته شده است و بر روی پلتفرم[7] تنسورفلو یادگیری ماشین اجرا می‌شود. این کتابخانه با تمرکز بر امکان آزمایش‌های سریع توسعه داده شده است و به کاربر این امکان را می‌دهد که در انجام تحقیق‌ها هر چه سریع‌تر به نتیجه مطلوب دست پیدا کند.

کراس سطح بالای API از تنسورفلو ۲ است، همچنین یک رابط کاربری بسیار مفید برای حل مشکلات یادگیری ماشین با تمرکز بر یادگیری عمیق مدرن است. این سیستم چکیده‌های اساسی و بلوک‌های سازنده برای توسعه راهکارهای یادگیری ماشین با سرعت تکرار بالا فراهم می‌کند. کراس به مهندسان و محققان قدرت می‌دهد تا از قابلیت‌های تنسورفلو ۲ و پلتفرم‌های مشابه استفاده کنند. همچنین می‌توان کراس را بر روی TPU یا در خوشه‌های بزرگ GPUs اجرا کرد و می‌توان شبکه‌های کراس را برای اجرا در مرورگر و یا بر روی یک دستگاه سیار راه‌اندازی کرد.

سپس با استفاده از کتاب خونه‌های زیر را فرآخوانی می شوند:

```
import os

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[1] Tensor Processing Unit (TPU)
[2] Google Drive
[3] Github
[4] Keras
[5] Tensorflow
[6] Application Programming Interface (API)
[7] Platform

```
import h5py
import cv2
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import random
import glob
import keras
import random
import tensorflow as tf
from keras.layers import *
import keras.backend as k
from keras.models import *
from keras.optimizers import *
from skimage.transform import resize
from skimage.io import imread, imshow, imsave
from keras.losses import categorical_crossentropy
from keras.callbacks import ModelCheckpoint, LearningRateScheduler,
EarlyStopping
from skimage.metrics import peak_signal_noise_ratio
from skimage.restoration import (denoise_wavelet , estimate_sigma)
from sklearn.model_selection import StratifiedKFold
from tensorflow.keras.metrics import Recall, Precision
from sklearn.metrics import confusion_matrix
import gc
from tqdm import tqdm
from PIL import Image

from sklearn.model_selection import train_test_split

from tensorflow.keras.utils import plot_model
from tensorflow.keras.preprocessing.image import img_to_array, load_img
```

۳-۲ مجموعه داده

داده‌های جمع‌آوری‌شده در ابتدا شامل تصاویر سونوگرافی پستان در بین زنان ۲۵ تا ۷۵ ساله است. این داده‌ها در سال ۲۰۱۸ جمع آوری شده است. تعداد بیماران ۶۰۰ بیمار زن است. مجموعه داده شامل ۷۸۰ تصویر با اندازه

تصویر متوسط ۵۰۰ * ۵۰۰ پیکسل است. تصاویر با فرمت PNG هستند. تصاویر پیش زمینه (ماسک)[۱]با تصاویر اصلی ارائه شده است. تصاویر به سه دسته طبیعی[۲]، خوش خیم[۳]و بدخیم[۴]دسته بندی می‌شوند.

داده‌ها تصاویر پزشکی سرطان سینه را با استفاده از اسکن اولتراسوند بررسی می‌کند. مجموعه داده‌های سونوگرافی پستان به سه دسته طبیعی، خوش خیم و بدخیم طبقه بندی می‌شود. تصاویر اولتراسوند پستان می‌تواند نتایج بسیار خوبی در طبقه بندی، تشخیص و تقسیم بندی سرطان سینه در صورت ترکیب با استفاده از روش‌های هوش مصنوعی ایجاد کند.

این مجموعه داده به اسم Breast Ultrasound Images Dataset یا Dataset_BUSI_with_GT است که از لینک زیر قابل دسترسی است:

https://www.kaggle.com/datasets/aryashah2k/breast-ultrasound-images-dataset/data

GT==Ground Truth، در پایین هر تصویر موجود است.

داده ها با استفاده از کد زیر فرآخوانی می‌شوند:

```
Data_dir = "/content/drive/MyDrive/Dataset_BUSI_with_GT"
folders = os.listdir(Data_dir)
folders
```

['malignant', 'benign', 'normal']

```
image_paths, labels, mask_paths = [], [], []

for folder in folders:
    files = os.listdir(os.path.join(Data_dir, folder))
    for file in files:
        if "mask_" in file:
            continue
```

[۱] Ground truth
[۲] Normal
[۳] Benign
[۴] Malignant

```
        elif "mask" in file:
            mask_paths.append(os.path.join(Data_dir, folder, file))
        else:
            image_paths.append(os.path.join(Data_dir, folder, file))
            labels.append(folder)
```

مشخصات تعداد داده‌های هر دسته به صورت زیر است:

```
df["label"].value_counts()
```

label

benign 437

malignant 210

normal 133

Name: count, dtype: int64

۴-۲ عملیات پیش‌پردازش بر روی داده‌ها

سپس عملیات پیش‌پردازش با استفاده از قطعه کد زیر انجام می‌شود:

که شامل،

- ✓ اندازه تصاویر و ماسک ها را به اندازه دلخواه تغییر دهید.
- ✓ تبدیل تصاویر به فرمت RGB (اگر قبلا نبوده‌اند).
- ✓ مقادیر پیکسل تصاویر و ماسک‌ها را نرمال کنید.
- ✓ تصاویر و ماسک ها را به آرایه‌های numpy تبدیل کنید.

است.

```
def preprocess_images_and_masks(folder_path):

  # Sort the list of filenames
  filenames = sorted(os.listdir(folder_path))

  # Load the images and masks from the folder
  # Taking out multiple masks for one image because I am a coward
```

```
  images = [cv2.imread(os.path.join(folder_path, f)) for f in filenames if
f.endswith('.png') and not (f.endswith('_mask.png') or
f.endswith('_mask_1.png') or f.endswith('_mask_2.png'))]
  masks = [cv2.imread(os.path.join(folder_path, f), cv2.IMREAD_GRAYSCALE)
for f in filenames if f.endswith('_mask.png')]

  # Resize the images and masks to a desired size
  desired_size = (256, 256)
  images = [cv2.resize(image, desired_size) for image in images]
  masks = [cv2.resize(mask, desired_size, interpolation=cv2.INTER_NEAREST)
for mask in masks]

  # Convert the images to RGB format (if they are not already)
  images = [cv2.cvtColor(image, cv2.COLOR_BGR2RGB) for image in images]

  # Normalize the pixel values of the images and masks
  images = [image / 255.0 for image in images]
  masks = [mask / 255.0 for mask in masks]

  # Convert the images and masks to numpy arrays
  images = np.array(images)
  masks = np.array(masks)

  return images, masks
```

سپس داده‌ها با استفاده از مسیر تعریف شده در درایو هر فایل (دسته) خوانده می‌شوند. و ماسک و هر تصویر در یک لیست اختصاصی با حفظ ترتیب ذخیره می‌شوند.

```
# Define the paths to the three folders
benign_folder = '/content/drive/MyDrive/Dataset_BUSI_with_GT/benign'
malignant_folder = '/content/drive/MyDrive/Dataset_BUSI_with_GT/malignant'
normal_folder = '/content/drive/MyDrive/Dataset_BUSI_with_GT/normal'

# Preprocess the images and masks from the folders
benign_images, benign_masks = preprocess_images_and_masks(benign_folder)
malignant_images, malignant_masks =
preprocess_images_and_masks(malignant_folder)
normal_images, normal_masks = preprocess_images_and_masks(normal_folder)

# Concatenate the images and masks from the three folders
images = np.concatenate((benign_images, malignant_images, normal_images),
axis=0)
```

```
masks = np.concatenate((benign_masks, malignant_masks, normal_masks),
axis=0)
```

سپس ابعاد لیست مربوط به تصاویر و ماسک چک می‌شوند و همچنین ترتیب و جایگاه داده ها در لیست تغییر میکند که در آموزش مدل بیش برازش صورت نگیرد.

```
# Check the dimensions of the images and masks arrays
if len(images) == 0 or len(masks) == 0:
  print("Error: Images or masks array is empty")
elif len(images) != len(masks):
  print("Error: Mismatch in number of images and masks")
else:
  # Shuffle the images and masks together
  combined = list(zip(images, masks))
  np.random.shuffle(combined)
  images, masks = zip(*combined)

  # Convert the images and masks to numpy arrays
  images = np.array(images)
  masks = np.array(masks)
```

با استفاده از کد زیر ۹۰ درصد داده ها به عنوان داده‌های آموزشی، ۷٫۵ درصد به عنوان داده اعتبارسنجی و ۲٫۵ درصد به عنوان داده تست انتخاب می‌شوند.

```
  # Calculate the number of images in each split
  num_images = len(images)
  num_train = int(0.90 * num_images)
  num_val = int(0.075 * num_images)
  num_test = num_images - num_train - num_val

  # Split the images and masks into training, validation, and test sets
  train_images = images[:num_train]
  train_masks = masks[:num_train]
  val_images = images[num_train:num_train+num_val]
  val_masks = masks[num_train:num_train+num_val]
  test_images = images[num_train+num_val:]
  test_masks = masks[num_train+num_val:]
```

سپس ابعاد و داده های آموزشی و اعتبار سنجی و تست به صورت زیر مشخص می‌شوند.

آموزشی:

```
print(np.shape(train_images))
print(np.shape(train_masks))
```

(702, 256, 256, 3)

(702, 256, 256)

اعتبار سنجی:

```
print(np.shape(val_images))
print(np.shape(val_masks))
```

(58, 256, 256, 3)

(58, 256, 256)

تست:

```
print(np.shape(test_images))
print(np.shape(test_masks))
```

(20, 256, 256, 3)

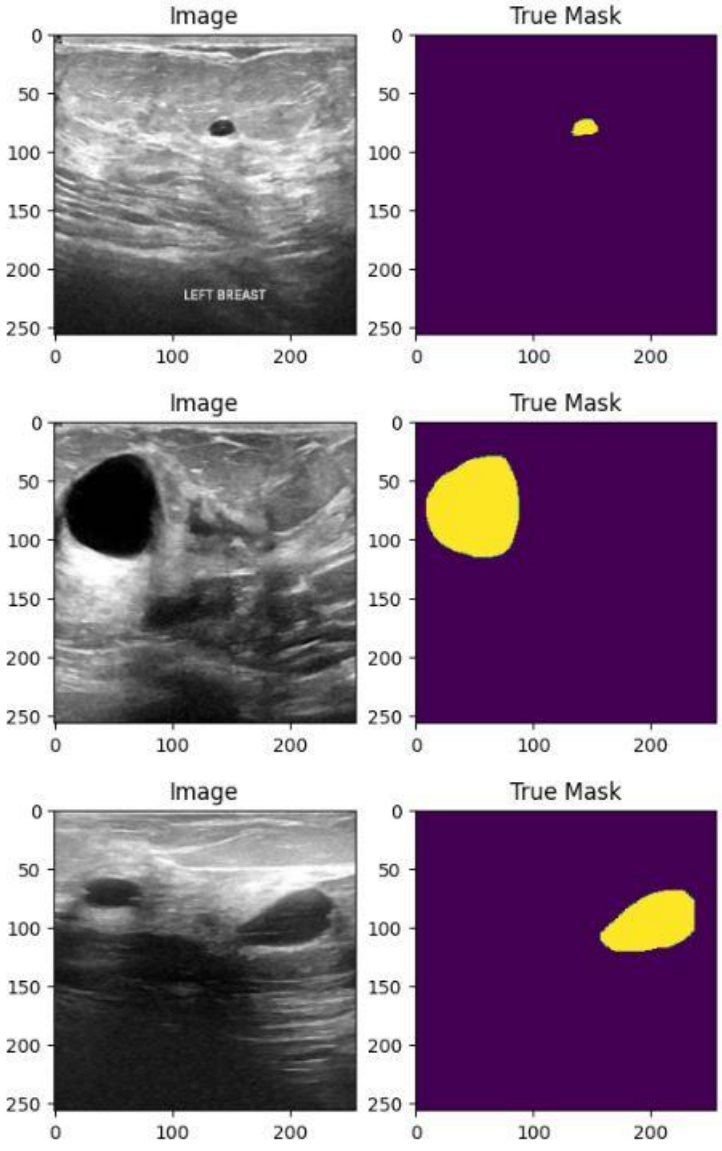
(20, 256, 256)

۵-۲ نمایش داده‌های مربوط به هر دسته

نمایش داده‌های دسته خوش خیم:

```
for i in range(3):
    image = benign_images[i]
    mask = benign_masks[i]

    # Display the image and the true mask
    fig, (ax1, ax2) = plt.subplots(1, 2)
    ax1.imshow(image)
    ax1.set_title("Image")
```

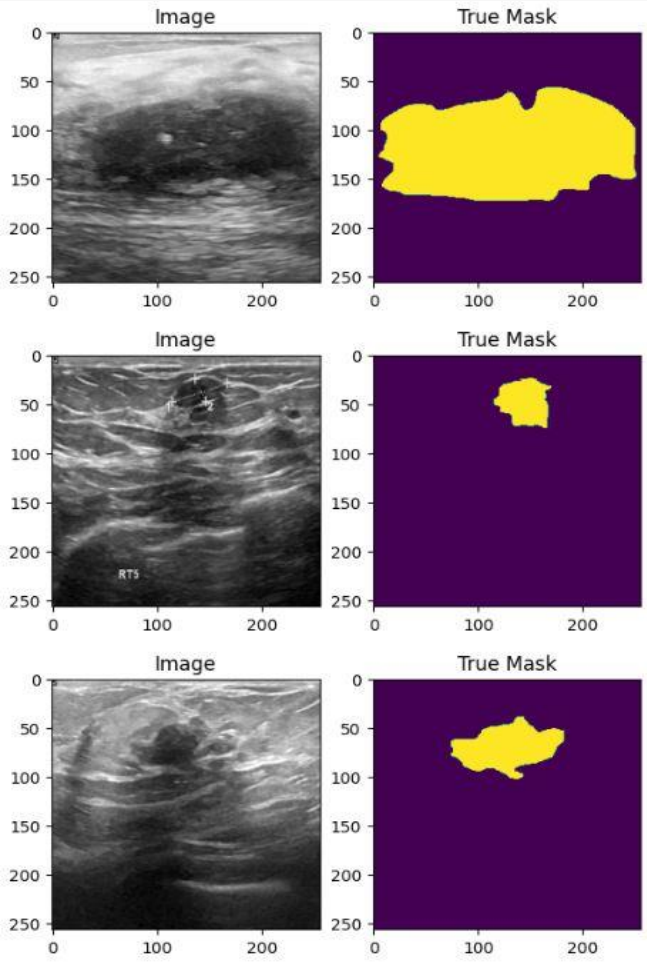
```
    ax2.imshow(mask)
    ax2.set_title("True Mask")

    plt.show()
```



شکل ۲- ۱ تصاویر سرطان سینه در دسته خوش خیم به همراه ماسک مربوطه.

نمایش داده‌های دسته بد خیم:

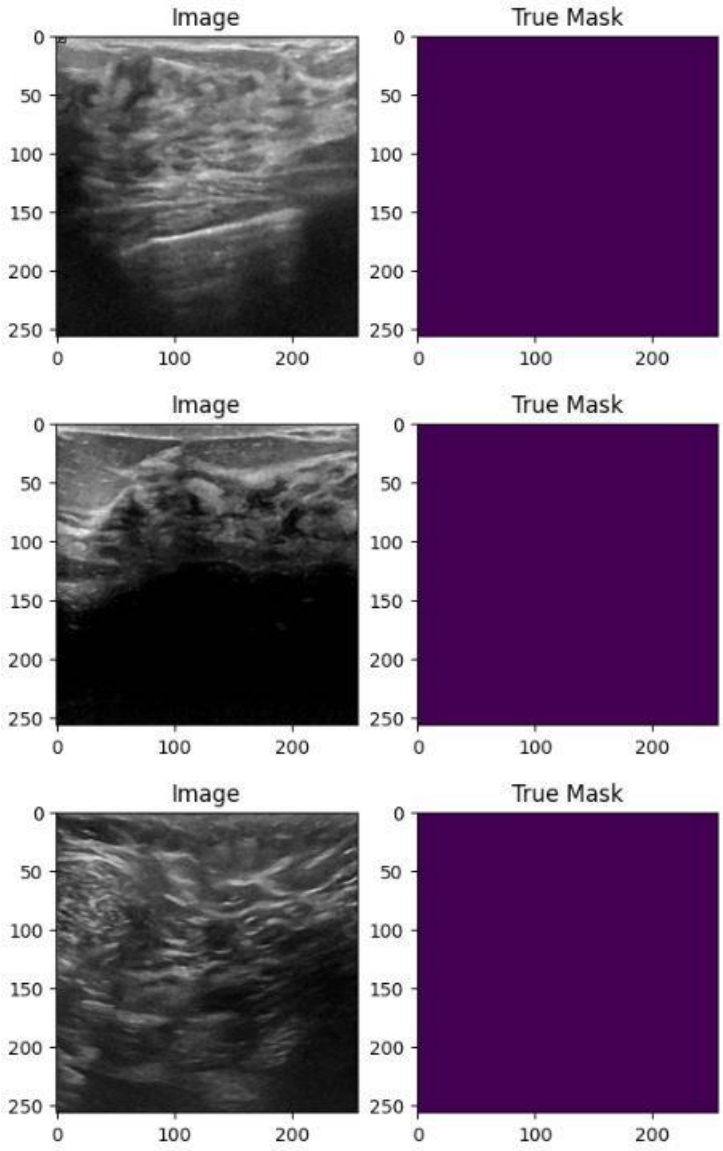
```
for i in range(3):
    image = malignant_images[i]
    mask = malignant_masks[i]

    # Display the image and the true mask
    fig, (ax1, ax2) = plt.subplots(1, 2)
    ax1.imshow(image)
    ax1.set_title("Image")
    ax2.imshow(mask)
    ax2.set_title("True Mask")

    plt.show()
```



شکل ۲- ۲ تصاویر سرطان سینه در دسته بد خیم به همراه ماسک مربوطه.

نمایش داده‌های دسته سالم:

```
for i in range(3):
    image = normal_images[i]
    mask = normal_masks[i]

    # Display the image and the true mask
    fig, (ax1, ax2) = plt.subplots(1, 2)
    ax1.imshow(image)
    ax1.set_title("Image")
    ax2.imshow(mask)
    ax2.set_title("True Mask")

    plt.show()
```



شکل ۲- ۳ تصاویر سرطان سینه در دسته سالم به همراه ماسک مربوطه.

۲-۶ ساخت مدل

استفاده معمول از شبکه‌های کانولوشن در کاربردهای دسته‌بندی است، که در آن خروجی یک تصویر دارای یک برچسب واحد است. اما در بسیاری از وظایف بصری[۱] به خصوص در پردازش تصویر پزشکی، خروجی مطلوب باید شامل مکان‌یابی باشد، بدین صورت که یک برچسب کلاس باید به هر پیکسل اختصاص داده شود. در سال‌های اخیر یادگیری عمیق پیشرفت بزرگی در بخش‌بندی تصاویر پزشکی ایجاد کرده است. در این راستا، U-Net

[۱] Visual tasks

محبوب‌ترین معماری در بخش‌بندی تصاویر پزشکی بوده است. شبکه U-Net عملکرد کلی برجسته در بخش‌بندی تصاویر پزشکی از طریق آزمایش‌های گسترده بر روی مجموعه داده‌های چالش برانگیز داشته است. شبکه‌های بهبودیافته U-Net در کاربردهای خاص علاوه بر مزایای شبکه U-Net ازمزیت‌های بهبود ایجاد شده در شبکه متناسب با کاربرد خاص بهره می‌برد. بنابراین برخی اصلاحات برای بهبود مدل U-Net در مقالات مختلف پیشنهاد شده است.

که با استفاده از کد زیر مدل ساخته می‌شود.

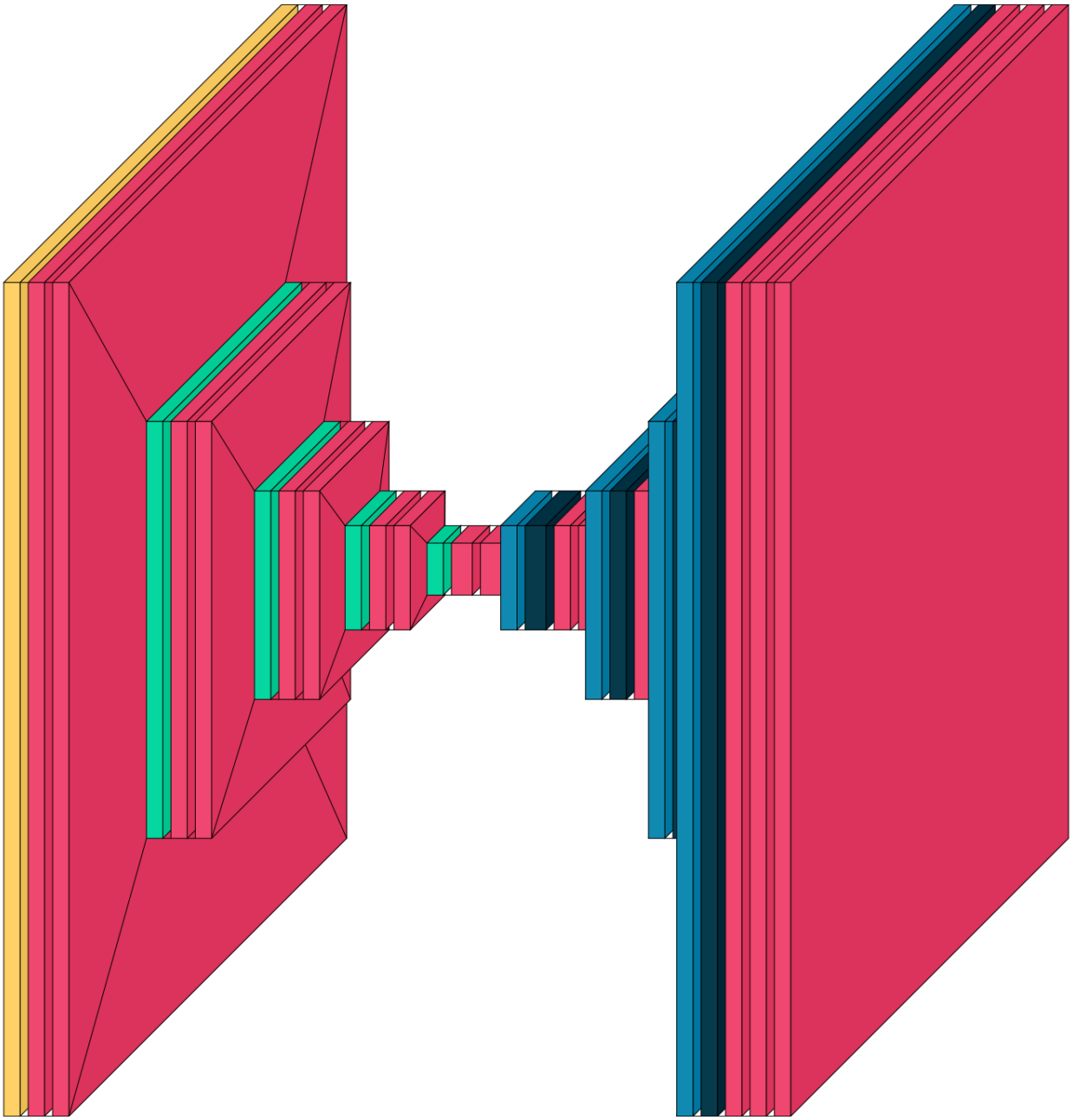
```
ini_input=keras.Input(shape=(256,256,3),name="image")
#n = Lambda(lambda x: x/127.5 - 1.)(ini_input)


c1 = Conv2D(16, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(ini_input)
#c1 = layers.Dropout(0.1)(c1)
c1 = Conv2D(16, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c1)
p1 = MaxPooling2D((2,2))(c1)


c2 = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(p1)
#c2 = layers.Dropout(0.1)(c2)
c2 = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c2)
p2 = MaxPooling2D((2,2))(c2)


c3 = Conv2D(64, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(p2)
#c3 = layers.Dropout(0.1)(c3)
c3 = Conv2D(64, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c3)
p3 = MaxPooling2D((2,2))(c3)


c4 = Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(p3)
#c4 = layers.Dropout(0.1)(c4)
c4 = Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c4)
p4 = MaxPooling2D((2,2))(c4)
```

```
c5 = Conv2D(256, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(p4)
#c5 = layers.Dropout(0.1)(c5)
c5 = Conv2D(256, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c5)



u6 = Conv2DTranspose(128, (2,2), strides=(2,2), padding='same')(c5)
u6 = concatenate([u6, c4])
c6 = Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(u6)
#c6 = layers.Dropout(0.1)(c6)
c6 = Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c6)


u7 = Conv2DTranspose(64, (2,2), strides=(2,2), padding='same')(c6)
u7 = concatenate([u7, c3])
c7 = Conv2D(64, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(u7)
#c7 = layers.Dropout(0.1)(c7)
c7 = Conv2D(64, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c7)

u8 = Conv2DTranspose(32, (2,2), strides=(2,2), padding='same')(c7)
u8 = concatenate([u8, c2])
c8 = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(u8)
#c8 = layers.Dropout(0.1)(c8)
c8 = Conv2D(32, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c8)


u9 = Conv2DTranspose(16, (2,2), strides=(2,2), padding='same')(c8)
u9 = concatenate([u9, c1], axis = 3)
c9 = Conv2D(16, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(u9)
#c9 = layers.Dropout(0.1)(c9)
c9 = Conv2D(16, (3,3), activation='relu', kernel_initializer='he_normal',
                padding='same')(c9)

outputs = Conv2D(1,(1,1), activation='sigmoid')(c9)
```

```
model1 = Model(inputs = [ini_input], outputs = [outputs])
```

که در خروجی زیر مقادیر مربوط به ورودی و خروجی بعد از هر لایه نمایش داده شده است.

```
import visualkeras
visualkeras.layered_view(model1, legend=True) # without custom font
from PIL import ImageFont
#font = ImageFont.truetype("arial.ttf", 12)
visualkeras.layered_view(model1) # selected font
```



شکل ۲- ۴نمایش شبکه.

۲-۷ ارزیابی عملکرد مدل

مدل با استفاده از توابع تعریف شده در زیر و همچنین توابع تعریف شده در کتاب خانه ها ارزیابی می‌شوند.

```
def dice_loss(y_true, y_pred):
    # Flatten the predictions and ground truth
```

```
    y_true_flat = tf.reshape(y_true, [-1])
    y_pred_flat = tf.reshape(y_pred, [-1])

    # Compute the intersection and union
    intersection = tf.reduce_sum(y_true_flat * y_pred_flat)
    union = tf.reduce_sum(y_true_flat) + tf.reduce_sum(y_pred_flat)

    # Compute the Dice loss
    dice_loss = 1 - 2 * intersection / union

    return dice_loss



from keras import backend as K

def f1_score(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    recall = true_positives / (possible_positives + K.epsilon())
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val
```

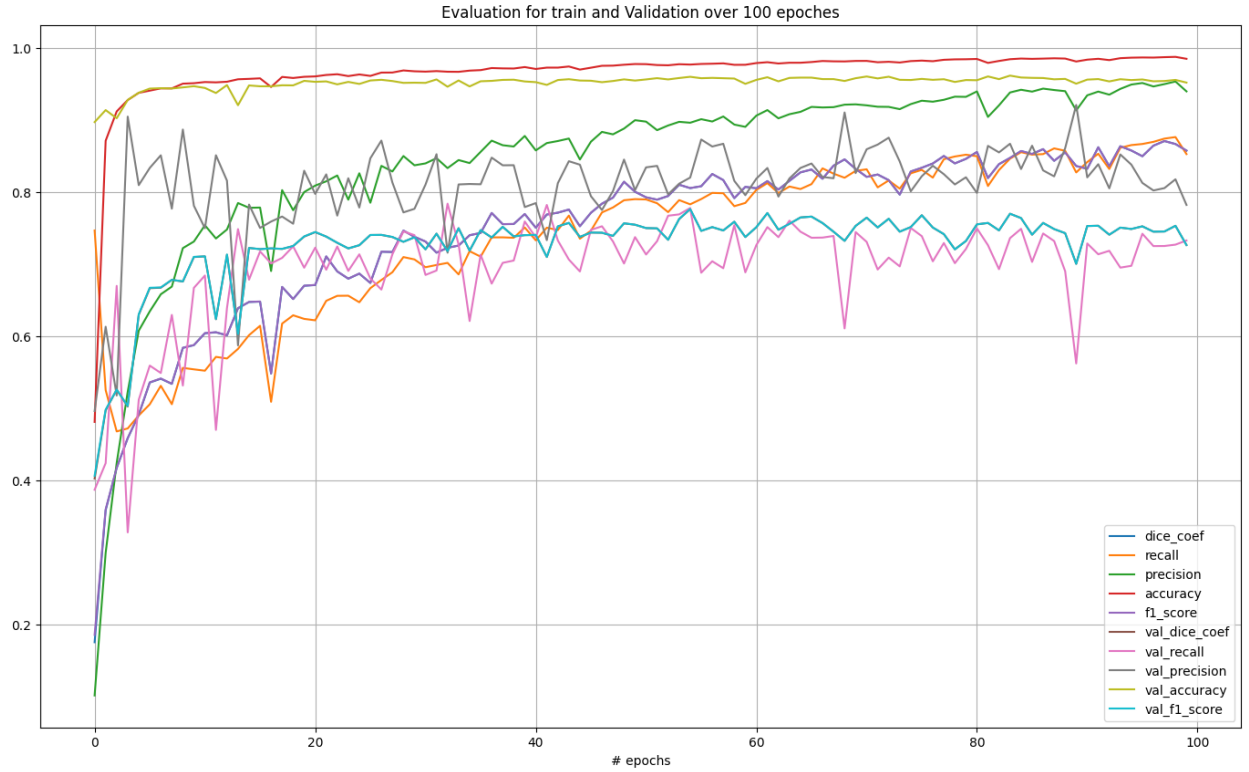
سپس مدل با استفاده از تابع اتلاف و توابع بهینه ساز آماده سازی می‌شوند.

```
model1.compile(optimizer=Adam(learning_rate=1e-4), loss = dice_loss,
metrics = [dice_coef, Recall(), Precision(), 'accuracy', f1_score ])

history1 = model1.fit(train_images, train_masks, batch_size=2, epochs=100,
verbose=1, validation_data=(val_images, val_masks))
```

ارزیابی عملکرد مدل بعد از ۱۰۰ ایپاک به صورت زیر است:

```
keys = history1.history.keys()
l = []
plt.figure(figsize=(17, 10))
for key in keys:
    if key[-1] != 's':
        plt.plot(history1.history[key])
        l.append(key)
plt.grid()
plt.legend(l)
```

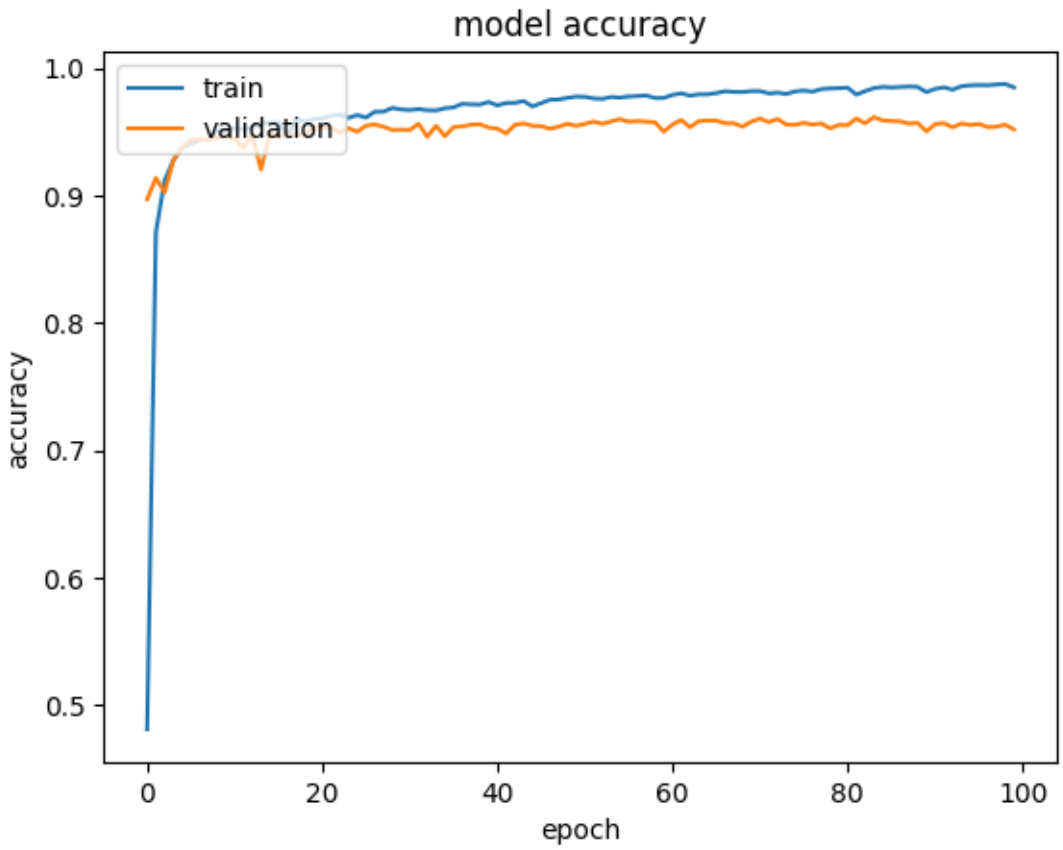
```
plt.title('Evaluation for train and Validation over 100 epoches')
plt.xlabel('# epochs');
```



شکل ۲- ۵ ارزیابی عملکرد مدل بعد از ۱۰۰ ایپاک.

نمودار دقت مدل:

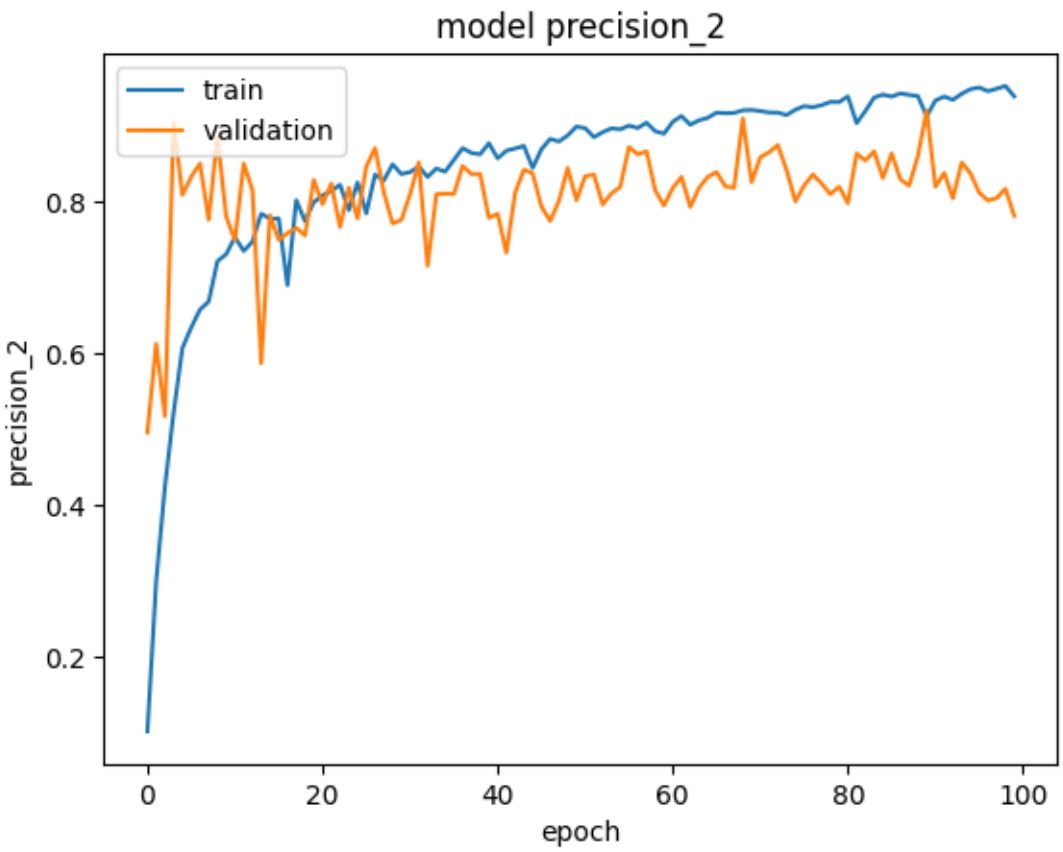
```
#  "accuracy"
plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



شکل ۲- ۶ نمودار دقت مدل.

نمودار صحت مدل:

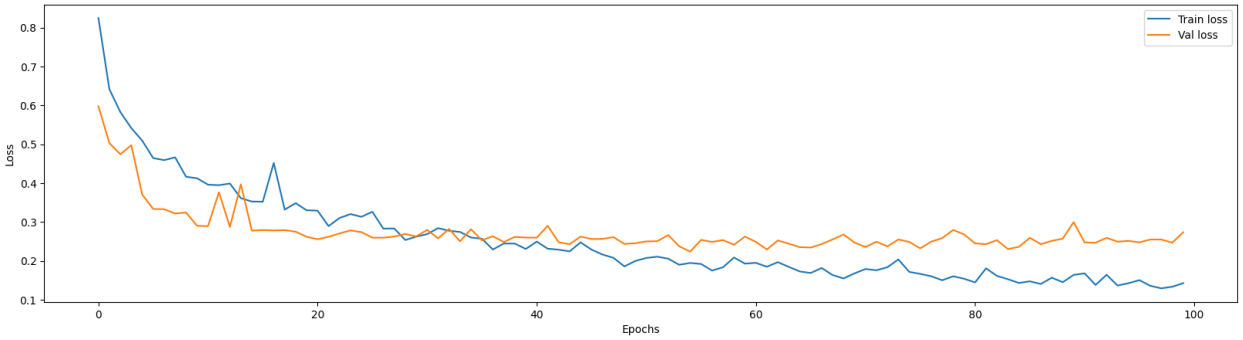
```
#  "precision"
plt.plot(history1.history['precision'])
plt.plot(history1.history['val_precision'])
plt.title('model precision_2')
plt.ylabel('precision_2')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```



شکل ۲- ۷ نمودار صحت مدل.

نمودار تابع اتلاف:

```
plt.figure(figsize=(20, 5))
plt.plot(model1.history.history['loss'], label='Train loss')
plt.plot(model1.history.history['val_loss'], label='Val loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```



شکل ۲- ۸ نمودار تابع اتلاف.

۲-۸ ارزیابی مدل بر روی داده‌های تست

ارزیابی دقت داده‌های تست به صورت زیر است:

```
score = model1.evaluate(test_images, test_masks, verbose = 1)
```

Loss: 0.2826

Dice_coef: 0.7174

Recall: 0.7536

Precision: 0.6847

Accuracy: 0.9662

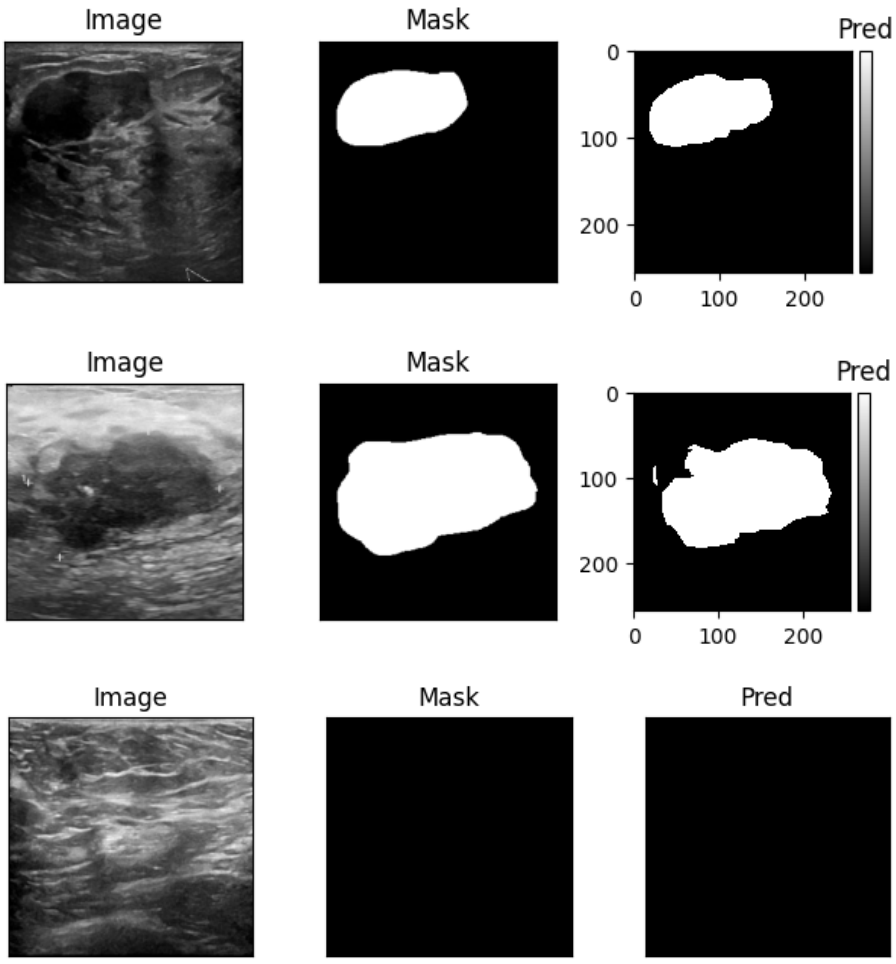
F1_score: 0.7175

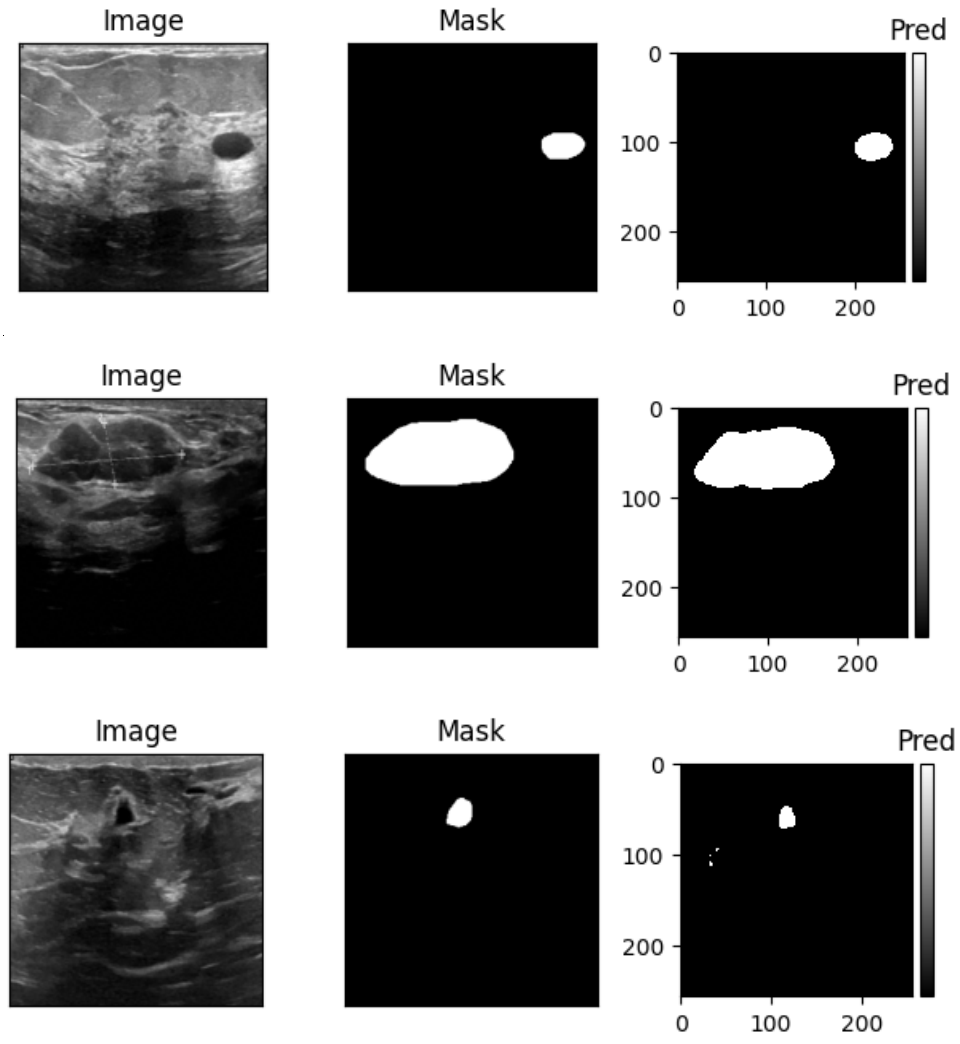
همچنین چند نمونه از داده‌های تست به همراه ماسک اصلی و ماسک پیش‌بینی شده در شکل زیر نمایش داده شده است.

```
preds_test = model1.predict(test_images, verbose=1)
preds_test_t = (preds_test>(0.5)).astype(np.uint8)
```

```
for i in range(20):
  plt.subplot(1,3,1)
  r = i
  plt.imshow(test_images[r].reshape(256,256,3),cmap='gray')
  plt.xticks([])
  plt.yticks([])
```

```
plt.title("Image")

plt.subplot(1,3,2)
plt.imshow(test_masks[r].reshape(256,256,1),cmap='gray')
plt.xticks([])
plt.yticks([])
plt.title("Mask")

plt.subplot(1,3,3)
imshow((preds_test_t[r]).reshape(256,256,1), cmap = 'gray')
plt.xticks([])
plt.yticks([])
plt.title("Pred")

plt.figure()
```





شکل ۲- ۹ نمونه داده‌های تست به همراه ماسک پیش‌بینی شده توسط مدل.