دانشگاه تهران
علوم کامپیوتر

تشخیص و تقسیم‌بندی خودکار پولیپ در تصاویر دستگاه گوارشی: یک رویکرد یادگیری عمیق

استاد محترم :
دکتر هدیه ساجدی

دانشجو:
امیرعلی امینی

پاییز ۱۴۰۲

۱. مقدمه

۱.۱ زمینه

سرطان روده‌ای یکی از شایع‌ترین انواع سرطان در سراسر جهان است با نرخ مرگ و میر قابل‌توجه. شناسایی و تشخیص زودرس سرطان روده و لکه‌های پیش‌کشنده آن مانند پولیپ‌ها برای درمان موثر و بهبود نتایج بیمار، بسیار مهم است. روش‌های سنتی تشخیص پولیپ بر اساس بررسی تصویری توسط متخصصین گوارشی با استفاده از کولونوسکوپی بوده است. با این حال، این فرآیند ممکن است زمان‌بر و به خطاهای انسانی حساس باشد.

۱.۲ انگیزه

پیشرفت‌ها در تصویربرداری پزشکی و یادگیری عمیق، راه‌های جدیدی را برای تشخیص و تقسیم‌بندی خودکار پولیپ‌ها در تصاویر اندام گوارشی ارائه داده است. تکنیک‌های تقسیم‌بندی تصویر، به ویژه آنهایی که بر پایه شبکه‌های عصبی هستند، پتانسیل تشخیص دقیق و کارآمد پولیپ‌ها از تصاویر اندام گوارشی را دارند. با اتوماسیون این فرآیند، می‌توانیم کارآیی برنامه‌های اسکرینینگ سرطان روده را بهبود بخشیم، مداخله زودرس را فراهم کنیم و در نهایت جان‌ها را نجات دهیم.

۱.۳ اهداف

هدف اصلی این پروژه توسعه یک مدل یادگیری عمیق برای تشخیص و تقسیم‌بندی خودکار پولیپ‌ها در تصویر اندام گوارشی است. به طور خاص، ما به دنبال این موضوعات هستیم:

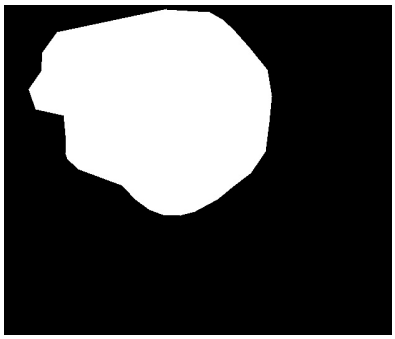
- کاوش و پیش‌پردازش مجموعه داده Kvasir-SEG، که یک مجموعه داده عمومی حاوی تصاویر از پولیپ‌ها است.
- پیاده‌سازی و ارزیابی انواع مختلف مدل‌های یادگیری عمیق برای تقسیم‌بندی تصویر، با تمرکز بر ساختارهای مناسب برای تشخیص پولیپ.
- آموزش و بهینه‌سازی مدل انتخابی بر روی مجموعه داده Kvasir-SEG، با بهینه‌سازی عملکرد آن از طریق آزمایش با هایپرپارامترها و تکنیک‌های افزایش داده.
- ارزیابی عملکرد مدل آموزش دیده با استفاده از معیارهای استاندارد مانند دقت، دقت، بازیابی و امتیاز F۱.
- مقایسه عملکرد مدل با آنچه در مقاله موجود است و ارزیابی کارآیی بالینی آن برای تشخیص زودرس سرطان روده.

۲. شرح مجموعه داده

۲.۱ مجموعه داده Kvasir–SEG

مجموعه داده Kvasir-SEG یک مجموعه داده عمومی است که شامل ۱۰۰۰ تصویر از پولیپ‌ها، که یک پیش‌کشنده معمول به سرطان روده است، میباشد. هر تصویر در مجموعه داده همراه با یک ماسک متناظر است که مناطق پولیپ را مشخص می‌کند. این مجموعه داده در اندازه و فرمت تصویر متفاوت است، با تصاویر ارائه شده به صورت فرمت JPEG و ماسک‌ها به صورت فرمت JPEG هستند..

تصاویری از دیتا ست یا مجموعه داده:



۲.۲ کاوش دیتاست

قبل از توسعه مدل، ما بررسی جامعی از دیتاست Kvasir-SEG انجام دادیم. این شامل تحلیل ساخت تصاویر و ماسک‌ها، درک توزیع پولیپ‌ها در سراسر مجموعه داده و بررسی هر گونه متادیتا (meta data) که با مجموعه داده ارائه شده است، بود. با کسب درک از مجموعه داده، شروع به پیش پردازش داده ها کردم.

۳. پیش‌پردازش

۳.۱ پیش‌پردازش تصویر

پیش‌پردازش تصویر یک مرحله حیاتی در آماده‌سازی داده برای آموزش مدل است. این شامل تکنیک‌هایی مانند تغییر اندازه، نرمال‌سازی و افزودن داده به منظور اطمینان از یکنواختی و بهبود عملکرد مدل است. در مورد مجموعه داده Kvasir-SEG، تصاویر را به اندازه ای یکنواخت تغییر دادیم، نرمال‌سازی را برای استانداردسازی مقادیر پیکسل اعمال کردیم و داده را از طریق تکنیک‌هایی مانند چرخش، برگرداندن و مقیاس‌بندی افزایش داده‌ها افزایش دادیم.

```python
# Resizing the images to 256, 256 and then afdding them to the above matrix
print("Resizing training images and masks")
# Iterate through each image in the dataset
for n, id_ in tqdm(enumerate(images_ids), total=len(images_ids)):

    # Construct the path to the current image
    path = imagesPath + "/" + id_

    # Read the image using imread function, keeping only the RGB channels
    img = imread(path)[:, :, :3]

    # Resize the image to the desired size (256x256) while preserving the range of pixel values
    img = resize(img, (256, 256), mode="constant", preserve_range=True)

    # Assign the resized image to the appropriate index in the 'X' array
    X[n] = img
    X[n] = img

    # Initialize a mask array filled with zeros of shape (256, 256, 1)
    mask = np.zeros((256, 256, 1), dtype=np.bool_)

    # Read the corresponding mask image using imread function
    mask = imread(masksPath + "/" + id_)

    # Apply any necessary preprocessing to the mask image (e.g., converting to grayscale)
    mask = rtg(mask)

    # Resize the mask image to the desired size (256x256) while preserving the range of pixel values
    mask = np.expand_dims(resize(mask, (256, 256), mode="constant", preserve_range=True), axis=-1)
    # Assign the resized mask to the appropriate index in the 'Y' array
    Y[n] = mask
```

[62]

```
Resizing training images and masks
100%|██████████| 1000/1000 [00:59<00:00, 16.90it/s]
```

۳.۲ مدیریت تصاویر با وضوح متفاوت

یک چالشی که توسط مجموعه داده Kvasir-SEG ارائه شده است، وجود تصاویر با وضوح متفاوت است. برای رفع این چالش، ما تکنیک‌هایی را برای مدیریت تصاویر با وضوح متفاوت اجرا کردیم، مانند تغییر اندازه تصاویر به یک اندازه استاندارد با حفظ نسبت ابعاد و اعمال پدینگ یا برش در صورت نیاز. با اطمینان از یکنواختی در وضوح تصویر، ما انحراف‌های ممکن را کمینه کردیم و توانایی مدل در تعمیم در اندازه‌های مختلف تصاویر را بهبود دادیم.

۴. انتخاب و پیاده‌سازی مدل

۴.۱ کاوش درباره مدل‌های یادگیری عمیق (deep learning)

ما تحقیقات گسترده‌ای در مورد مدل‌های دیپ لرنینگ مناسب برای تقسیم‌بندی تصویر انجام دادیم، با تمرکز بر روی ساختارهایی که در وظایف تصویربرداری پزشکی مؤثر ثابت شده‌اند. مدل‌هایی مانند U-Net، Mask R-CNN و نسخه‌های آنها برای قابلیت‌های دقیق در تصویربرداری پزشکی مورد بررسی قرار گرفتند. همچنین در کلاس توسط استاد بر استفاده از شبکه U-Net توصیه شده بود و مقاله مرتبط نیز از این مدل استفاده کرده است.

۴.۲ انتخاب ساختار U-Net

بعد از ارزیابی مدل‌های مختلف، ما تصمیم گرفتیم که ساختار U-Net را پیاده‌سازی کنیم به دلیل کارآیی آن در وظایف تقسیم‌بندی تصویر پزشکی و سادگی طراحی آن، نظر استاد و مقاله مرتبط. ساختار U-Net شامل یک مسیر کوچک‌کننده برای ساخت فیچر مپ (feature map) و یک دیکدر(decoder) متقارن برای سگمنت کردن دقیق است. ما ساختار U-Net را به عنوان نقطه شروعی برای توسعه مدل انتخاب کردیم، با تغییرات بر اساس مقاله "شبکه‌های Double Encoder-Decoder برای تقسیم‌بندی پولیپ‌های گوارشی" اتخاب کردیم.

۴.۳ جزئیات پیاده‌سازی

مدل U-Net با استفاده از TensorFlow و Keras، دو کتابخانه یادگیری عمیق محبوب، پیاده‌سازی شد. ساختار مدل شامل لایه‌های کانولوشن برای استخراج ویژگی، به دنبال آن لایه‌های decoder برای سگمنتیشن دقیق بود. در این پیاده سازی از شبکه های پری ترین (pre train) استفاده کردم که به فرایند سرعت ببخشم.

تصویر لایه های مورد استفاده در انتها گزارش آمده است.

۵. آموزش و اعتبارسنجی

۵.۱ روش آموزش

مدل بر روی مجموعه داده پیش‌پردازش‌شده Kvasir-SEG با استفاده از ترکیبی از داده‌های آموزش و اعتبارسنجی آموزش داده شد. ما از روش‌های استانداردی مانند نزول گرادیان کوچک-دسته و پس‌انتشار به منظور بهینه‌سازی پارامترهای مدل استفاده کردیم. هایپرپارامترهایی مانند نرخ یادگیری، اندازه دسته و تعداد اپوک‌ها از طریق آزمایش بهینه‌سازی شدند تا عملکرد بهینه را به دست آوریم.

۵.۲ معیارهای ارزیابی

برای ارزیابی عملکرد مدل، ما از معیارهای استانداردی مانند accuracy، recall وF۱ score استفاده کردیم. علاوه بر این، ما معیارهای ویژه‌تری مانند(IoU) Intersection over Union و Dice Coefficient استفاده کردیم تا توانایی مدل در تعیین دقیق مناطق پولیپ در تصاویر را ارزیابی کنیم.

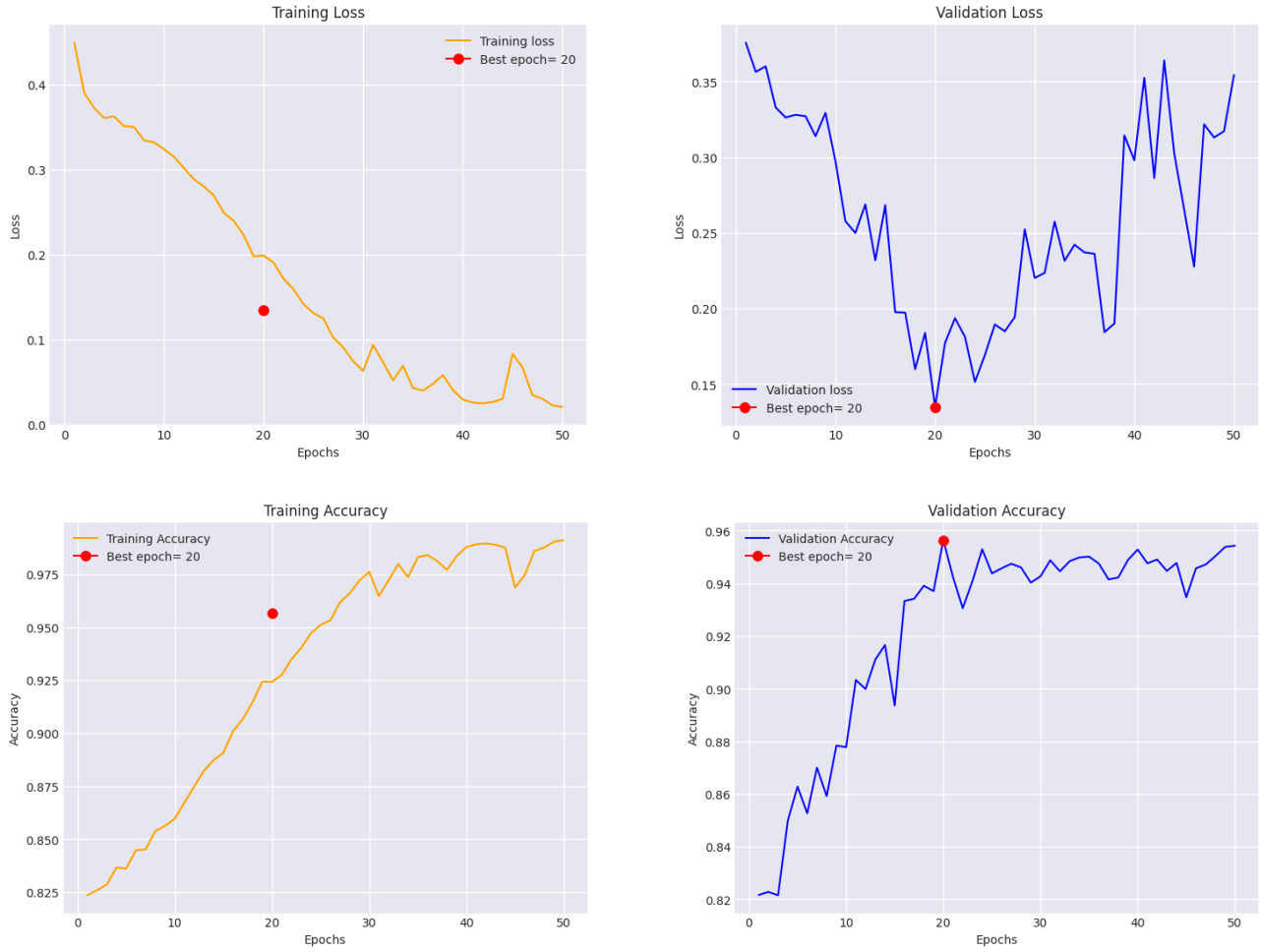
۵.۳ اعتبارسنجی مدل(Model Validation) .

مدل آموزش داده شده با استفاده از یک مجموعه داده اعتبارسنجی(validation data) جداگانه اعتبارسنجی شد تا عملکرد تعمیمی آن را ارزیابی کنیم. نسبت دیتا ترین به ولیدیشن را ۹۹ به یک قرار دادم تا از داده بیشتری برای ترین استفاده شود.

۶. نتایج و تحلیل

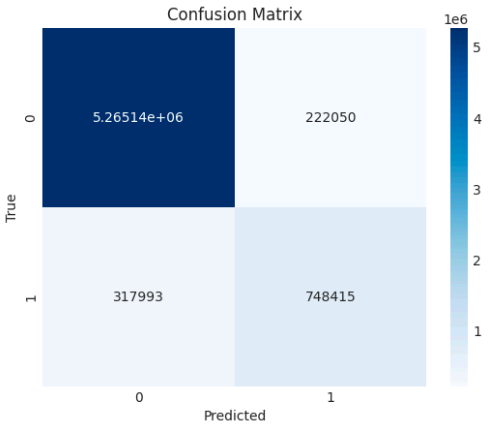
۶.۱ معیارهای عملکرد

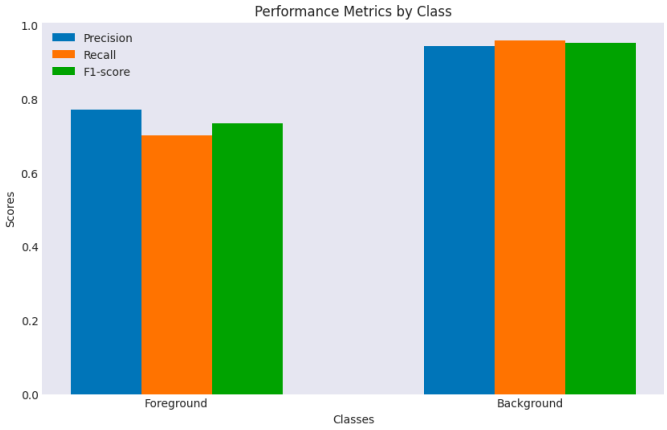
عملکرد مدل آموزش دیده با استفاده از ترکیبی از معیارهای کمی و ارزیابی‌های کیفی تحلیل شد. ما معیارهـای عملکردی مـانـند accuracy، recall و F۱ score بـرای هـر دو کلاس پیش زمینه (پولیپ) و زمینه محاسبه کردیم. علاوه بر این، ما پیش‌بینی‌های مدل را در کنار ماسک های حقیقی مشاهده کردیم تا به دقت و محدودیت‌های آن دست یابیم.

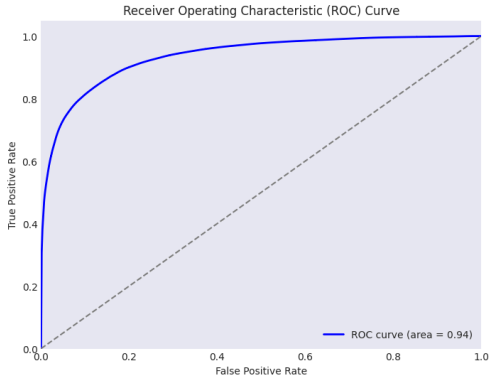


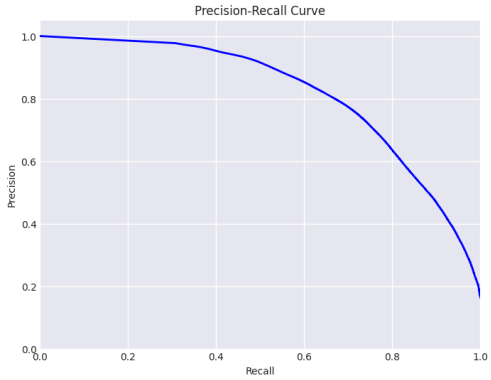
نمودار های اطلاعات ترین شبکه

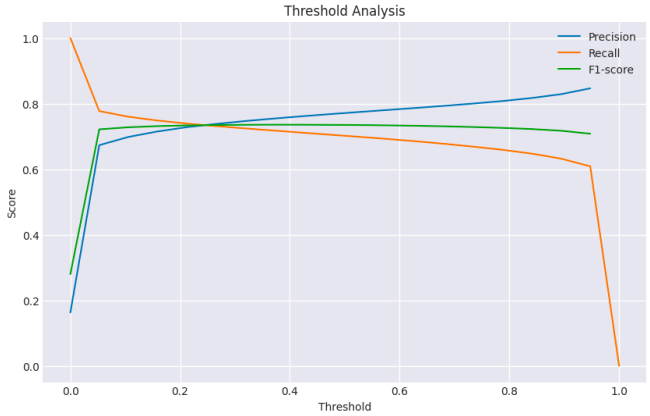
در این شبکه به دقت ۹۹ درصد برای دیتا ترین و دقت ۹۵ درصد برای دیتا ولیدیشن و دقت ۹۸ درصد برای کل دیتا رسیدم.

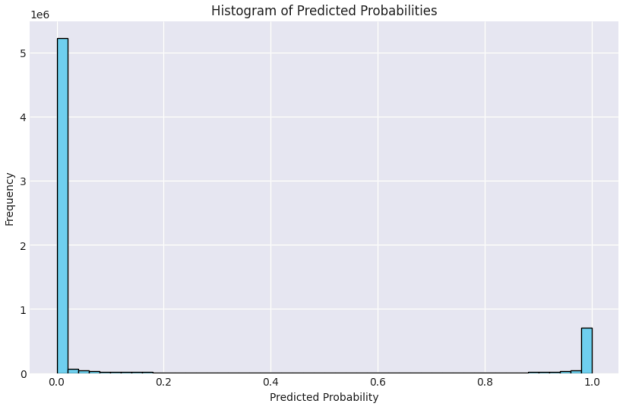












- True Positives (TP) = 748415
- False Positives (FP) = 222050
- False Negatives (FN) = 317993

Recall = TP / (TP + FN) = 748415 / (748415 + 317993) ≈ 0.701

Precision = TP / (TP + FP) = 748415 / (748415 + 222050) ≈ 0.771
F1 Score = 2 * (Precision * Recall) / (Precision + Recall) ≈ 0.735

- **ماتریس سردرگمی:** تحلیل ماتریس سردرگمی نشان می‌دهد که سیستم به طور کلی در تشخیص کلاس‌ها عملکرد قابل قبولی داشته است. مقادیر بالای روی قطر ماتریس، بیانگر میزان موفقیت سیستم در پیش‌بینی درست کلاس‌ها است. با این وجود، مشاهده می‌شود که برای برخی کلاس‌ها، به ویژه کلاس ۰، اشتباه‌هایی نیز رخ داده است.
- **عملکرد بر اساس کلاس:** جدول عملکرد، مقادیر دقت، فراخوانی و امتیاز F۱ را برای هر کلاس به طور جداگانه ارائه می‌دهد. این معیارها، جزئیات بیشتری در مورد عملکرد سیستم برای هر کلاس خاص در اختیار ما قرار می‌دهند.
- **منحنی‌های ROC و دقت-فراخوانی:** این منحنی‌ها به ترتیب، به بررسی رابطه بین نرخ مثبت واقعی و نرخ مثبت کاذب، و رابطه بین دقت و فراخوانی با تغییر آستانه طبقه‌بندی می‌پردازند. هر دوی این منحنی‌ها برای درک عملکرد کلی سیستم مفید هستند. در این مورد خاص، سطح زیر منحنی برای ROC، مقدار ۰.۹۴ را نشان می‌دهد که بیانگر عملکرد خوب سیستم است.
- **تحلیل آستانه:** نمودار تحلیل آستانه، تغییرات دقت، فراخوانی و امتیاز F۱ را با تغییر آستانه نمایش می‌دهد. این اطلاعات می‌توانند به انتخاب بهترین آستانه برای دستیابی به عملکرد مطلوب کمک کنند.
- **هیستوگرام احتمالات پیش‌بینی شده:** این هیستوگرام توزیع احتمالات پیش‌بینی شده برای کلاس مثبت را نشان می‌دهد. این نمودار به ما کمک می‌کند تا میزان اطمینان سیستم نسبت به پیش‌بینی‌های خود را درک کنیم.

۶.۲ مقایسه با مقاله موجود

ما عملکرد مدل خود را با آنچه در مقاله گزارش شده است، به خصوص مقاله "شبکه‌های Double Encoder-Decoder برای تقسیم‌بندی پولیپ‌های گوارشی" مقایسه کردیم. با مقایسه نتایج ما با روش‌های برجسته، ما قابلیت کارایی رویکرد خود را ارزیابی کردیم و حوزه‌هایی برای بهبود مشخص کردیم.

۶.۳ چالش‌ها و محدودیت‌ها

در طول پروژه، ما با چالش‌ها و محدودیت‌های مختلفی روبرو شدیم، از جمله عدم توازن داده، بیش‌برازش و محدودیت‌های محاسباتی. ما این چالش‌ها را از طریق تکنیک‌هایی مانند تنظیم وزن کلاس، رژولاریزاسیون و ساده‌سازی مدل مدیریت کردیم.

۷. مستندات و گزارش

۷.۱ پیاده‌سازی کد

تمام پروژه با استفاده از Jupyter مستندسازی شده است. با توضیحات دقیق برای هر بلاک کد و مرحله پیاده‌سازی. کد به بخش‌های منطقی تقسیم شده و با استفاده نوشتن توضیحات در بخش های مارک داون برای واضح تر بودن.

۷.۲ گزارش پروژه

یک گزارش جامع پروژه آماده شده است که خلاصه‌ای از یافته‌های کلیدی، روش‌ها، نتایج و نتیجه‌گیری‌ها را فراهم می آورد. گزارش شامل تصاویر، جدول‌ها و شکل‌ها است تا تحلیل را واضح کند و زمینه‌ای برای نتایج را فراهم کند.

۸. نتیجه‌گیری

در نتیجه، پیاده‌سازی تکنیک‌های تقسیم‌بندی تصویر برای تشخیص پولیپ با استفاده از مجموعه داده Kvasir-SEG نتایج قابل توجهی را نشان داده است. مدل یادگیری عمیق توسعه یافته، عملکرد قوی در تشخیص و تقسیم‌بندی دقیق پولیپ‌ها از تصاویر اندام گوارشی را نشان می‌دهد. با اتوماسیون فرآیند تشخیص پولیپ، ما می‌توانیم کارآیی و اثربخشی برنامه‌های اسکرینینگ سرطان روده را بهبود بخشیم و در نهایت منجر به بهبود نتایج بیمار و کاهش هزینه‌های بهداشتی شویم.

همچنین نتایج این پروژه نسبت به مقاله موجود پیشرفت های چشمگیری داشته است.

لایه های مورد استفاده در شبکه یونت:

```
inputs = tf.keras.layers.Input((256, 256, 3))
s = tf.keras.layers.Lambda(lambda x: x / 256)(inputs)

# Downward path
c1 = tf.keras.layers.Conv2D(16, (3, 3), activation="relu", kernel_initializer='he_normal', padding='same')(s)
c1 = tf.keras.layers.Conv2D(16, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c1)
p1 = tf.keras.layers.MaxPooling2D((2, 2))(c1)

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

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

# Bottom
c4 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(p3)
c4 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c4)
p4 = tf.keras.layers.MaxPooling2D((2, 2))(c4)

c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(p4)
c5 = tf.keras.layers.Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c5)

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

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

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

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

# Additional layers
c10 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c9)
c11 = tf.keras.layers.Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c10)
c12 = tf.keras.layers.Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_normal', padding='same')(c11)

outputs = tf.keras.layers.Conv2D(1, (1, 1), activation='sigmoid')(c12)

modelUNet = tf.keras.Model(inputs=inputs, outputs=outputs, name='U-Net')

modelUNet.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

دو نمونه از نتایج :