UW – Madison GI Tract Image Segmentation

Praneet Kumar Alamuri, Shreyansh Agrawal, Soumya Agrawal, Aashi Aashi, Apurva Audi

Agenda

01 Problem Statement

02 Data Description

03 Pre-Processing

04 Methodology and Modelling

05 Results and Conclusion

06 Future Scope

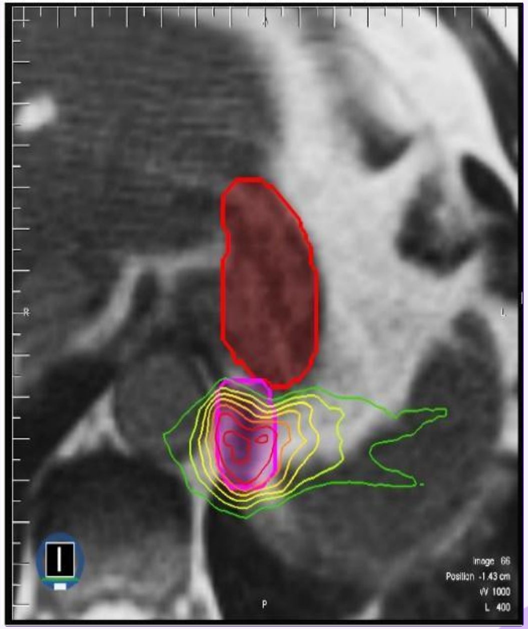
Treatment of Gastrointestinal Cancer

1. Around 5 million people are diagnosed with cancer of the gastrointestinal tract worldwide.

2. Radiation oncologists deliver high doses of radiation using X-ray beams pointed at tumors while avoiding the stomach and intestines.

3. Oncologists manually outline the position of the stomach and intestines in order to adjust the direction of the X-ray beams to target the tumor and avoid the stomach and intestines

4. This is a time-consuming and intensive process that can prolong treatments from 15 minutes a day to an hour a day



The tumor (pink line) is close to the stomach (red line)

* This problem is part of a Kaggle competition (Link)

Problem Statement

1. We want to segment the stomach, large intestine, and small intestine on MRI scan images using semantic image segmentation to accurately deliver radiation doses to GI tract tumor patients while avoiding the stomach and intestines

2. This will help doctors deliver effective and faster treatment to cancer patients

3. Additionally, since semantic modeling is a black box, we are also attempting to understand which pixels are important in distinguishing the segments

Dataset

This dataset is part of the Kaggle Competition **UW-Madison GI Tract Image Segmentation** to track healthy organs in medical scans to improve cancer treatment. We are using anonymized MRI scan images sourced from UW's Cancer Center.

Image Scans

16-bit gray scale in PNG format with annotations as RLE-encoded masks and different scans

Cases

Each case identified by day of scan with total scans of 38.5k

Input File

Image scans for each case with csv file consisting of id, predicted class & Encoded RLE pixels

Methodology

1

Data Pre-Processing

Cleaning, normalizing, RLE Decoding and vectorizing the images to create dataset for training

2

EDA

Visualizing the images and understanding the proportion of data available in each segments

3

Model Training

We trained 4 different models using UNET and LinkNet architectures

4

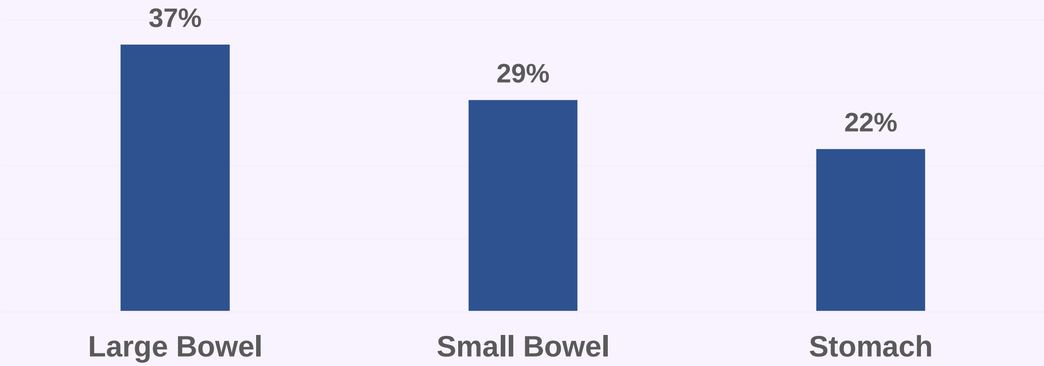
Evaluation & Inferences

Comparing the model performances on test set and interpreting the pixel importance

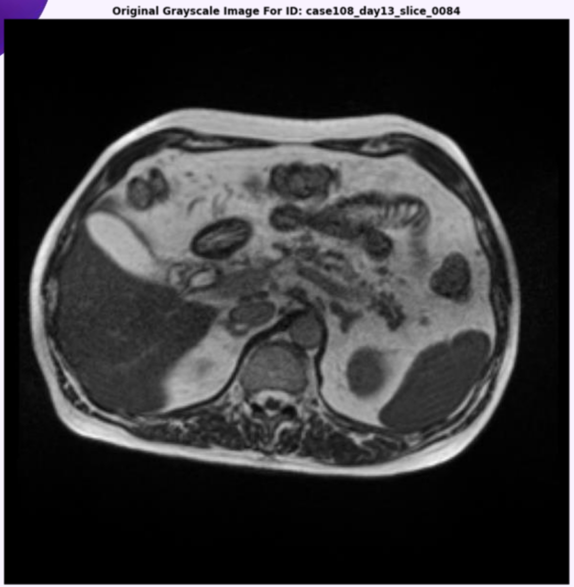
Investigating Image Sizes and Masks

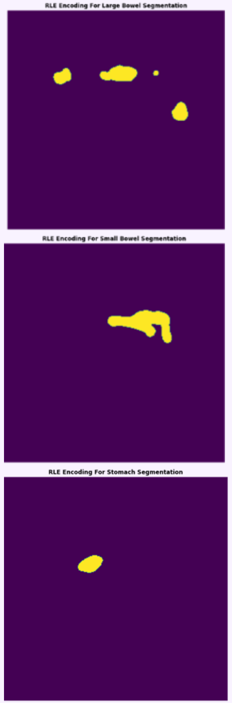
Total images : **38,496**

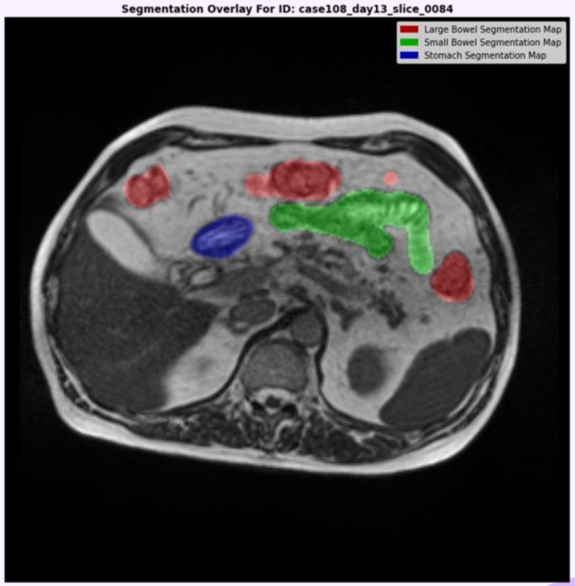
1. Annotation Segments : **no annotation** (56%) , **single** (6%) , **double** (28%) , **all segments** (10%)
2. Image Sizes : **266 * 266** (67%) , **310 * 310** (29%) , **276 * 276** (3%) , **234 * 234** (1%)
3. Below is the share of training images with respective masks



Visualizing MRI scan and masks







MRI Scan

Decoded RLE Masks

Segment Overlay

Evaluation Metric – Dice Coefficient

- Dice coefficient is one of the most broadly used evaluation metrics in Image Segmentation
- The coefficient ranges from 0 to 1; 0 indicating no overlap and 1 indicating complete overlap
- It compares the pixel-wise agreement between a predicted segmentation and its corresponding ground truth by calculating the similarity between the two datasets

$$\frac{2 * |X \cap Y|}{|X| + |Y|}$$

1. X and Y are the two datasets to be compared
2. X ∩ Y represents the number of pixels with the same segment
3. X + Y are the total number of pixels across the two datasets

Pre-Processing

Creating the training dataset

1. The masks for Stomach, large bowel and small bowel were available in the RLE format indicating the pixels for the respective object in a CSV file

2. So, we had to map the image paths and masks in order to overlay the masks on top of the images

Image Resizing

1. Our dataset had images of 4 different sizes
   - 234 x 234 - < 1%
   - 266 x 266 – 67%
   - 276 x 276 – 3%
   - 310 x 310 – 29%

2. To create our feature space, we normalized the images and masks to 256 x 256 size

Pre-Processing

Image Vectorization

1. We used the Keras Data Generator to vectorize the images to create our feature space

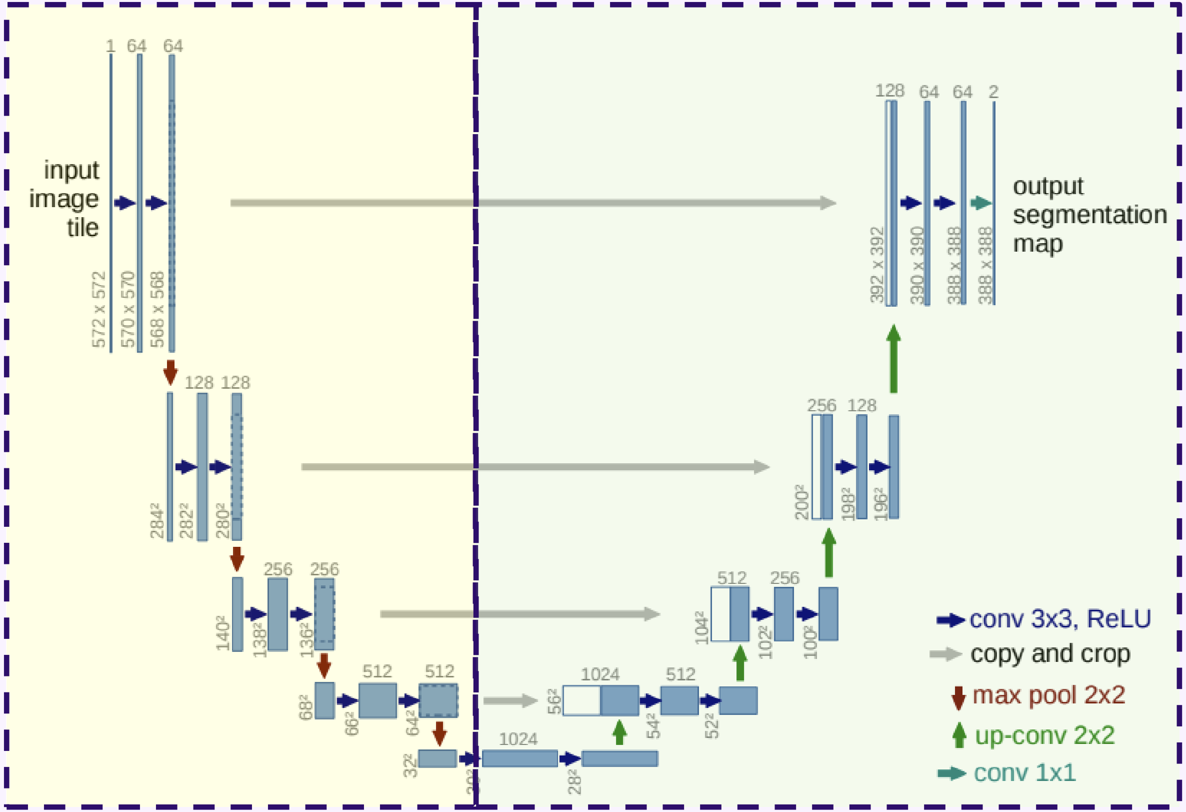
2. The RLE masks were then decoded to represent the target classes in vectorized format

Train – Test - Validation Split

1. As Kaggle does not release the test dataset, we split our train dataset using a 90-10 split to create the test set

2. The training set was then further divided into 5-fold validation set

U-Net Model Overview



Encoder Block

Decoder Block

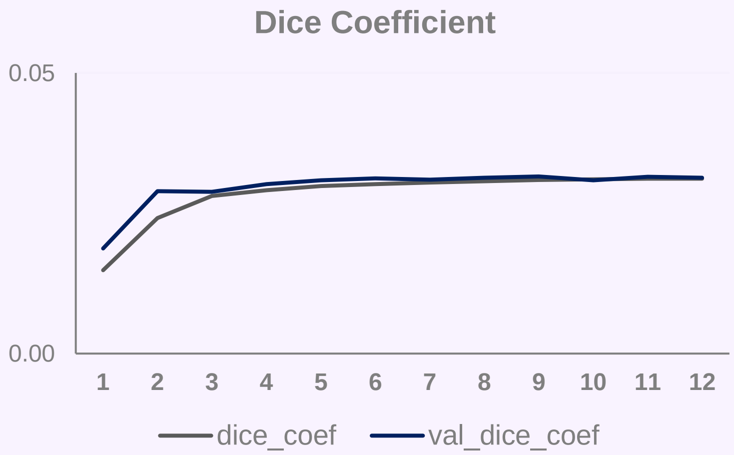
Total Parameters: **~2 M**

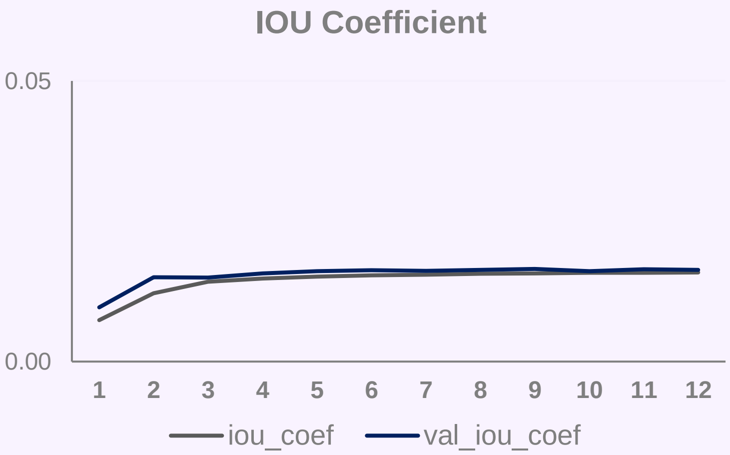
We tweaked some parameters such as -

- convolution layer size (16,32,64,128,256)
- added 20% drop out between the convolution layers to avoid overfitting

Image Source: https://arxiv.org/pdf/1505.04597v1.pdf

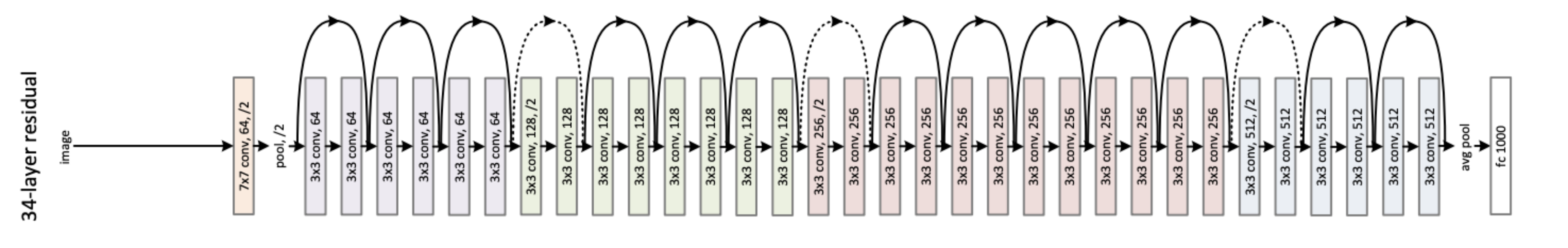
U-Net Results





- The model performance stagnated after 12 Epochs, and we see that the train and validation dice coefficient are roughly 0.03, and the IOU coefficient is around 0.02
- This model has very poor performance and hence we tried using a more involved architecture to improve the model performance

U-Net w ResNet Model Overview



Total Parameters: **~24 M**

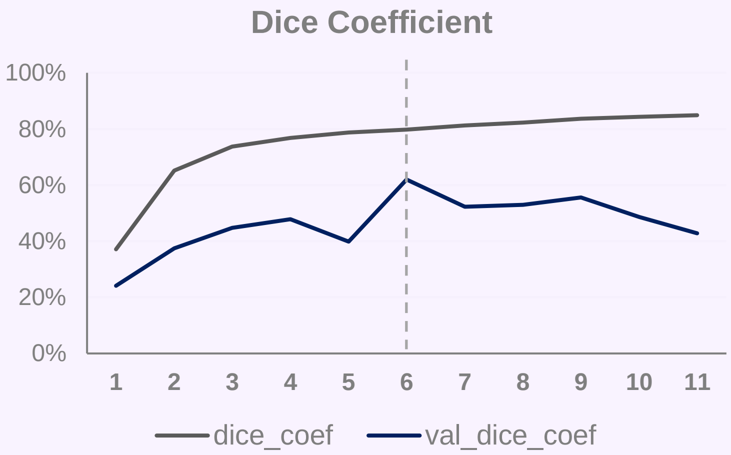
MIT has developed a package for semantic segmentation (segmentation models) which can use these architectures as backbone to create UNET models.

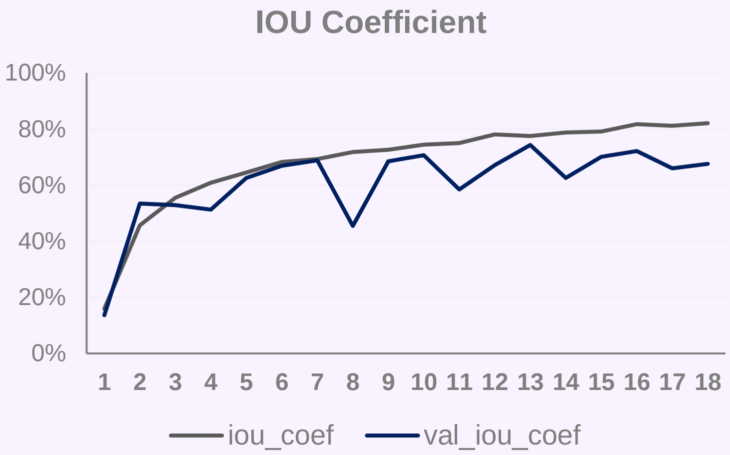
We trained the model using Resnet-32 as the backbone for UNET and initialized the weights for encoder block using pre-trained ImageNet weights

The model gave significantly better results compared to the vanilla UNET model

Image Source:

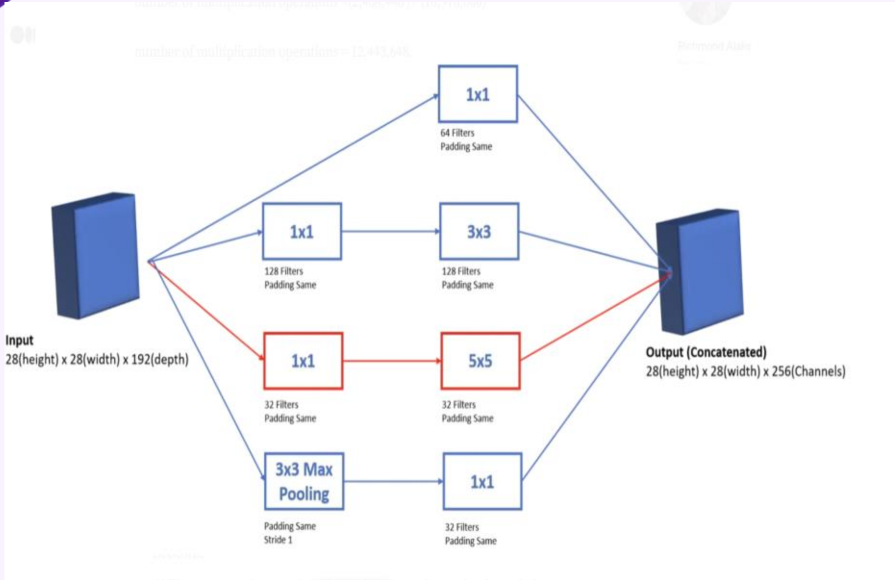
U-Net w ResNet Model Results





- The model stopped after 11 Epochs, and we see that the train dice coefficient is ~0.8 and validation is ~0.62, however the IOU coefficient is comparable for test and validation sets

U-Net w Inception Overview



Total parameters: **29.9M**
Trainable parameters: **29.8M**
Non-trainable parameters: **36K**

Reasons

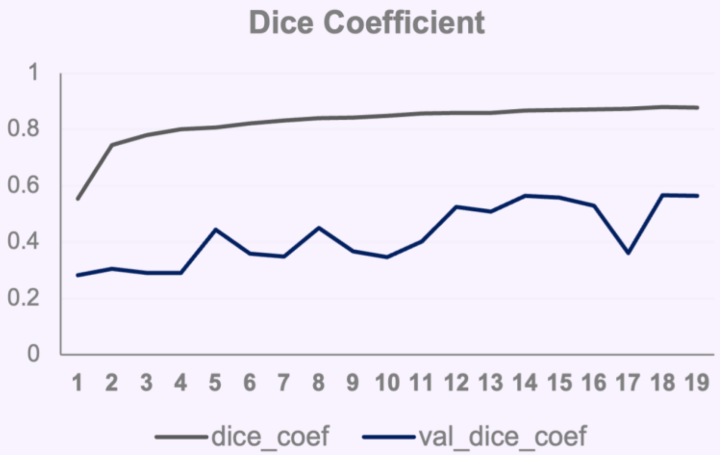
When multiple deep layers of convolutions were used in a model it resulted in the overfitting of the data.

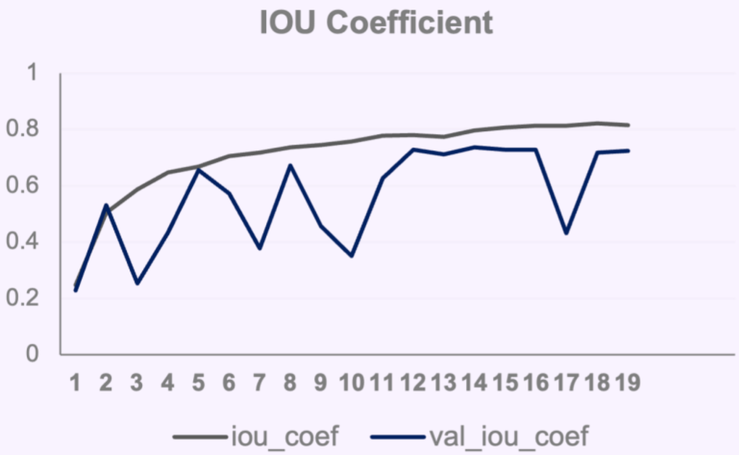
Logic

Inception V1 model uses the idea of using multiple filters of different sizes on the same level

Instead of having deep layers, it has parallel layers thus making model wider rather than making it deeper.

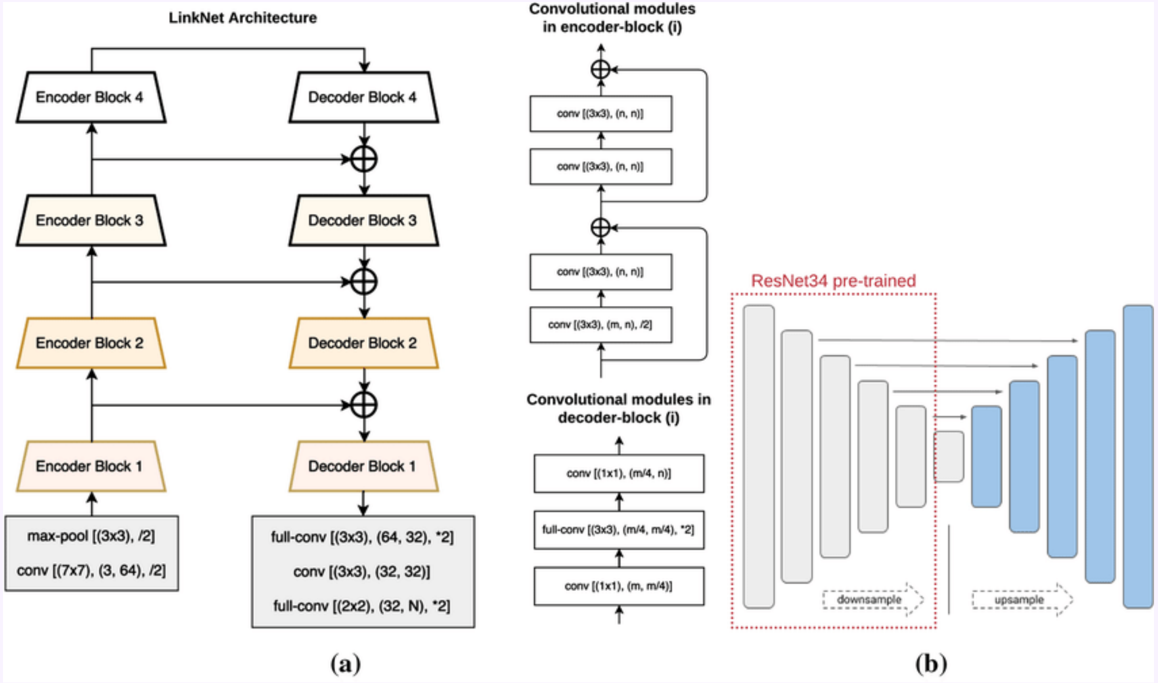
U-Net w Inception Results





- The model performance stagnated after 19 Epochs, and we see that the train dice coefficient is ~0.87 and validation is ~0.57, however the IOU coefficient is comparable for test and validation sets

LinkNet w ResNet Model Overview



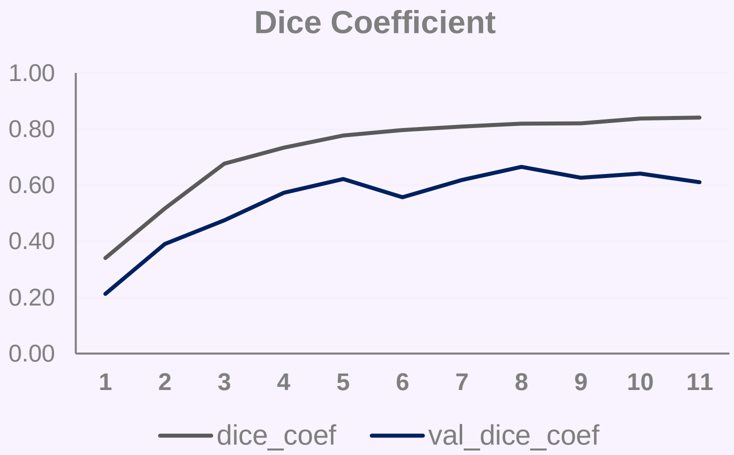
Total Parameters: **22 M**

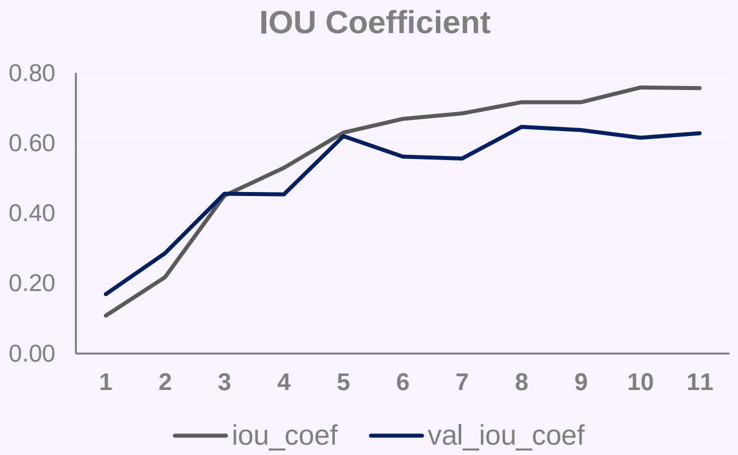
We trained the model using Resnet-32 as the backbone for LinkNet and initialized the weights for encoder block using pre-trained ImageNet weights

The model gave significantly better results compared to the vanilla UNET model

Image Source: https://www.researchgate.net/publication/354225771_Automatic_segmentation_of_melanoma_skin_cancer_using_transfer_learning_and_fine-tuning

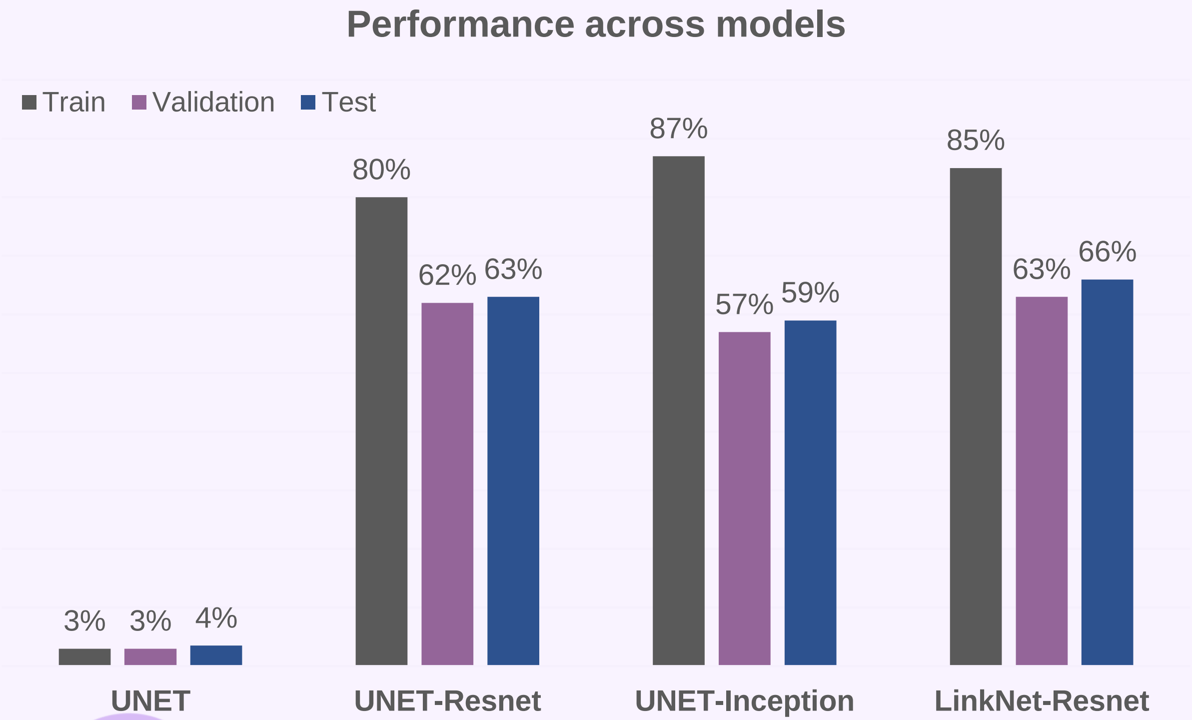
LinkNet w ResNet Model Results





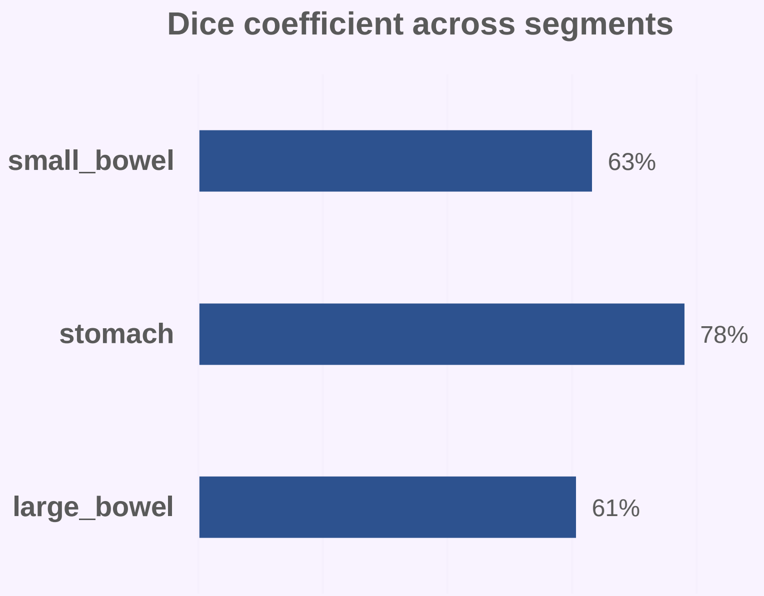
- The model performance stagnated after 11 Epochs, and we see that the train dice coefficient is ~0.85 and validation is ~0.63, however the IOU coefficient is comparable for test and validation sets

Comparison of Results



- All models gave comparable results; however, we recommend picking LinkNet-Resnet because it is giving slightly better results
- Vanilla UNET model gives a very poor performance owing to less complexity

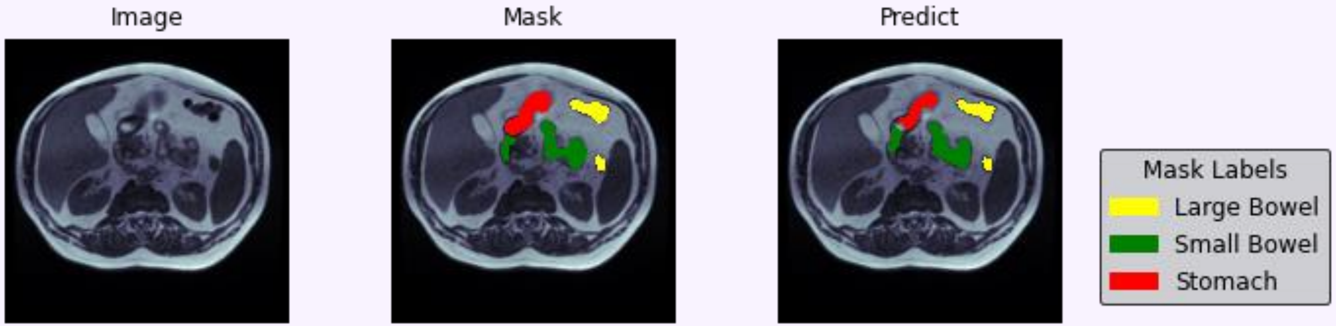
Model performance across segments



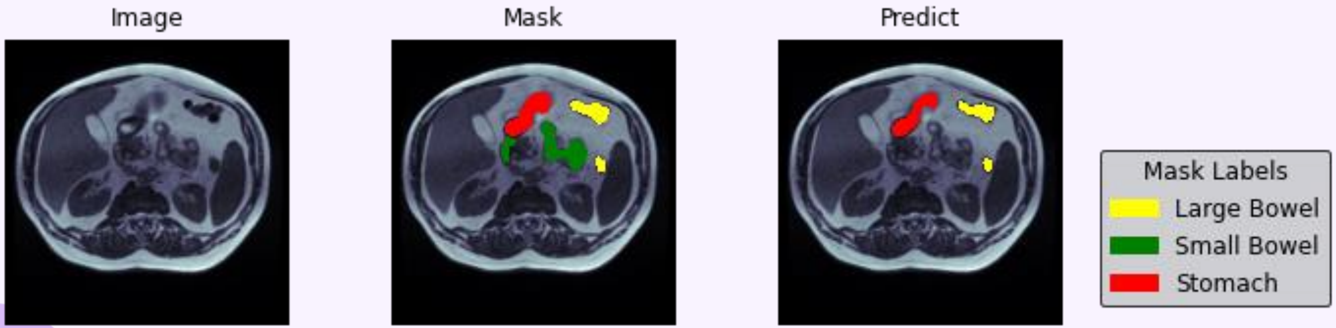
- We see that the dice coefficient for small and large bowels are comparable close to 60%
- Dice coefficient for stomach is higher close to 80%
- This makes sense as stomach segment has a relatively larger mask
- So, our model is more accurate in segmenting stomach compared to intestines

Variation in predictions across models

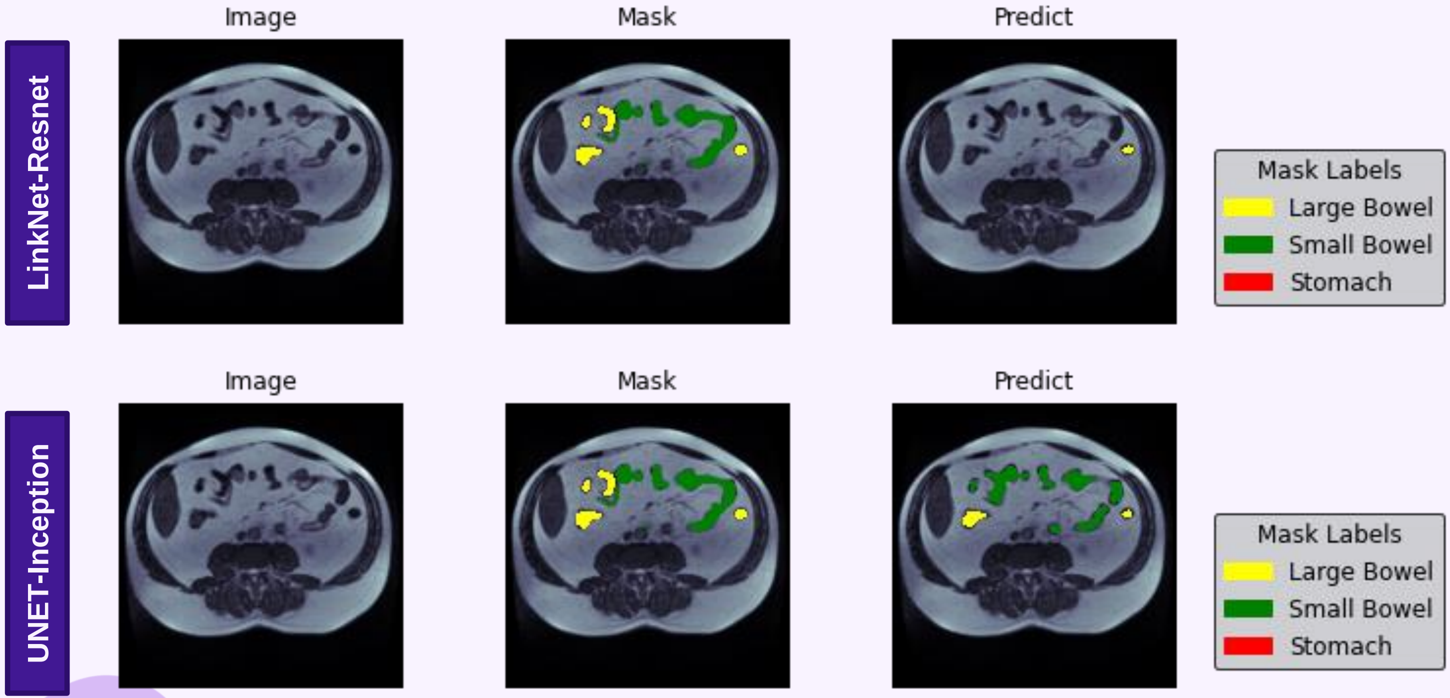
LinkNet-Resnet



UNET-Inception



Variation in predictions across models



Interpreting the black box

t-CAV

- Explored this but couldn’t find the implementation for semantic segmentation
- Could only find use cases for image segmentation
- Creating concepts is challenging without medical domain expertise

SEG Grad Cam

- Extension of Grad Cam, specifically designed to interpret the pixel in a semantic segmentation use case
- Only compatible with TensorFlow 1.14.0
- TensorFlow 1 is not supported in either Kaggle or Google Collab
- Required retraining the models in local system which was infeasible owing to computational limitations

Looking Forward...

- We can use data augmentation to generate more training data which could potentially improve our model performance
- We can also try creating an ensemble of the models we proposed to create a better performing model
- Advanced methods such as 3D and 2.5D modelling can improve the model performance
- Figure out a way to regularize the UNET/ LinkNet segmentation models to improve out of sample performance

Thank You!