



DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation



Debesh Jha
(UiT The Arctic University of Norway)



Michael A. Riegler
(SimulaMet, Norway)



Pål Halvorsen
(Oslo Metropolitan University, Norway)



Dag Johansen
(UiT The Arctic University of Norway)



Håvard D. Johansen
(UiT The Arctic University of Norway)



AIM

Semantic image segmentation

Built for medical purposes

Proposed upgradation over the original U-net

Motivation

The output of U-net can be further refined

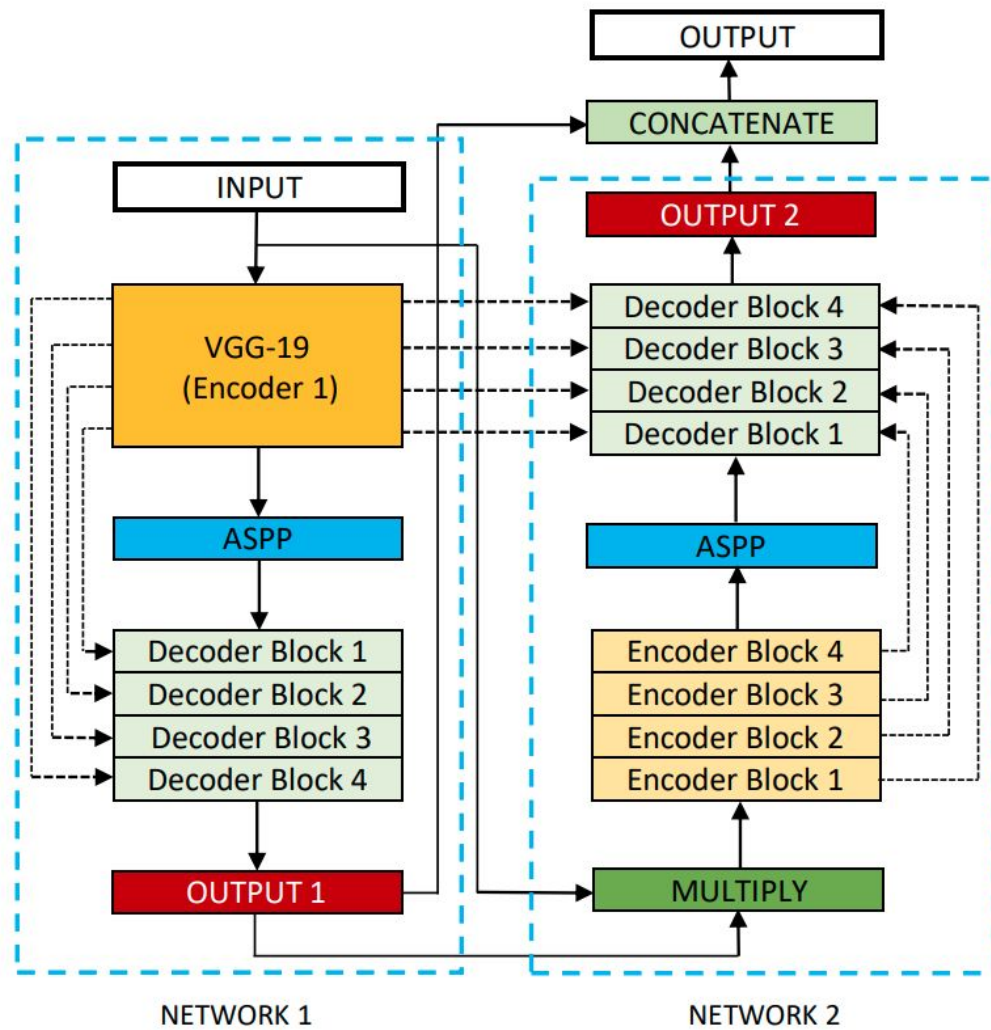
The Double U-net Architecture

U-net gained much popularity

Many variants of U-net were made (attention U-net, inception u-net, double U-net)

- Two U-nets in series
- Use of VGG-19
- Atrous Spatial Pyramid Pooling (ASPP)
- Squeeze and excite blocks





VGG-19

Pre-trained on ImageNet database

Compensates for lack of enough training data

Squeeze and Excite

Shrinks every 2-D channel into a scalar

Forces network to learn channel wise dependencies

weighted concatenation

ASPP

Multiple parallel atrous convolutional layers
with different rates

Feature extraction at wide range of scales





DATASET

DATA

DoubleU-Net was evaluated using four medical segmentation datasets, covering various imaging modalities such as colonoscopy, dermoscopy, and microscopy.

Dataset	No. of Images	Input size	Application
2015 MICCAI sub-challenge on automatic polyp detection dataset	808	384×288	Colonoscopy
CVC-ClinicDB	612	384×288	Colonoscopy
Lesion Boundary Segmentation challenge	2594	Variable	Dermoscopy
2018 Data Science Bowl Challenge	670	256×256	Nuclei

A single image was converted into 25 different images; thus, in total, 26 images including the original image. The augmentation techniques used were optical distortion, grid distortion, elastic transform, vertical and horizontal flip to name a few.

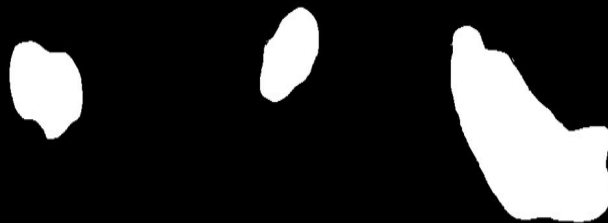
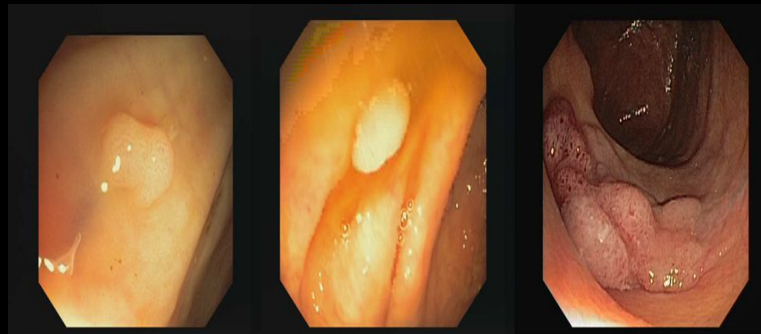
CVC - ClinicDB

CVC-ClinicDB is a database of frames extracted from colonoscopy videos.

It has 612 still images from 29 different sequences. Each image has its associated manually annotated ground truth covering the polyp.

Hyperparameters particular to the dataset:

- *Loss - Binary crossentropy*
- *Optimizer - Nadam*
- *Learning Rate - $1e-5$*



Loss : Binary crossentropy

- Measure of the difference between two probability distributions for a given random variable or set of events.

Binary Cross-Entropy is defined as:

$$L_{BCE}(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

Here, \hat{y} is the predicted value by the prediction model.

Optimizer : Nadam

- Extension of the Adam algorithm that incorporates Nesterov momentum



Implementation Details

The code is written in python and framework used is TensorFlow.

We have used online platform Kaggle for training and testing of the model using GPU accelerator.

The dataset is split in 80:10:10 for train, validation and test respectively.

The main components of the code are -

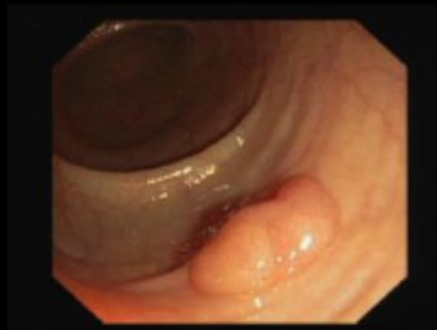
- Data loading and augmentation
- Building and training the model
- Evaluation on test set

Data augmentation

- Center Crop
- Random rotate 90 degrees
- Transpose
- Elastic transform
- Grid distortion
- Optical distortion
- Vertical flip
- Horizontal flip
- Grayscale
- Grayscale vertical flip
- Grayscale horizontal flip
- Grayscale center crop
- Random brightness
- Random contrast
- Random gamma
- RGB Shift
- Hue Saturation
- Motion blur
- Gaussian Blur
- Gaussian noise
- Channel chuffle
- Coarse dropout



A few examples



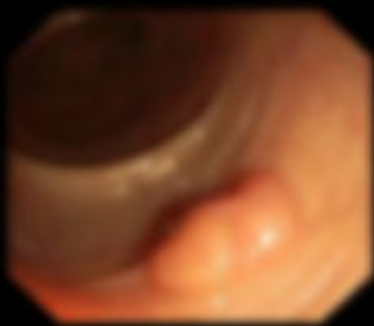
Original polyp image



Vertical Flip



Horizontal Flip



Random Gaussian blur



Random Contrast



Random brightness

Encoder 1

Encoding is done by VGG19, with weights pre-trained on Imagenet.

```
def encoder1(inputs):
    skip_connections = []

    model = VGG19(include_top=False, weights='imagenet', input_tensor=inputs)
    names = ["block1_conv2", "block2_conv2", "block3_conv4", "block4_conv4"]
    for name in names:
        skip_connections.append(model.get_layer(name).output)

    output = model.get_layer("block5_conv4").output
    return output, skip_connections
```

Encoder 2

```
def encoder2(inputs):
    num_filters = [32, 64, 128, 256]
    skip_connections = []
    x = inputs

    for i, f in enumerate(num_filters):
        x = conv_block(x, f)
        skip_connections.append(x)
        x = MaxPool2D((2, 2))(x)

    return x, skip_connections
```

Decoder

```
def decoder1(inputs, skip_connections):
    num_filters = [256, 128, 64, 32]
    skip_connections.reverse()
    x = inputs

    for i, f in enumerate(num_filters):
        x = UpSampling2D((2, 2), interpolation='bilinear')(x)
        x = Concatenate()([x, skip_connections[i]])
        x = conv_block(x, f)

    return x
```

Training

The model is trained for 120 epochs and evaluated on step-loss, dice coefficient, iou, recall and precision. Total training time is approximately 33 hours.

Training parameters

Batch size = 8

Initial learning rate = $1e-5$

Optimizer = Nadam

```
callbacks = [  
    ModelCheckpoint(model_path),  
    ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=20),  
    CSVLogger("files/data.csv"),  
    TensorBoard(),  
    EarlyStopping(monitor='val_loss', patience=50, restore_best_weights=False)  
]
```


Metrics


❑ DSC (F1 Score)

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

❑ Recall

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

❑ IoU

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$
The diagram shows two overlapping rectangles. The top rectangle is light blue, and the bottom rectangle is light red. The intersection of the two rectangles is shaded light purple. The formula for IoU is shown to the left of the rectangles.

❑ Precision

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$



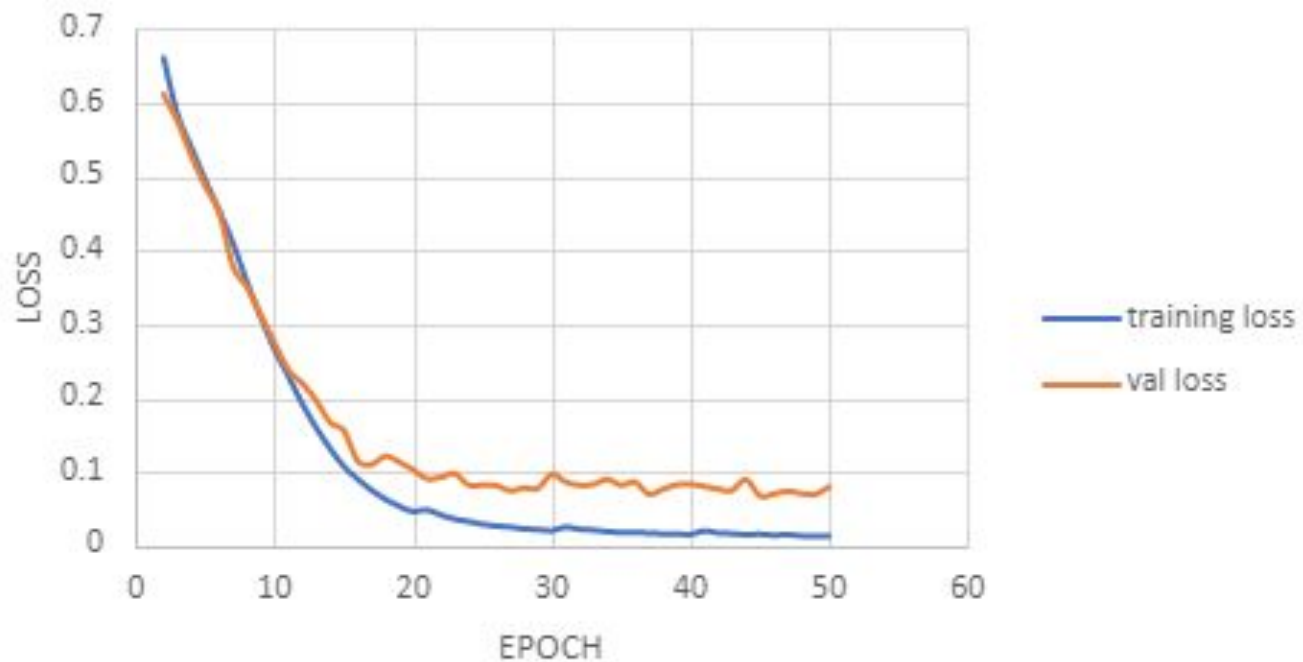
Training Results

Training and Validation set results after 110 epochs.

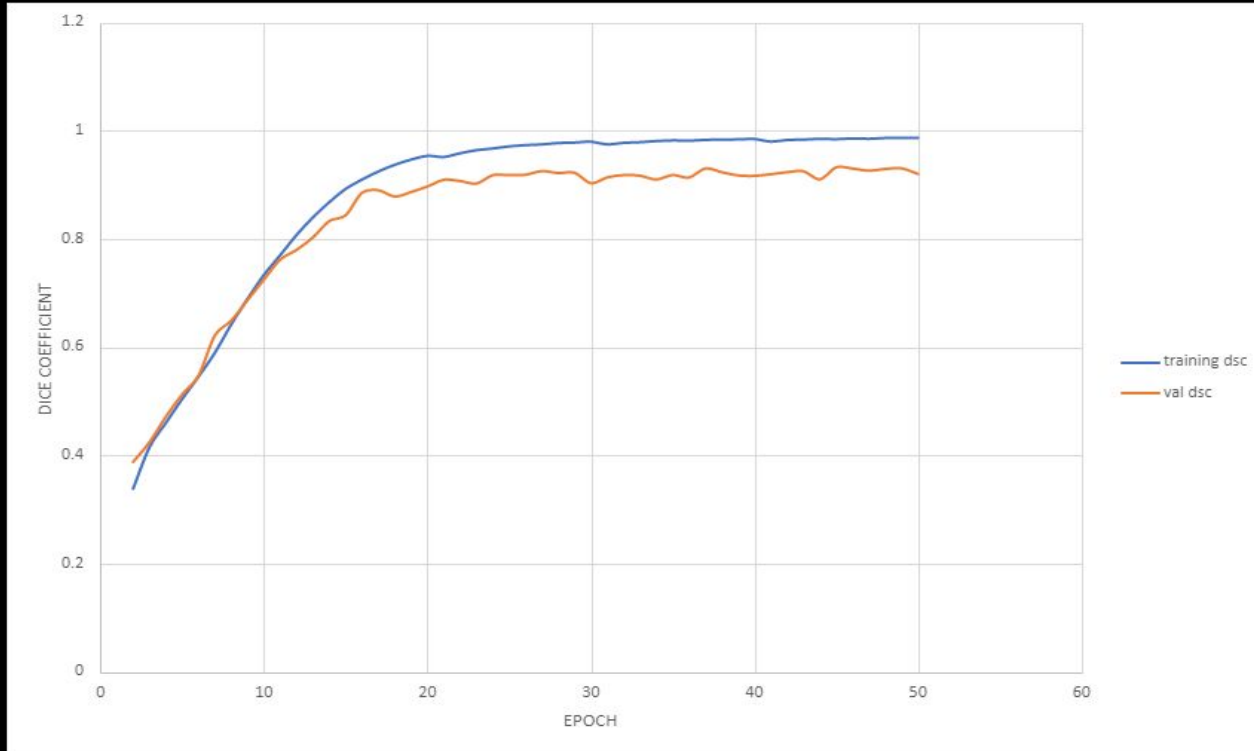
	DSC	IoU	Recall	Precision
Training set	0.9914	0.9830	0.9238	0.9959
Validation set	0.9340	0.8870	0.8904	0.9505



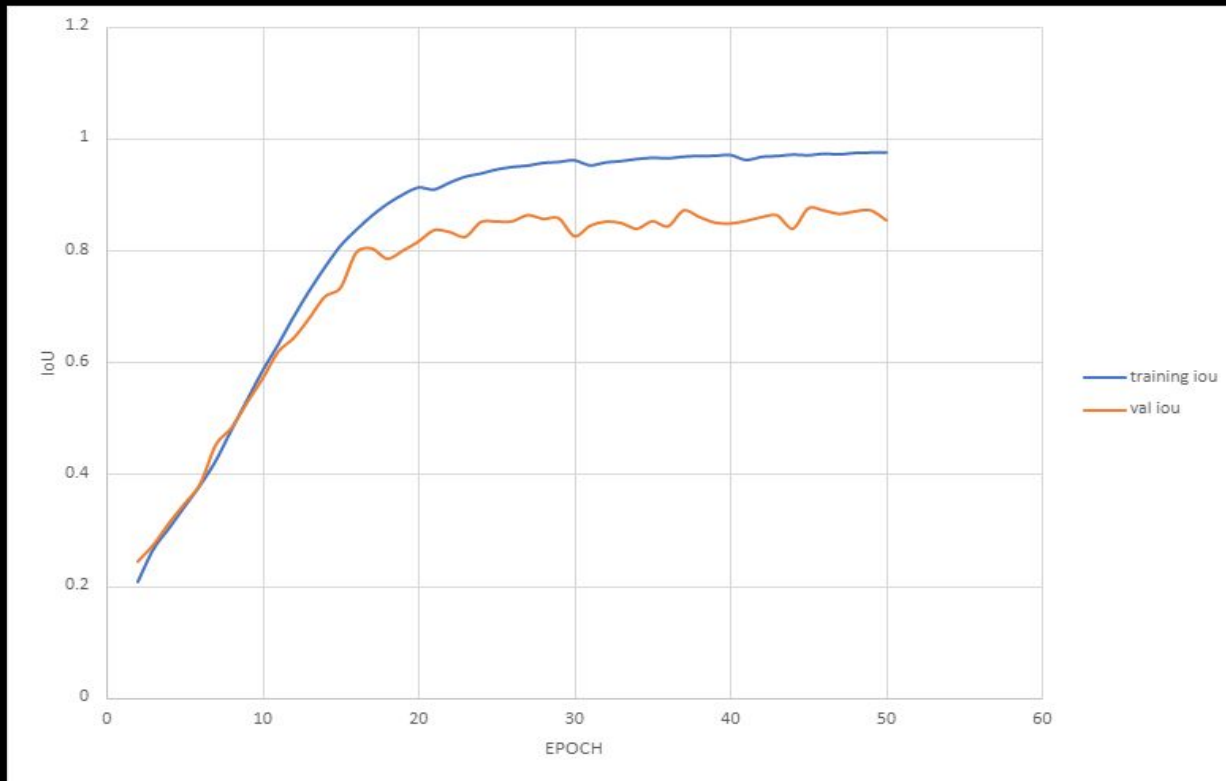
Loss



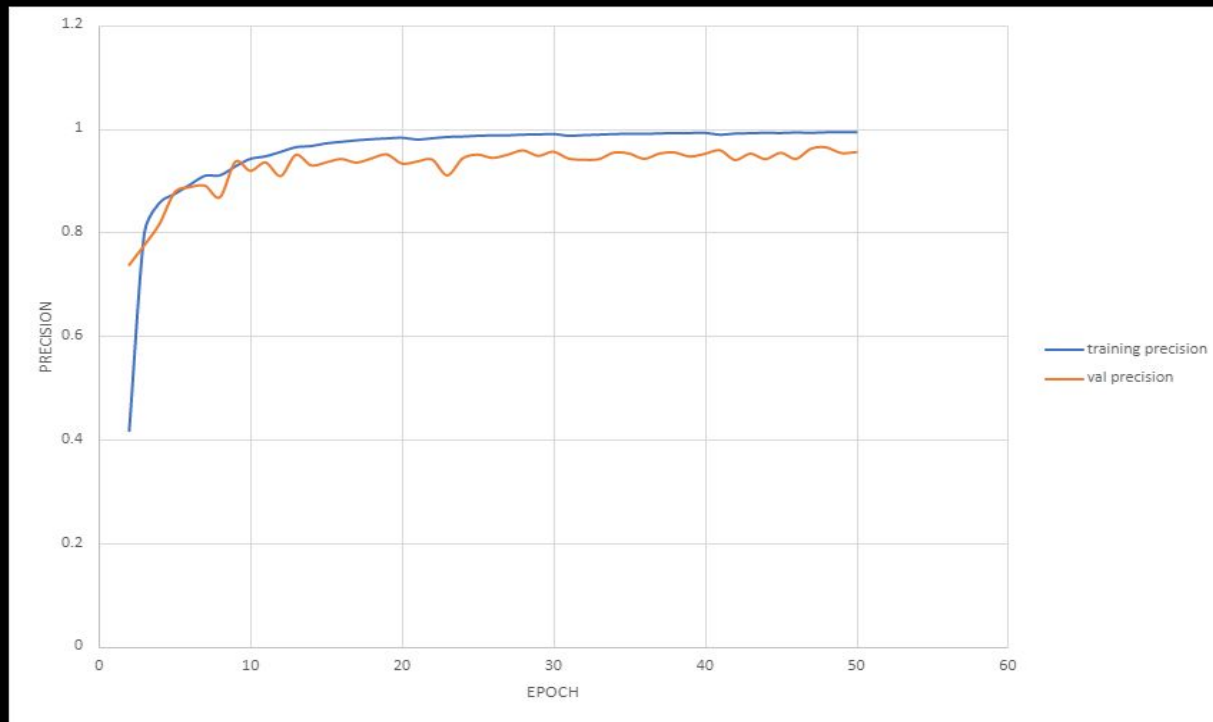
Dice coefficient



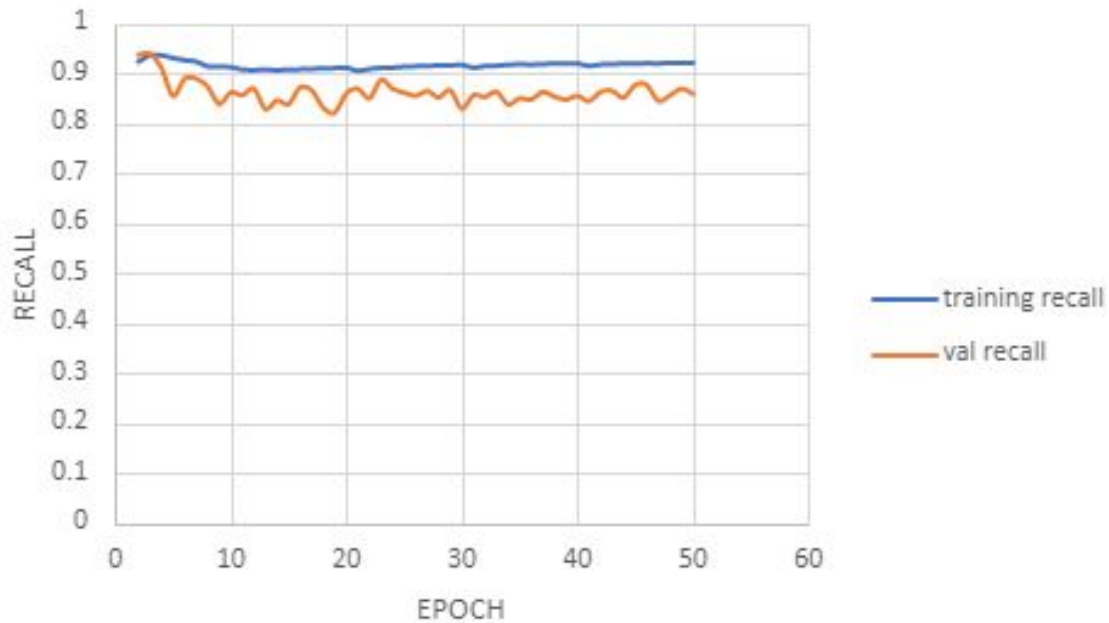
IoU



Precision



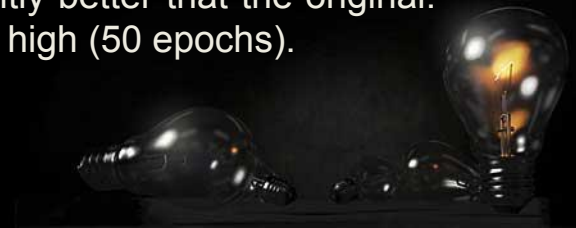
Recall



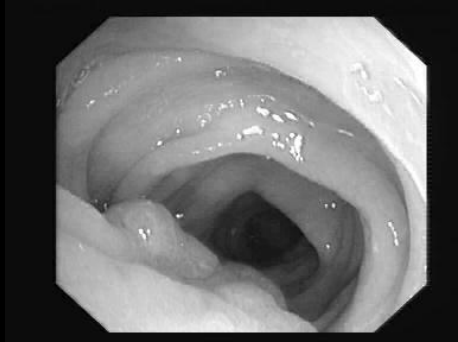
Result comparison on CVC-ClinicDB by authors

	DSC	IoU	Recall	Precision
Our results	0.9441	0.8945	0.8967	0.9557
Author's results	0.9239	0.8611	0.8457	0.9592

Originally the model is trained for 300 epochs. We trained the model for 120 epochs. The comparison table shows that our trained model has performed slightly better than the original. The reason for this could be that the patience for early stopping was high (50 epochs).



Model Predictions



Challenges faced in implementation

Challenges faced in implementation

- ❑ With batch size of 16, the time estimated for 300 epochs was around 70 hours with each epoch taking around 14 minutes.
- ❑ We implemented the paper with batch size of 8, due to ResourceExhaustsError and time taken for each epoch was 950 seconds.
- ❑ Learning rate was reduced to $\frac{1}{4}$ of the original learning rate after 110 epochs to check for some significant improvements in metrics like val_dice.
- ❑ Stable internet connectivity
- ❑ Debugging author's code

Future Scope

- ❑ Observe the model output with changing encoders. Resnet or Densenet can be used for encoding.
- ❑ Initialize the weights pre-trained on medical image segmentation database instead of Imagenet.
- ❑ Varying the filter size in encoder and decoder layers.
- ❑ Use different optimizer for training.
- ❑ Test the model on other medical image segmentation datasets to understand the shortcomings of the current model.



Learning outcomes


- ❑ Learnt about Kaggle as a platform where multiple persons can collaborate to run and train models.
- ❑ Understood about the issues that are faced while building a model, and various methods that are used to solve them.
- ❑ Learnt about medical image segmentation.
- ❑ Learnt about new architectures.



References

- Polyp - Grand Challenge. (n.d.). Retrieved from <https://polyp.grand-challenge.org/CVCClinicDB/>
- Chen, Liang-Chieh et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs.” IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (2018): 834-848.
- Hu, Jie et al. “Squeeze-and-Excitation Networks.” 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (2018): 7132-7141.
- Ronneberger, Olaf et al. “U-Net: Convolutional Networks for Biomedical Image Segmentation.” MICCAI (2015).



A close-up photograph of a hand holding a string of warm-toned lights. The lights are out of focus, creating a bokeh effect with soft, glowing circles of light against a dark background. The hand is visible on the left side, holding the string.

PURUSHARTH AMRUT
HOMI RAGHUVANSHI
REVA TEOTIA

THANK YOU