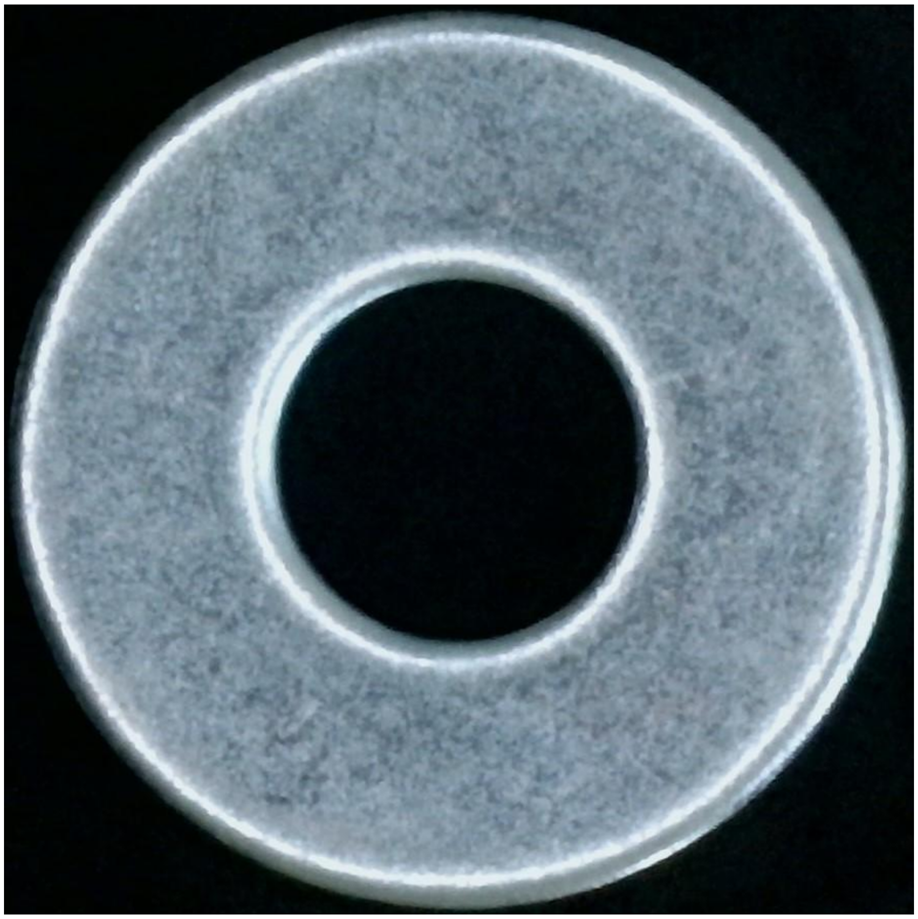


Project Name : Anomaly Detection on Washer Dataset.

Submitted By: Rahman Sagidur

Problem Statement:

Anomaly detection on washer dataset using a good learning algorithm (example: Variational AutoEncoder [VAE]). Train the model with "washer_ok" dataset and test the model with "washer_ng" dataset. Predict and detect the anomaly with generating heat maps.

OverView:

Anomaly detection is a task of Computer Vision. In Computer Vision there are few ways to detect anomalies. "Image Segmentation" is one of the ways to detect anomilies. There are many Image Segmentation algorithms. In this project task I applied 4 algorithms. After analysing the dataset I did my research and chose those algorithms. I applied those algorithms on the dataset to find out anomalies.

DataSet: The dataset are RGB images.

- ☐ washer_ok : 30 images
- ☐ washer_ng : 92 images

Training Data : washer_ok (30 images)

Validation Data : washer_ng_kizu (10 images)

Test Data : washer_ng_sabi (4 images)

Algorithms : For this project task I applied following algorithms:

1. Variational AutoEncoder (VAE).
2. U-Net.
3. U-Net Xception Style.
4. MultiResUnet.

Development Environment:

OS: Ubuntu 20.04

CPU: AMD Ryzen7 3700U

Development IDE: Anaconda (version: 4.9.2) , Jupyter Notebook(6.0.3)

Programming Language: python (version: 3.7.9)

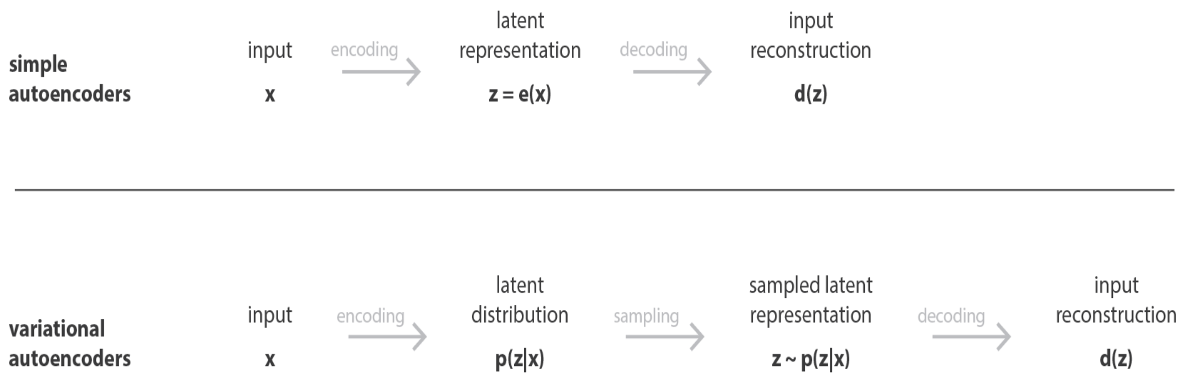
Frameworks: Tensorflow (version: 2.3.0); keras (version: 2.4.3)

Libraries: numpy(version: 1.20.2); matplotlib(version: 3.3.4); pillow(version: 8.2.0)

Model Training:

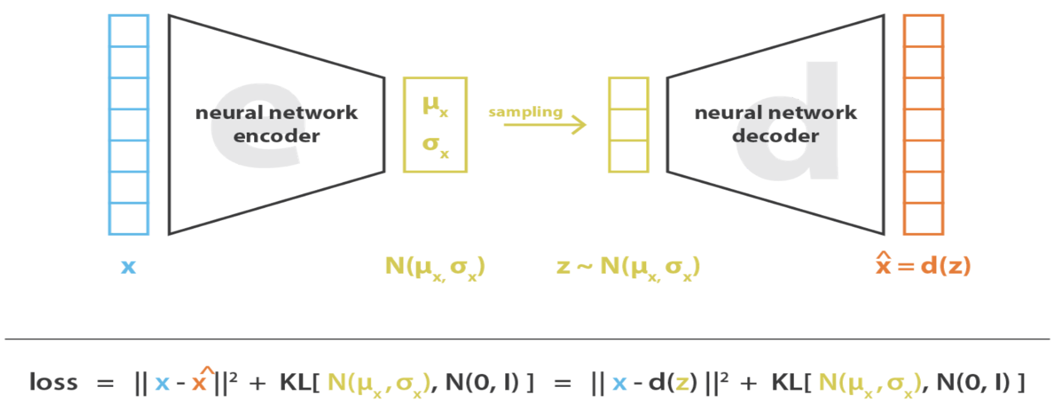
Algorithm 1: Variational AutoEncoder (VAE).

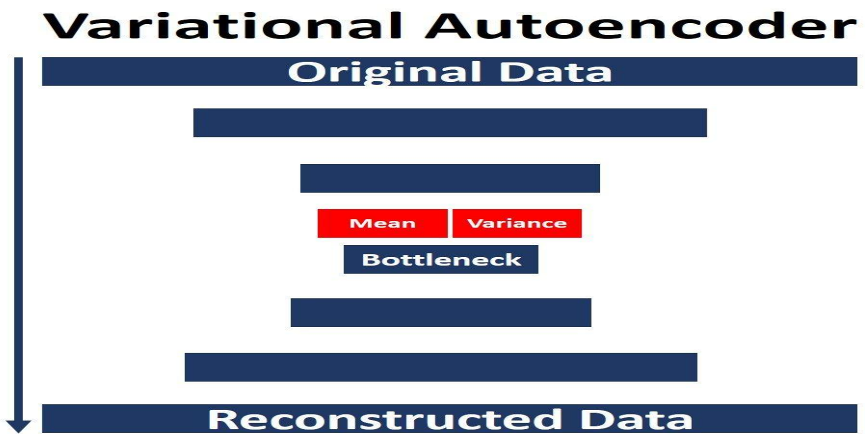
VAE: Variational AutoEncoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable generative process.



1. The input is encoded as distribution over the latent space.
2. A point from the latent space is sampled from that distribution.
3. The sampled point is decoded and the reconstruction error can be computed.
4. Finally, the reconstruction error is back propagated through the network.

VAE Architecture


neural network encoder
μ_x
σ_x
sampling
neural network decoder
x
$N(\mu_x, \sigma_x)$
$z \sim N(\mu_x, \sigma_x)$
$\hat{x} = d(z)$
loss = $\| x - \hat{x} \|^2$ + KL[$N(\mu_x, \sigma_x)$, N(0, I)] = $\| x - d(z) \|^2$ + KL[$N(\mu_x, \sigma_x)$, N(0, I)]



VAE Model Training , Performance and Result

Training:

img_size = (128, 128) # (64,64)

batch_size = 1 #2,3

number_classes = 2 #3,4

number_channels = 3

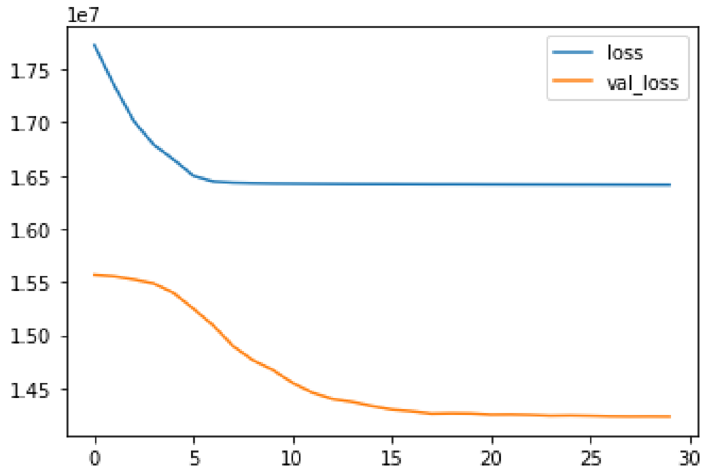
latent_space_dim = 4 #2, 20

optimizer = ADAM (lr = 0.0001) # SGD

epochs = 30 # 50, 100

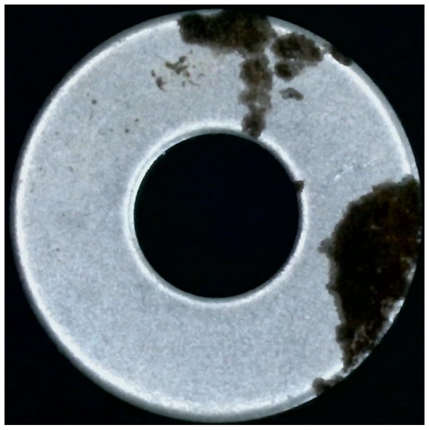
Performance : I applied different img_size, batch_size, latent_space_dimension, number_classes, optimizer and epochs. Here I have attached the optimized result.

Model Loss Graph: Loss starts from 1.77le7 and finish at 1.64le7.

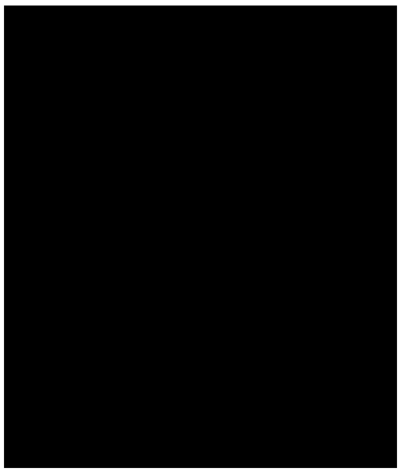


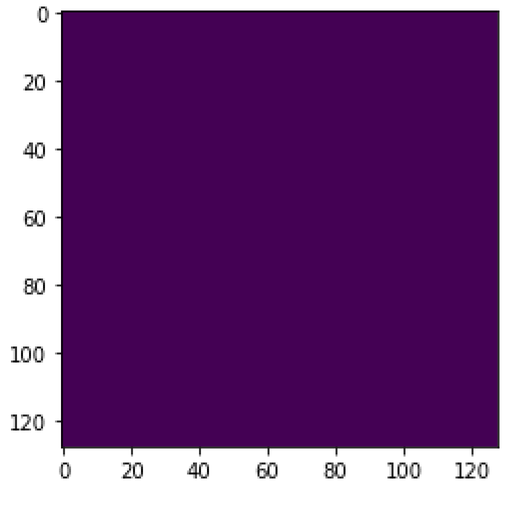
Loss and Validation Loss Graph of VAE model

Model Output on Heat Map:



Input Image





Output Image

Limitations : First of all, the dataset is the most essential point of a good model. Here the model faced a data limitation problem. The model could not learn from less data.

Recommendation : If the number of data is increased the model can learn better during the training period. The model needs fine tuning in the parameters such as batch size, optimizer, learning rate, data augmentation and latent (for VAE).

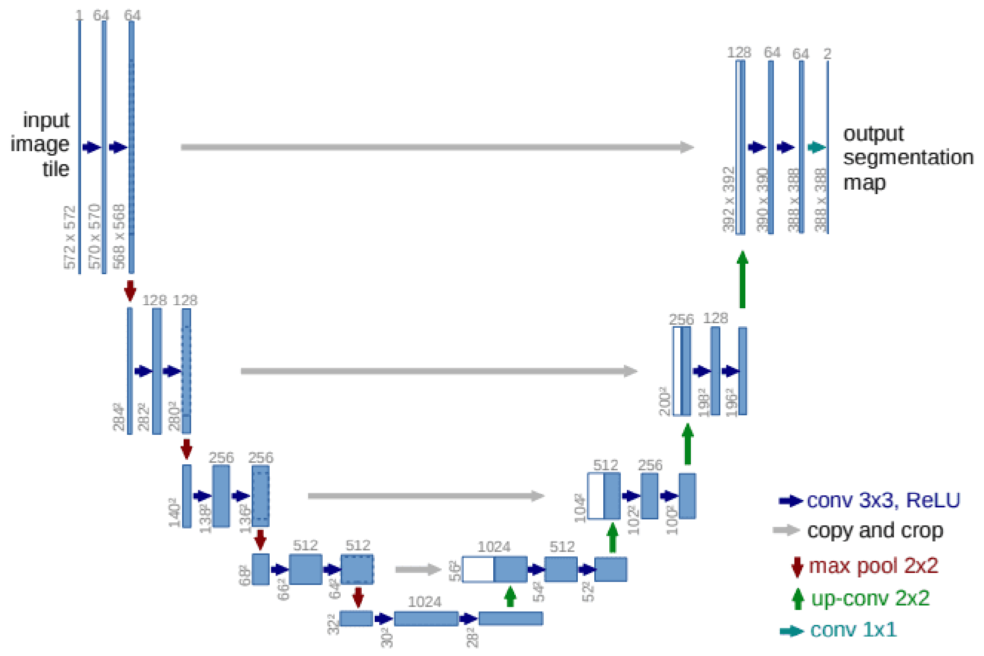
Reference:

1. https://ermongroup.github.io/cs228-notes/extras/vae/
2. https://www.jeremyjordan.me/variational-autoencoders/
3. https://towardsdatascience.com/variational-autoencoders-vaes-for-dummies-step-by-step-tutorial-69e6d1c9d8e9

Algorithm 2: U-Net.

U-Net: A U-shaped architecture consists of a specific encoder-decoder scheme. First path is the contraction path (also called as the encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The second path is the symmetric expanding path (also called as the decoder) which is used to enable precise localization using transposed convolutions. Thus it is an end-to-end fully convolutional network (FCN). It only contains Convolutional layers and does not contain any Dense layer because of which it can accept images of any size.

U-Net Architecture



U-Net Model Training , Performance and Result

Training:

img_size = (256, 256) # (572,572) (512,512)

batch_size = 1 #2,3

number_classes = 2 #3,4

number_channels = 3

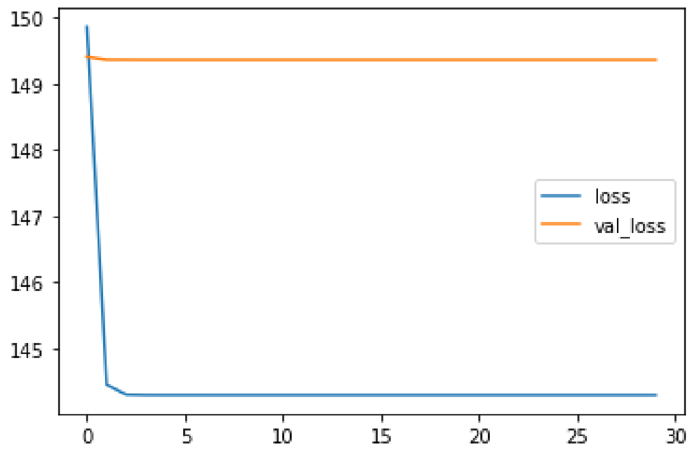
loss= categorical_crossentropy

optimizer = ADAM # SGD

epochs = 30 # 50, 100

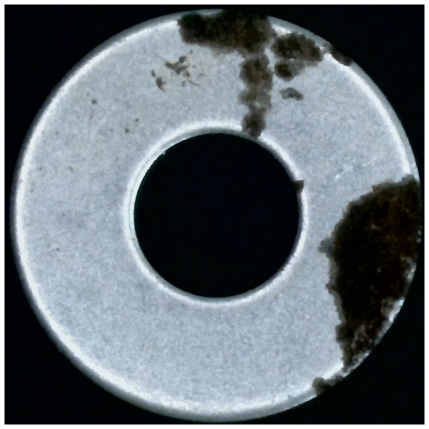
Performance : I applied different img_size, batch_size, number_classes, optimizer and epochs. Here I have attached the optimized result.

Model Loss Graph: Loss started from 145.3586 after 30 epochs it finished at 144.2891 .

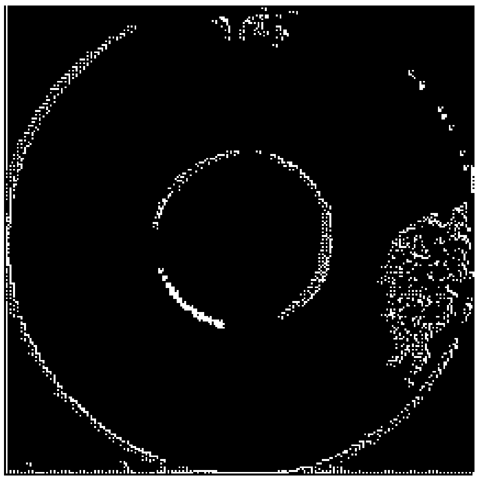


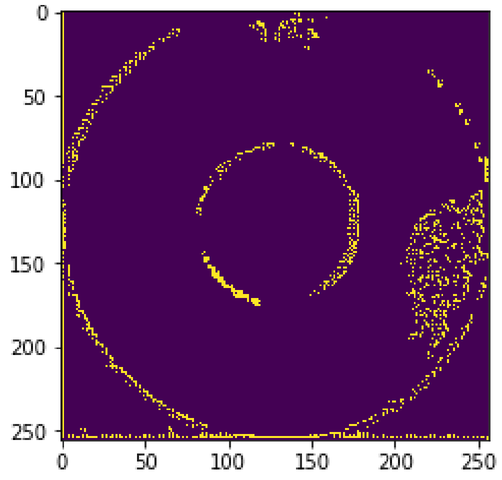
Loss and Validation Loss Graph of U-Net model

Model Output on Heat Map:



Input Image





Output Image

Limitations : First of all, data is the most essential point of a good model. Here the model was trained by very little data. That's why the U-Net model could not learn much from less data.

Recommendation : If the number of data is increased the model can learn better during the training period. The model needs fine tuning in the parameters such as batch size, optimizer, learning rate, data augmentation. Then there is a possibility the performance of the U-Net model will increase.

Reference :

1. http://www-o.ntust.edu.tw/~cweiwang/ISBI2015/challenge2/isbi2015_Ronneberger.pdf
2. https://arxiv.org/pdf/1505.04597.pdf

Algorithm 3: U-Net Xception Style.

This U-Net Xception Style model follows U-Net architecture. So this model also follows the ideal Encoder and Decoder method.

This model showed how to use a U-Net style ConvNet to map from a 160x160xRGB image of a PET into the same 160x160 dimensional annotation map of each pixel in the image. This involves segmenting the pet from its border and the background.

U-Net Xception Style Model Training , Performance and Result

Training:

img_size = (160, 160)   # (256,256) (224,224)

batch_size = 1 #2,3,4

number_classes = 2 #3,4

number_channels = 3

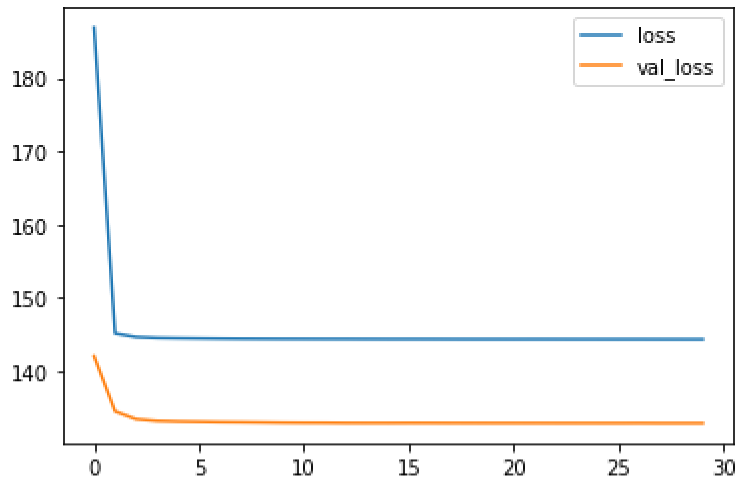
loss= categorical_crossentropy

optimizer = ADAM   # RMSPROP

epochs = 30 # 50, 100

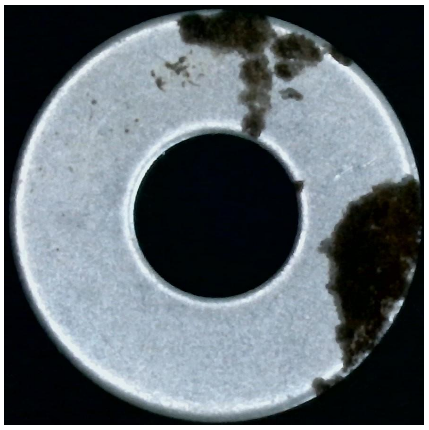
Performance : I applied different img_size, batch_size, number_classes, optimizer and epochs. Here I have attached the optimized result.

Model Loss Graph: Loss started from 186.9564 after 30 epochs it finished at 144.2846 .

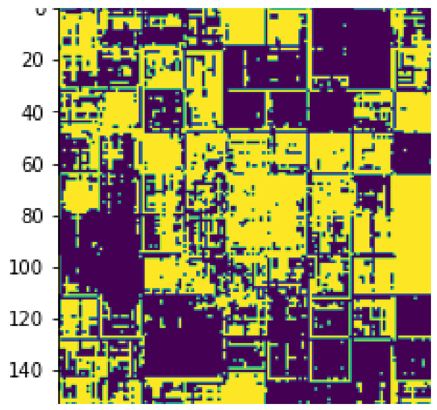


Loss and Validation Loss Graph of U-Net Xception model

Model Output on Heat Map:



Input Image



Output Image

Limitations : First of all, data is the most essential point of a good model. Here the model was trained using very little data. That's why the U-Net Xception Style model could not learn from less data.

Recommendation : If the number of data increases, the model can learn better during the training period. The model needs fine tuning in the parameters such as batch size, optimizer, learning rate, and data augmentation. Then there is a possibility the performance of the U-Net Xception Style model will increase.

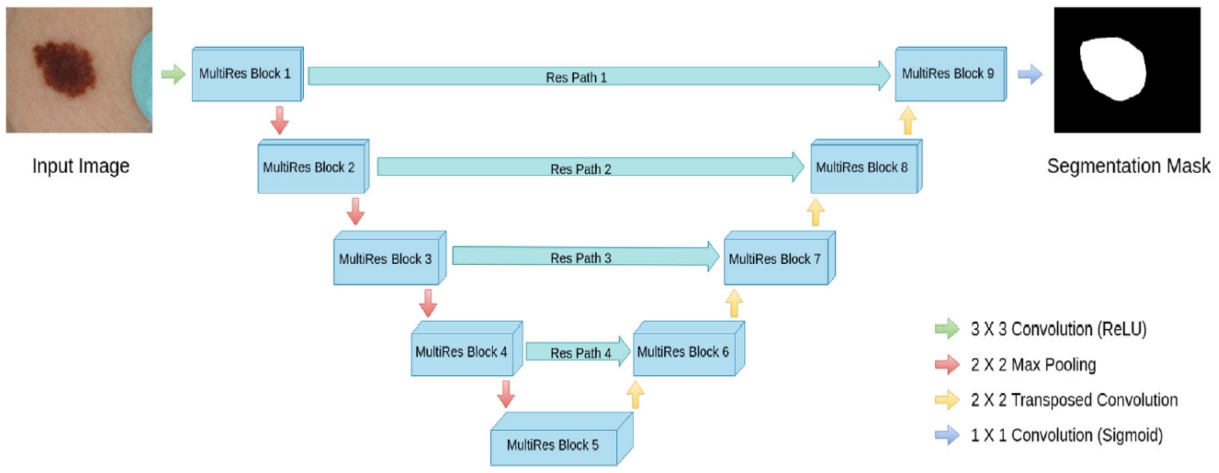
Reference:

1. https://keras.io/examples/vision/oxford_pets_image_segmentation/

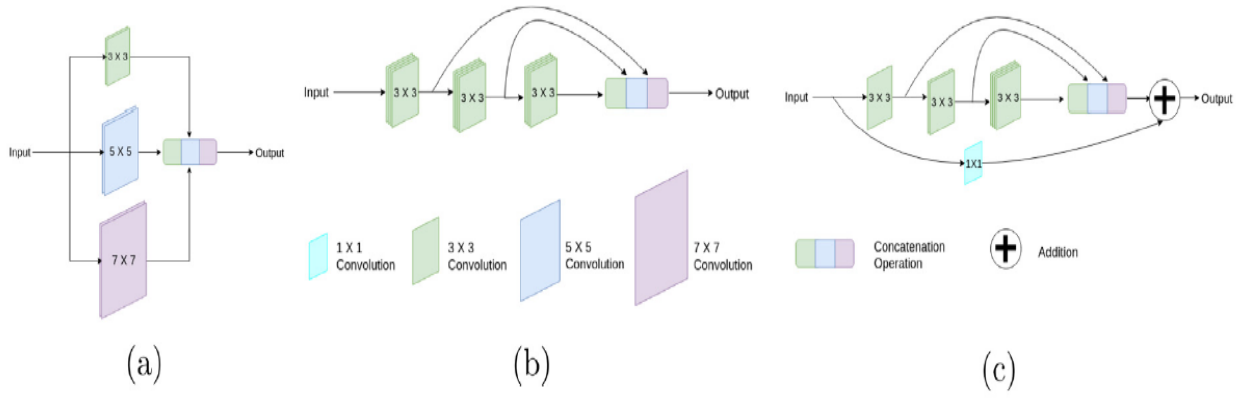
Algorithm 4: MultiResUNet.

MultiResUNet: In this project the authors take motivation from the phenomenal U-Net architecture and modify the U-Net model.

MultiResUNet: Modifications on U-Net



From 2 Conv Layers in U-Net to MultiRes Block in MultiResUNet



Developing the proposed MultiRes block from (a) to ©, © MultiRes Block

- For the sequence of two convolutional layers at each level in the original U-Net, they are replaced by the proposed MultiRes block.

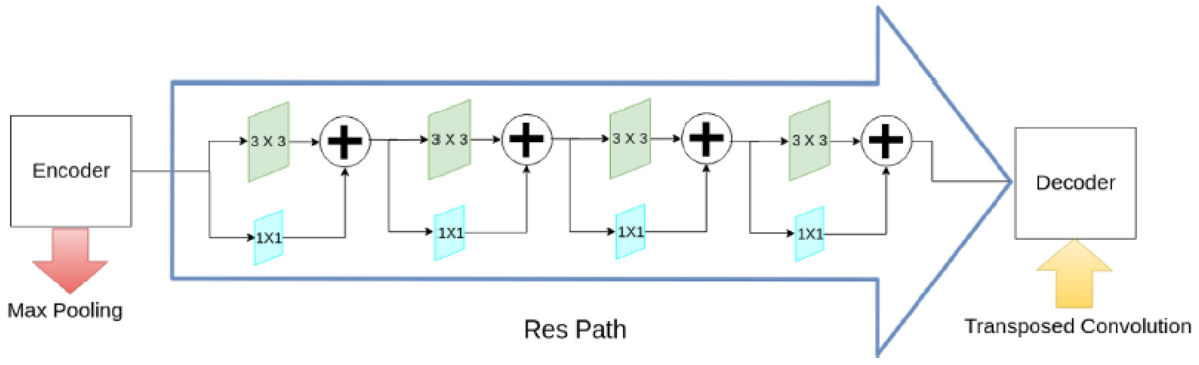
(a): First, start with a simple Inception-like block by using 3×3, 5×5 and 7×7 convolutional filters in parallel, to reconcile spatial features from different context sizes.

(b): Then, the large filter is factorized into a succession of 3 × 3 filters.

(c): Finally, **the MultiRes** block is established, by increasing the number of filters in the successive three layers gradually and adding a residual connection, along with 1×1 filters for conserving dimensions.

- This is similar to the DenseBlock in DenseNet, with the residual path originating in ResNet.

ResPath in MultiResUNet



ResPath with 1×1 and 3×3 filters

- **For the ResPath, there are 3×3 and 1×1 filters** as shown above. Number of 3×3 and 1×1 filters depends on the level inside the network, which is shown at the table below.
- Authors hypothesized that the intensity of the semantic gap between the encoder and decoder feature maps are likely to decrease.
- **These additional non-linear operations are expected to reduce the semantic gap between encoder and decoder features.**

Number of feature maps

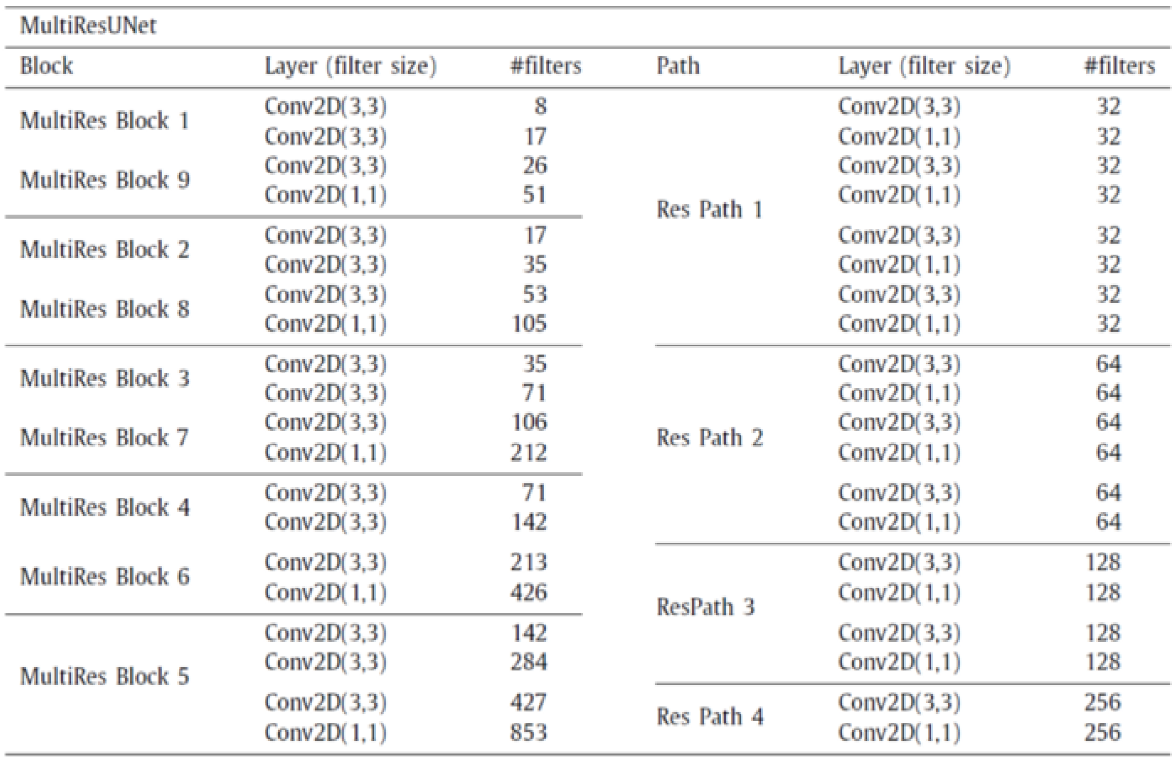
In order for fair comparison with U-Net, similar number of parameters should be maintained between two models:

$$W = \alpha \times U$$

- where U and W are the number of filters in one convolutional layer in U-Net and MultiResUNet respectively. $\alpha = 1.67$ is used. Thus, the filter numbers as shown below are multiplied by α already.

Architecture Summary

Below is the details of MultiResUNet architecture:



MultiResUNet

Block	Layer (filter size)	#filters	Path	Layer (filter size)	#filters
MultiRes Block 1	Conv2D(3,3)	8	Res Path 1	Conv2D(3,3)	32
	Conv2D(3,3)	17		Conv2D(1,1)	32
MultiRes Block 9	Conv2D(3,3)	26		Conv2D(3,3)	32
	Conv2D(1,1)	51		Conv2D(1,1)	32
MultiRes Block 2	Conv2D(3,3)	17		Conv2D(3,3)	32
	Conv2D(3,3)	35		Conv2D(1,1)	32
MultiRes Block 8	Conv2D(3,3)	53		Conv2D(3,3)	32
	Conv2D(1,1)	105		Conv2D(1,1)	32
MultiRes Block 3	Conv2D(3,3)	35	Res Path 2	Conv2D(3,3)	64
	Conv2D(3,3)	71		Conv2D(1,1)	64
MultiRes Block 7	Conv2D(3,3)	106		Conv2D(3,3)	64
	Conv2D(1,1)	212		Conv2D(1,1)	64
MultiRes Block 4	Conv2D(3,3)	71		Conv2D(3,3)	64
	Conv2D(3,3)	142		Conv2D(1,1)	64
MultiRes Block 6	Conv2D(3,3)	213	ResPath 3	Conv2D(3,3)	128
	Conv2D(1,1)	426		Conv2D(1,1)	128
MultiRes Block 5	Conv2D(3,3)	142		Conv2D(3,3)	128
	Conv2D(3,3)	284		Conv2D(1,1)	128
	Conv2D(3,3)	427	Res Path 4	Conv2D(3,3)	256
	Conv2D(1,1)	853		Conv2D(1,1)	256

Multi-Res-Unet Model Training , Performance and Result

Training:

img_size = (160, 160) # (256,192)

batch_size = 1 #2,3,4

number_classes = 2 #3,4

number_channels = 3

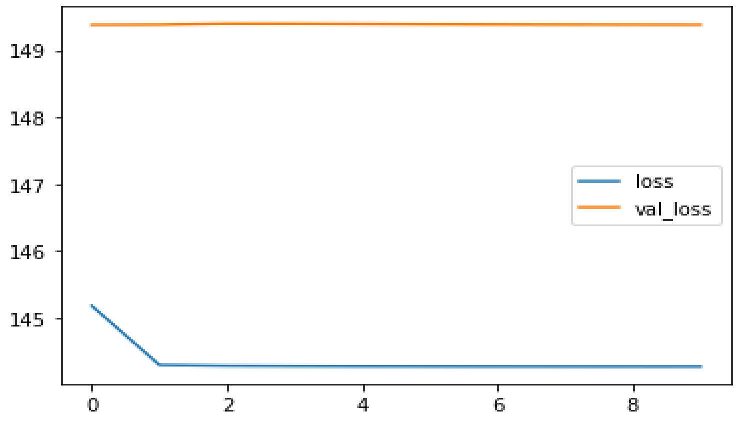
loss= categorical_crossentropy

optimizer = ADAM # RMSPROP

epochs = 10 # 30

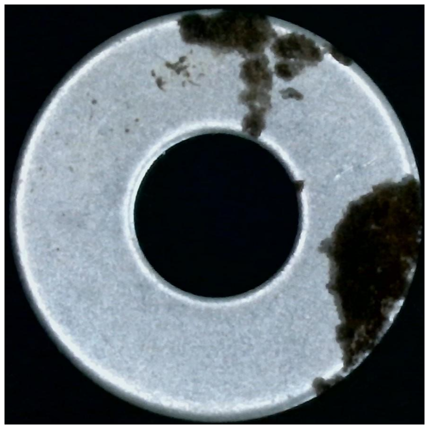
Performance : I applied different img_size, batch_size, number_classes, optimizer and epochs. Here I have attached the optimized result. Multi-Res-U-Net provides the best result among the algorithms.

Model Loss Graph: Loss started from 145.1759 after 10 epochs it finished at 144.2701.

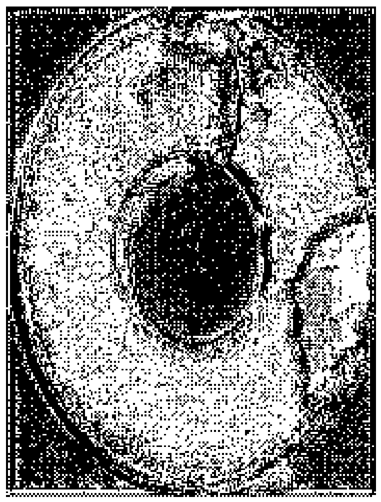


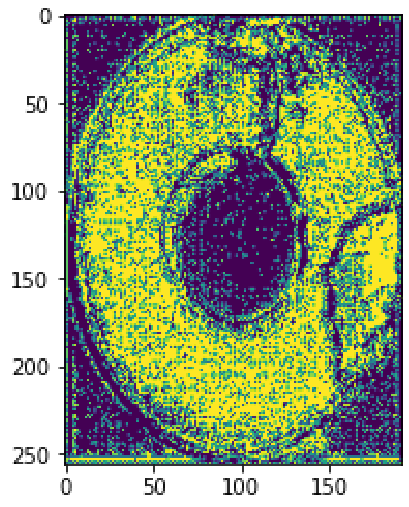
Loss and Validation Loss Graph of Multi-Res-U-Net

Model Output on Heat Map: Here I have attached the model output image when the input image size is (256,192). This image size is used in the research paper.



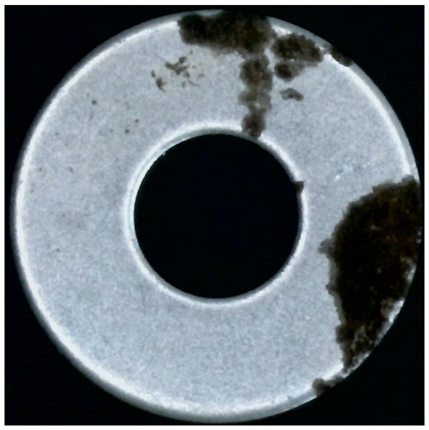
Input Image



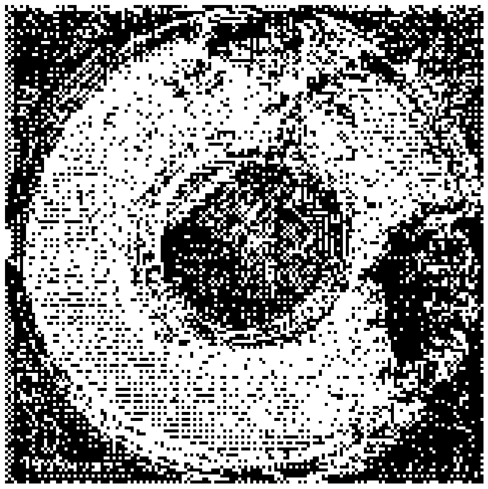


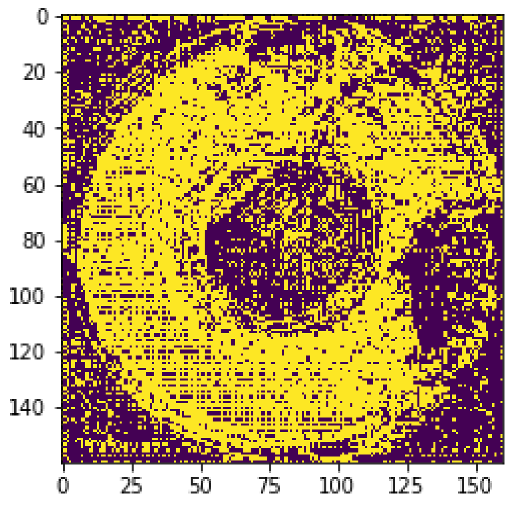
Output Image

Model Output on Heat Map: Here I have attached the model output image when the input image size is (160,160). (256,192) image size and (160,160) image size both provided better performance.



Input Image





Output Image

Limitations : First of all, data is the most essential point of a good model. Here the model was trained by very little data. The Multi-Res-U-Net model could learn little features from less data.

This model gives good results on most defected images.

Recommendation : If the number of data increases, the model can learn more during the training period. The model needs fine tuning in the parameters such as batch size, optimizer, learning rate, and data augmentation. Then there is a high possibility the performance of the Multi-Res-U-Net model will increase.

Reference : https://arxiv.org/pdf/1902.04049.pdf

Model Comparison

First I researched the VAE algorithm and then other Segmentation algorithms which are following the Encoder and Decoder idea. I listed 4 algorithms which were suitable for the "Washer DataSet".

After implementing training those 4 models. Among those models 2 models provided the best result.

The models are:

1. MultiResUNet
2. U-Net.