**Abstract**Chart

Description automatically generated

*In this assignment, the task was to use image segmentation to extract a foreground mask from a given image. Using the U-Net auto-encoder architecture with pre-trained MobileNetV2 weights, this was achieved with significantly good results subject to the proper tuning of the model parameters.*

# Introduction

Image segmentation is a very useful task in computer vision applications. It has uses in object detection, as well as object tracking in videos, to name a few. While many traditional computer vision algorithms exist to tackle this problem (Mean Shift, SIOX, Graph/Grab Cuts), there is still a promise that deep learning algorithms hold in this area, especially with transfer learning based on tried and tested model architectures. The advantage of this is that with the right training data, a model can be trained to segment images without the need for image seeds.

## Method

The data provided for the task included images from everyday life, and corresponding seed files (with varying amounts of seeding), as well as ground truth foreground segmentation masks. Corresponding to each image.

Ignoring the seed files, and using the original images as input, and the ground truth segmentations as output labels, I was able to train a convolutional neural network following the U-Net [1] architecture with pre-trained weights for the encoder section from MobileNetV2 [2], a tried and tested lightweight feature extractor.

## U-Net Architecture

The U-Net architecture (Figure 1) is a symmetric network of convolutional blocks shaped like a ‘U’. During training, it extracts a compressed representation of the image, and in the upsampled version is measured against a label image, in this case, the ground truth segmentation image/mask.

As training proceeds and loss is minimized, it learns to generate a mask representation for a given image.

## MobileNetV2

Considering that we had very limited training data, it was prudent to employ pre-trained weights from a model that could already extract relevant features to some extent from real-world images. MobileNetV2 was a prime candidate, since it is lightweight, but has very good performance.

## Training

The model was built and trained with TensorFlow (2.X) and the weights for the encoder were downloaded before training began.

After tuning parameters to avoid overfitting, the model was trained for 20-30 epochs, and the model summary is shown in Figure 2.

Table

Description automatically generated

**Figure 2 - Model Summary**

# Results

It took about 10 minutes to train for 30 on an 8-core Intel i7 CPU, and the results of training are compared with the ground truth segmentations in Figures 3-5.



**Figure 3 - Sample input image (a desktop computer)**

Icon

Description automatically generated

**Figure 4 - Sample mask for the desktop computer**

A picture containing dark

Description automatically generated

**Figure 5 - Foreground segment generated by the algorithm**

# Conclusion

While not perfect, the generated mask will return a very high Intersection of Union (IOU) with the original mask. The model learns very well, and very quickly how to correctly generate foreground masks for images.

The only limiting factor is the amount of training data, but even then, with jittered training and image augmentation, we can see better results with limited data on account of the efficiency produced through transfer learning.

Deep learning thus proves to be an efficient method for tackling this problem, especially since this task was completed without the use of seed images.

# References

1. U-Net**:** Convolutional Networks for Biomedical Image Segmentation, Ronnenberger et al, https://arxiv.org/abs/1505.04597
2. MobileNetV2: Inverted Residuals and Linear Bottlenecks, Sandler et al, <https://arxiv.org/abs/1801.04381>

# Appendix

To run the program on multiple images in a single directory, replace the input image with the path to the directory and add the flag “--multiple” without quotes to the command in the terminal.

Eg. **python hw04.py input\_images/ seed\_file.jpg output\_folder/ --multiple**

You can still run the program on single images using **python hw04.py input\_image.jpg seed\_file.jpg output\_folder/**

Note however that seed files will be ignored by the program.