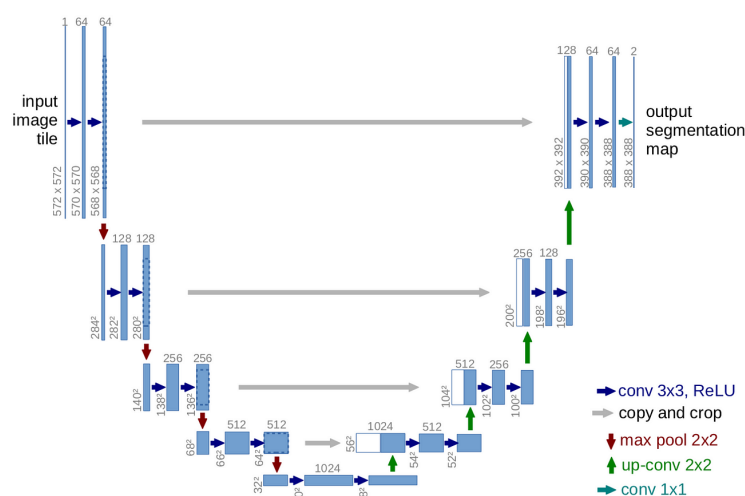


Image Segmentation Project Report

Introduction

This notebook summarizes the project as I used multiple notebooks and files for this project. One of the pivotal tasks in the field of computer vision is image segmentation. Image segmentation involves dividing an image into multiple segments or regions based on specific attributes. This process enables computers to understand the content of images, leading to a wide array of applications across diverse industries. Image segmentation plays a crucial role in various applications, including object recognition, medical image analysis, autonomous vehicles, and more.



UNets are a convolutional neural networks specifically designed for semantic segmentation tasks. UNets combine downsampling convolution blocks and upsampling convolution blocks with strategic skip connections to create the distinct U-shaped architecture. This structure allows the

model to perform accurate segmentation tasks along with many other applications. UNets are also used widely in image generation tasks like stable diffusion or generators in GANs.

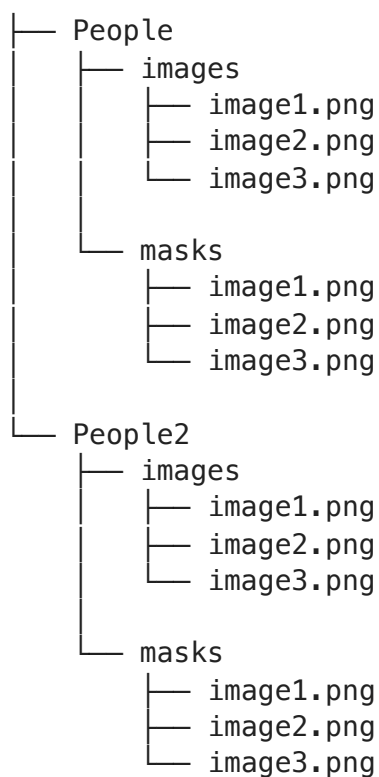
In this project, I will be tackling the problem of segmenting people within images using UNets. My goal is to learn how to build Unets from scratch using tensorflow and apply it to image segmentation.

Data

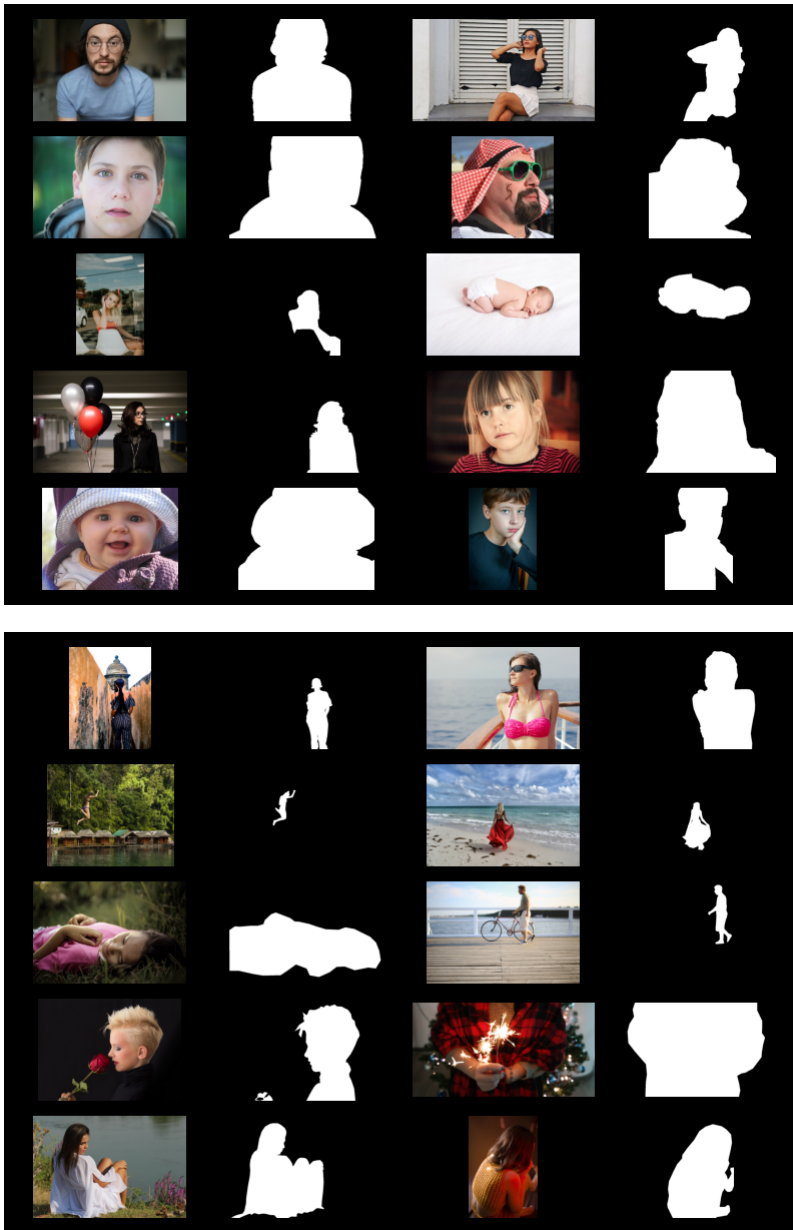
The data for this project is taken from two sources, "[People - Segmentation](#)" on Kaggle and "[Persons](#)". The datasets contain images of people and json annotations with the mask details. The exploratory data analysis and preprocessing code is located in `dataset_exploration.ipynb`.

EDA / Preprocessing

The datasets contain different notation for the masks so they will need to be processed separately. The masks were generated for each image and the dataset folders were reorganized as follows. Any images with an inconsistent json file format was removed from the dataset at this point.

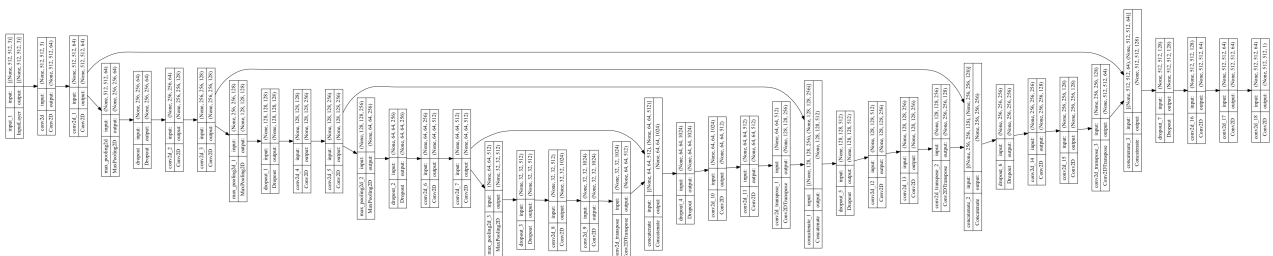


Samples from the datasets are shown below. The masks match the images very well so the processing went well. The sizes of the images are not normalized at this stage.



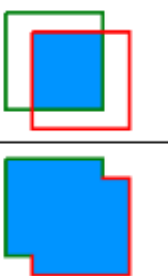
Model Architecture

The UNet architecture is implemented in the `utils.py` file using tensorflow. The model takes inputs of shape `(256,256,3)` and outputs a mask with shape `(256,256,1)`. The mask is binary so we only need one channel for the output image. The model architecture is shown below and the U-shaped architecture is visible as well.

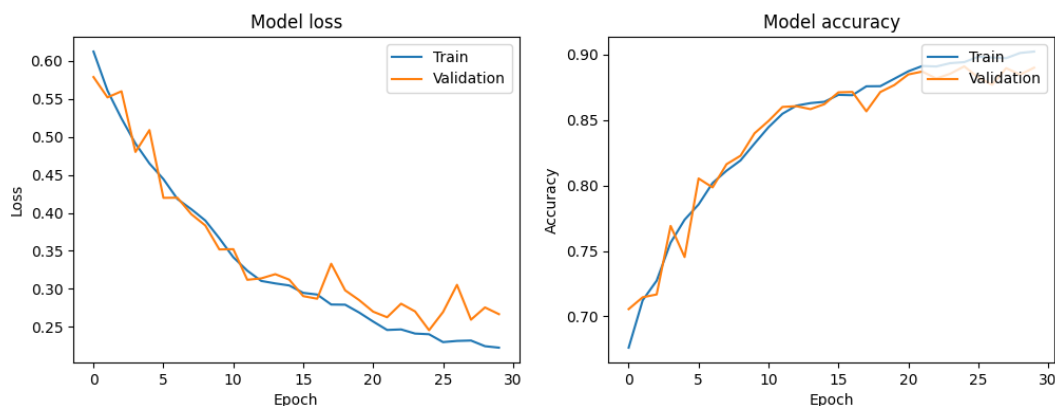


Training

The model training and data loading was done in `train.ipynb`. Since it is a binary classification task, I will use accuracy for the loss function, but there are other valid options as well. One common loss function for image segmentation is IoU which takes the intersection of the prediction and the true value and divides it by the union. The combined dataset was split into a training and validation set with 2643 images for the training set and 660 images for the validation set. The images were also augmented for training to artificially increase the dataset size.

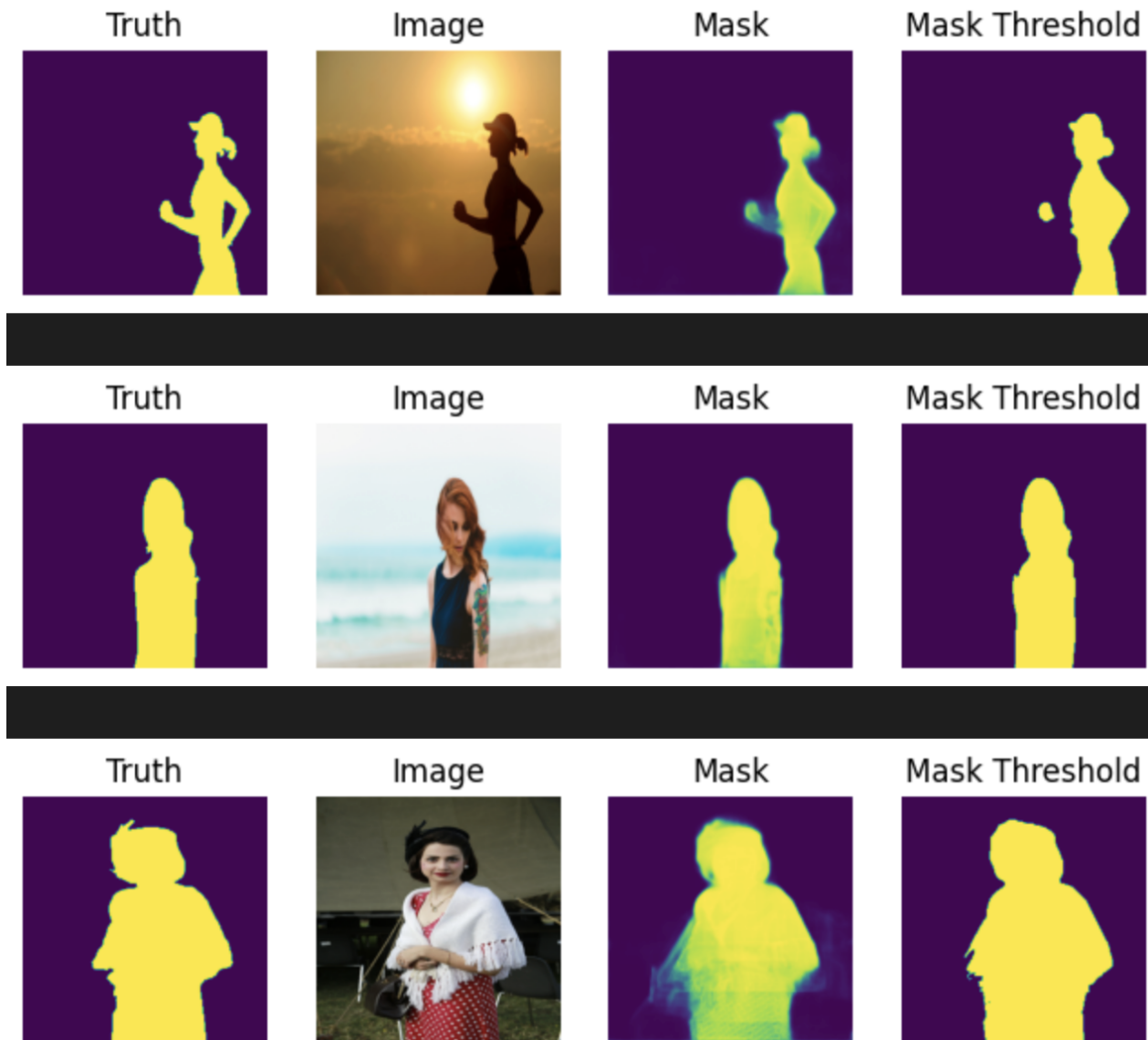
$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of intersection}}{\text{area of union}}$$


The optimizer used as ADAM and the early stopping and learning rate scheduling was also used. Training terminated after 30 epochs due to early stopping and the model achieved an training accuracy of 0.8942 and validation accuracy of 0.8909. The learning curves shown below are promising as the training loss and evaluation loss decrease at a good rate throughout the training. The model does not show any signs of severe overfitting and seems to generalize well even to the validation data.



Results

Below are some sample predictions on the validation set. The model outputs probabilities for each pixel so we need to convert to a mask using a threshold. With the examples shown below, the model is working extremely well. The predicted masks are almost identical to the true masks.



Application

I have applied the model to webcam footage in `main.py`. This file will take images from the webcam remove the background and insert a new background. The goal was to have the model run in real time to remove the background like some conferencing services like Zoom. The model works as intended but the it is too slow to render in real time on my laptop. If it is run on a machine with a GPU the results would likely be much better.

Conclusion

In summary, I was able to successfully train a well performing person segmenation model in this project. The UNet architecture for generating masks was proven to be effective as demonstrated with the training/validation accuracy and the example outputs.

However, the deployment in real time background replacement faced some issues with computation time. The model may work well for a computer with a GPU but is not realistic for less

powerful machines. If this project is revisited, I will look further into different architectures and methods that enable real time background removal that work with machines with less computation power.