COMPUTER VISION

COMPUTER VISION PROJECT MID EVALUATION REPORT

DEEP IMAGE MATTING USING CONVOLUTIONAL NEURAL NETWORKS

MNS Subramanyam(20161190) Team Number: 30

Sowrya Deep Thayi (20161029) Team Name: Mission Impossible

Appari Lalith(20161038)

INTRODUCTION

Problem statement and motivation

- Image matting involves extracting foreground object from an image.
- The objective of our project is to perform image matting using deep convolutional networks.

Advantages on solving this problem

- It is a key technology in image editing and film production and effective natural image matting methods can greatly improve current professional workflows.

Limitations of previous methods used for image matting process

- Previous algorithms have poor performance when an image has similar foreground and background colors or complicated textures.
- The main reasons are prior methods
    - Only use low-level features.
    - Lack high-level context.
- These approaches rely largely on color as the distinguishing feature (often along with the spatial position of the pixels), making them incredibly sensitive to situations where the foreground and background color distributions overlap which is a drawback.
- Even other deep learning methods are highly dependent on color-dependent propagation methods.

Advantages of this method

- To overcome these problems a new approach which considers structural and semantic features more into account for matting.
- Neural networks is capable of capturing such high order features and applying them to compute improved matting results.

Limitations of this method

- Since this method uses VGG16(deep neural network) which takes heavy amount of time for training.
- The network architecture weights themselves are quite large (concerning disk/bandwidth).

METHOD OVERVIEW

Pipeline

1. Preprocessing the dataset
2. Encoder Decoder Network
3. Refining the alpha matte

Preprocessing the dataset

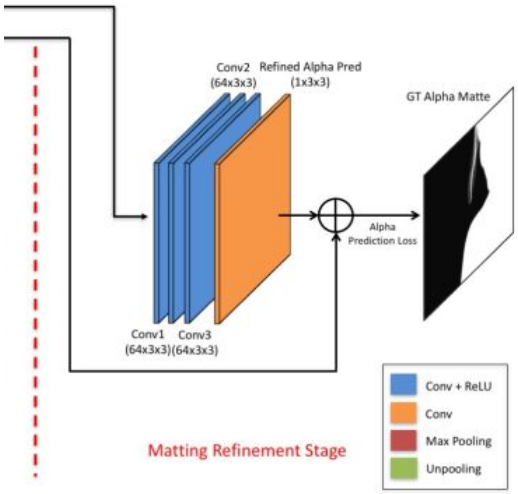
- A dataset consisting of 431 foreground images for training, 20 foreground images and 20 trimaps for testing were provided by the author of the paper upon our request.
- To train our matting network, we create a dataset by composing objects from foreground images onto new backgrounds for which we used the VOCO 2007 dataset.
- To test our matting network, we create a dataset by composing 20 foreground test images with 100 background images of VOCO dataset giving 1000 test images.
- We have considered 43,100 training examples, which are produced using 431 foreground images and 100 background images.
- We were also given the alpha mattes for each foreground image for training and testing dataset.

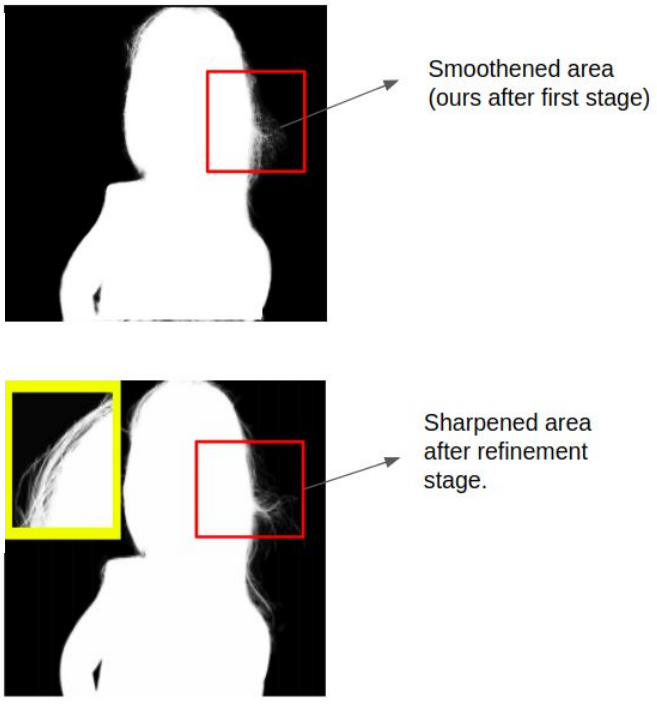
ENCODER-DECODER NETWORK

- This is a deep convolutional encoder-decoder network that takes a image and the corresponding trimap as inputs and predict the alpha matte of the image.

REFINING THE ALPHA MATTE

- This part involves a small convolutional network that refines the alpha matte predictions of the first network to have more accurate alpha values and sharper edges.



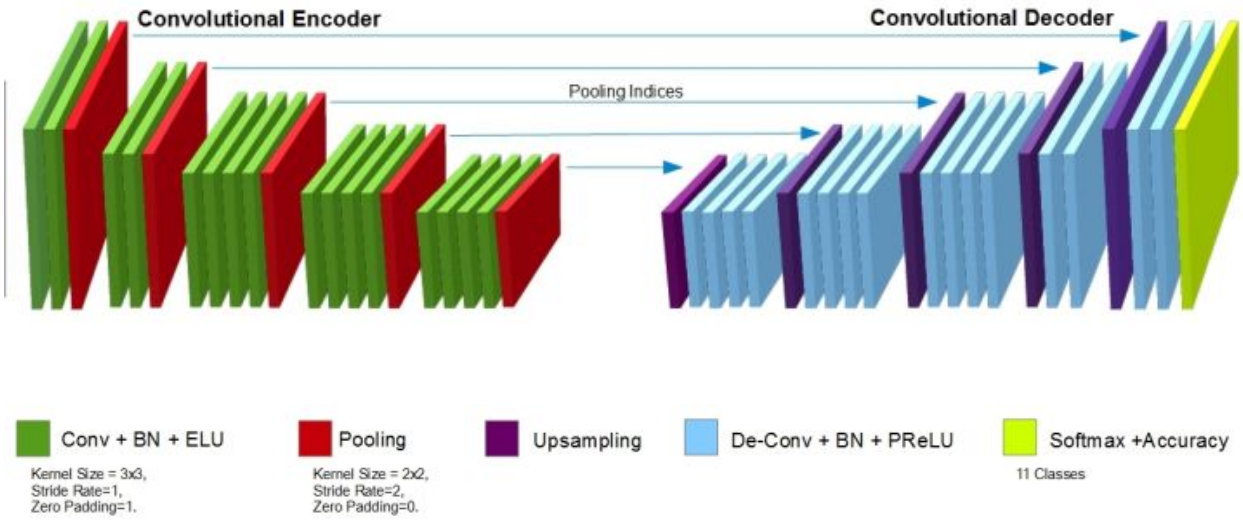


WORK DONE SO FAR

IMAGE DATASET PREPROCESSING

- First we created a training dataset consisting of 431K images by composing 431 foreground images and 1000 background images obtained from VOCO dataset.
- SImilarly we created a test dataset of 1000 images by composing 20 foreground images and 50 background images.
- Now, we have 431K training images, 1K testing images and ground truth alpha matte and trimaps for each training and testing images.

ENCODER DECODER NETWORK



- Takes the input image (3-channel) and trimap and concatenates to form a 4-channel dimensional input.
- Encoder involves downsampling the image. (14 conv layers and 5-max pooling layers).
- Decoder involves upsampling the image. (6 conv layers and 5-unpooling layers).
- The decoder network has a small structure to reduce the parameters and amount of computation required.

- This network leverages two losses, the first loss is called the alpha-prediction loss, which is the absolute difference between ground-truth alpha values and the predicted alpha values at each pixel.

$$\mathcal{L}_{\alpha}^{i} = \sqrt{(\alpha_p^i - \alpha_g^i)^2 + \epsilon^2}, \quad \alpha_p^i, \alpha_g^i \in [0, 1].$$

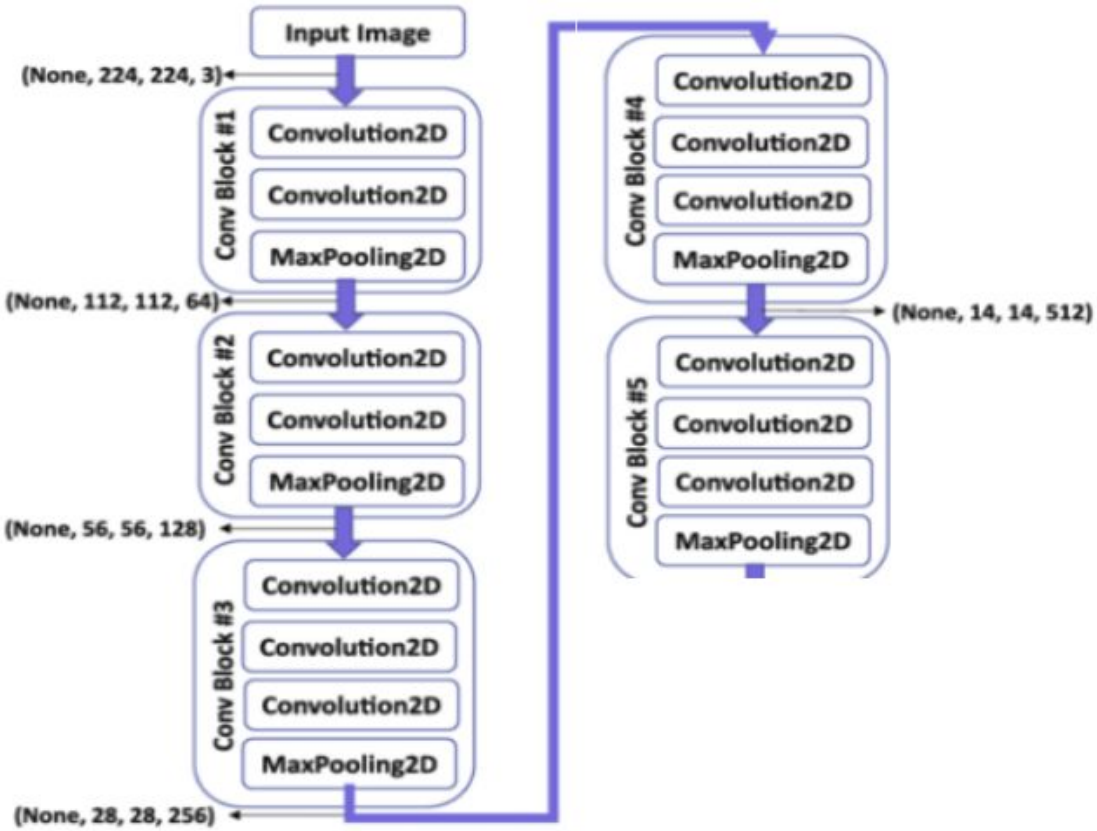
- The next loss is the composition loss which constrains the network to follow the compositional operation, leading to more accurate alpha predictions.

$$\mathcal{L}_{c}^{i} = \sqrt{(c_p^i - c_g^i)^2 + \epsilon^2}.$$

- The overall loss is a weighted combination of these two losses given by

$$\mathcal{L}_{overall} = w_l \cdot \mathcal{L}_{\alpha} + (1 - w_l) \cdot \mathcal{L}_{c},$$

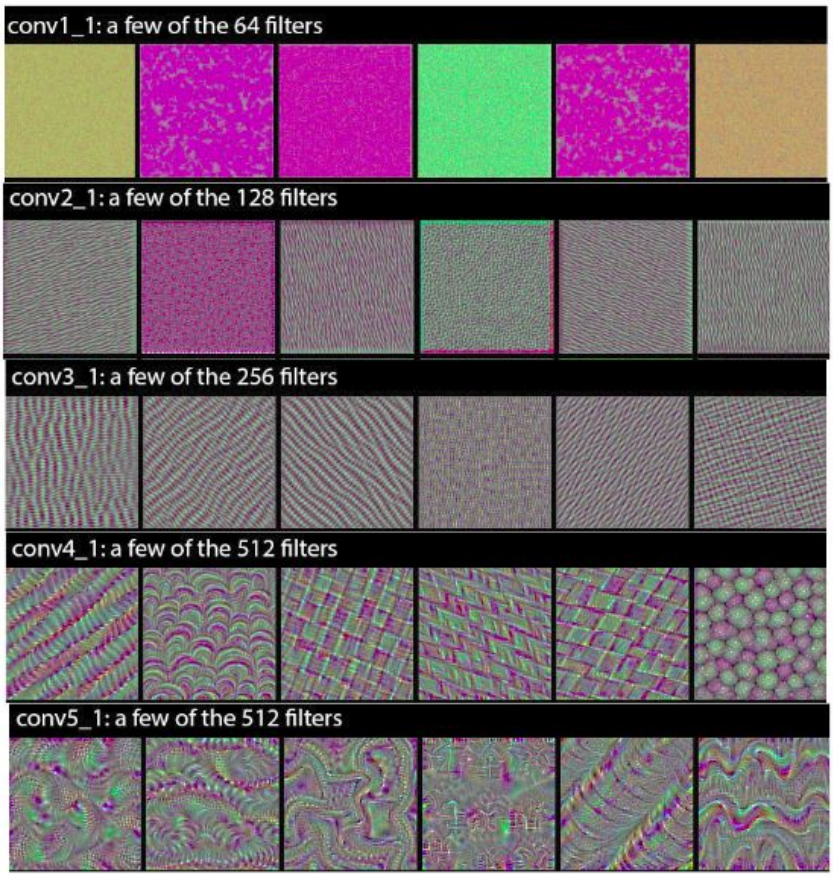
ENCODER NEURAL NETWORK



- A VGG16 network is used for the deep encoder-decoder network. The image is passed through a stack of convolutional (conv.) layers, where the filters were used. The kernel size(convolution filter), stride, padding, of each convolution layer are as in the below image:

```
EDNet(
  (conv1_1): Conv2d(4, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv1_2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2_1): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv2_2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3_1): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3_2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv3_3): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4_1): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4_2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv4_3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5_1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5_2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv5_3): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (conv6_1): Conv2d(512, 512, kernel_size=(1, 1), stride=(1, 1))
  (deconv6_1): Conv2d(512, 512, kernel_size=(1, 1), stride=(1, 1))
  (deconv5_1): Conv2d(512, 512, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (deconv4_1): Conv2d(512, 256, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (deconv3_1): Conv2d(256, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (deconv2_1): Conv2d(128, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (deconv1_1): Conv2d(64, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
  (deconv1): Conv2d(64, 1, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
)
```

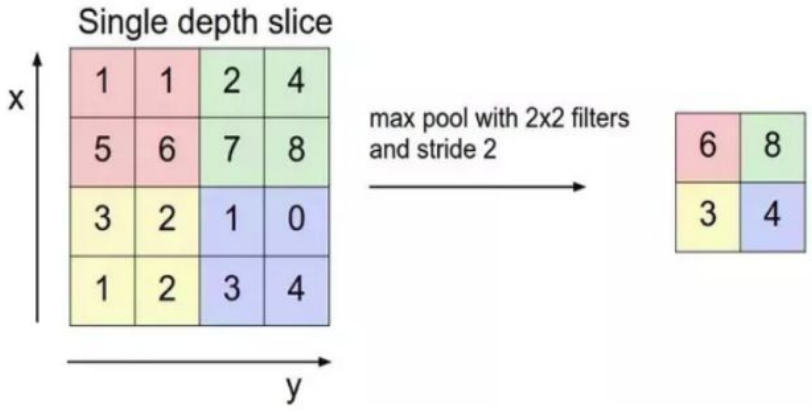
- Different layer filters in encoder neural network



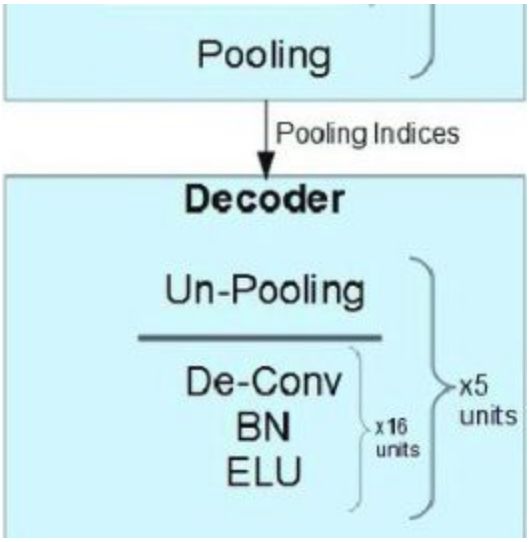
- Above are some of the visualizations of the filters used in encoder network.
- The first layers basically just encode direction and color. These direction and color filters then get combined into basic grid and spot textures.
- The textures gradually get combined into increasingly complex patterns.
- One of the key observations is that a lot of these filters are identical but rotated at different angles, this helps in achieving rotational invariance

MAX POOLING

- The objective is to down-sample an input representation (image) reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.
- This also helps to prevent overfitting and providing an abstracted version of the original image.
- This also reduces the computational cost, since now we have to deal with less number of parameters.
- Max pooling is done by applying a *max filter* to (usually) non-overlapping subregions of the initial representation.
- Pictorial representation:



DECODER NEURAL NETWORK



- The decoder CNN is used to map the low resolution features (extracted from pooling layers) to the final feature-maps of high resolution.
- The decoder model that is composed of upsampling, de-convolution, ReLU activation units. The decoder upsamples the encoder's convolution layer indices produced from pooling layers. Each decoder upsamples the activations generated by the corresponding encoder.
- The encoder offers low-resolution feature mapping for pixel-wise classification. The feature maps produced through the convolution layer are sparse, those later convolved using the decoder filters to generate detailed feature map.

MODEL ANALYSIS

- The training of the model took approximately 2-3 days with 30 epochs. The model was trained on 431K images for 30 epochs.
- At present the MSE test error is around 0.02 for 1000 images from test dataset.
- Some of the results obtained on test images are shown in the next section.

RESULTS

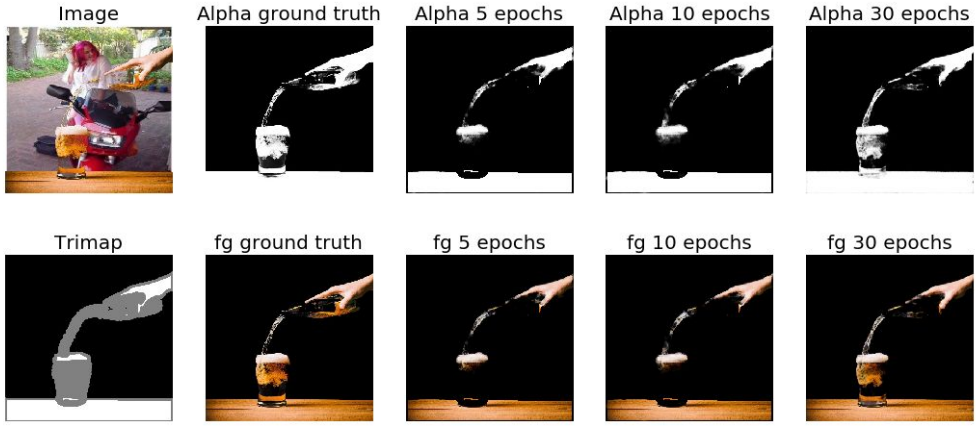


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

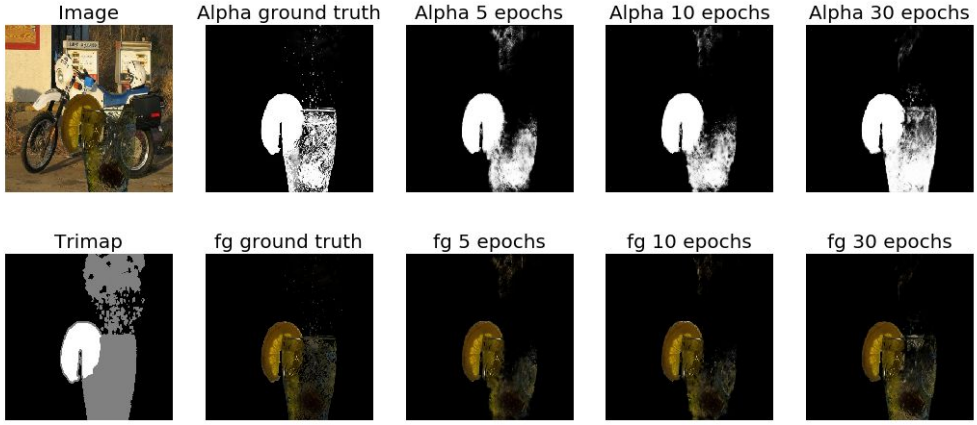


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

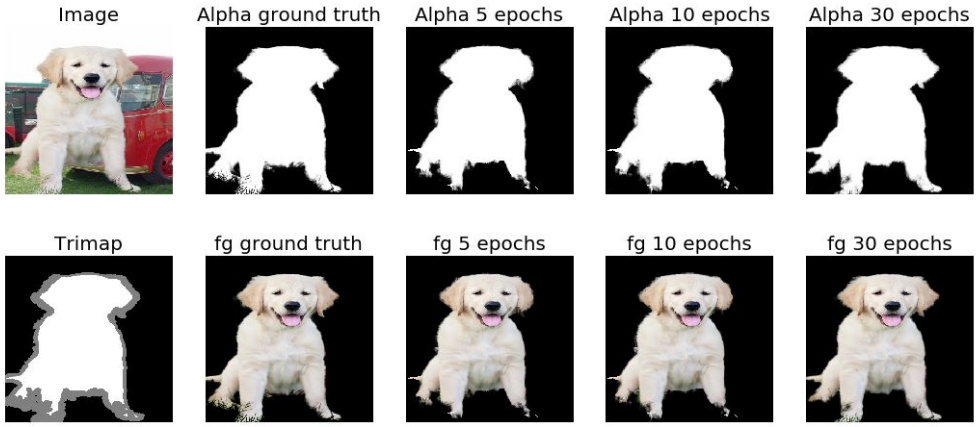


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

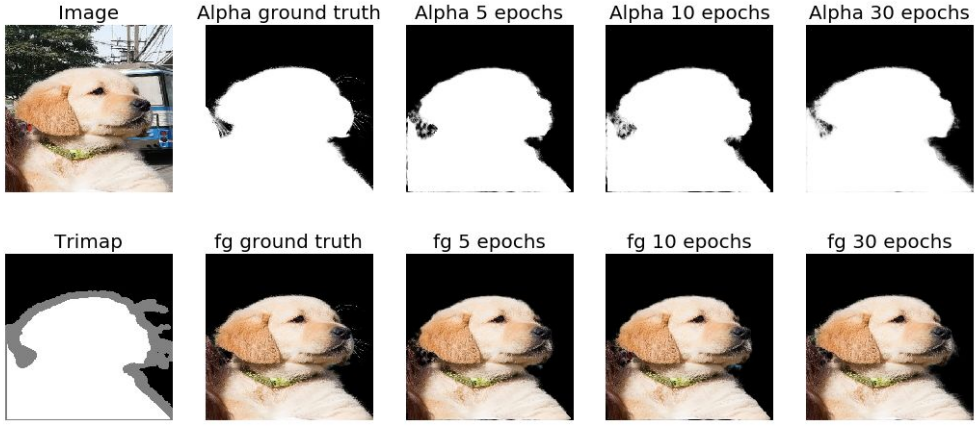


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

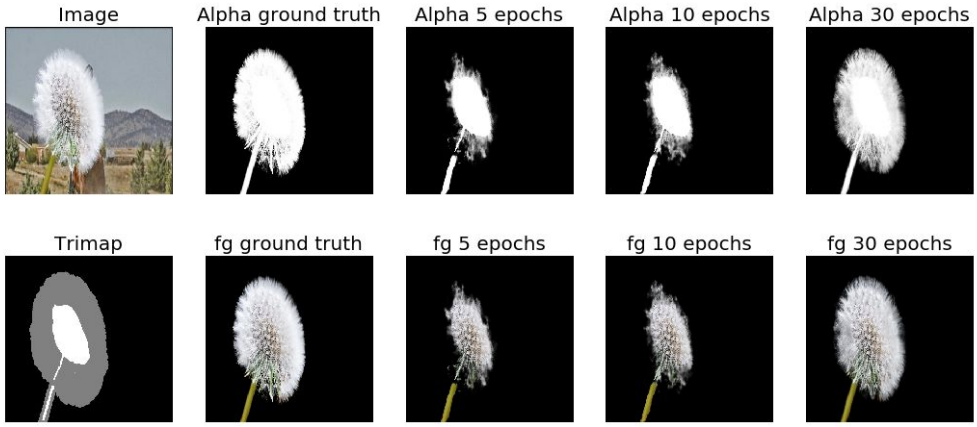


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

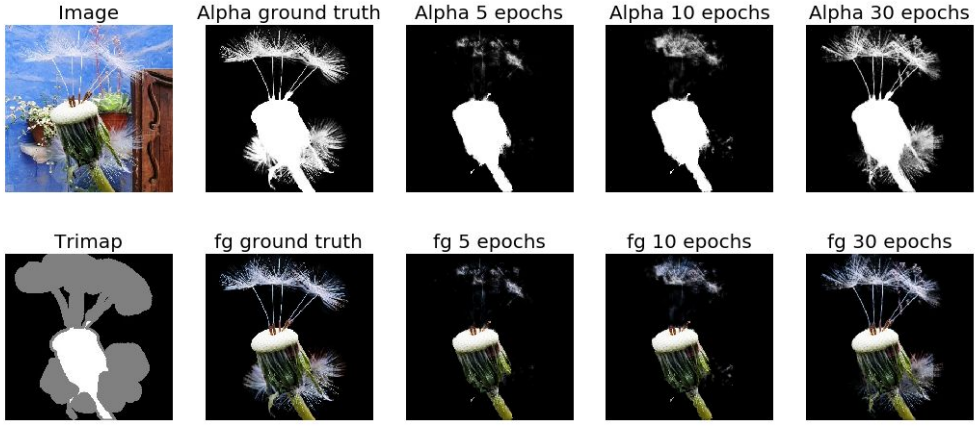


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

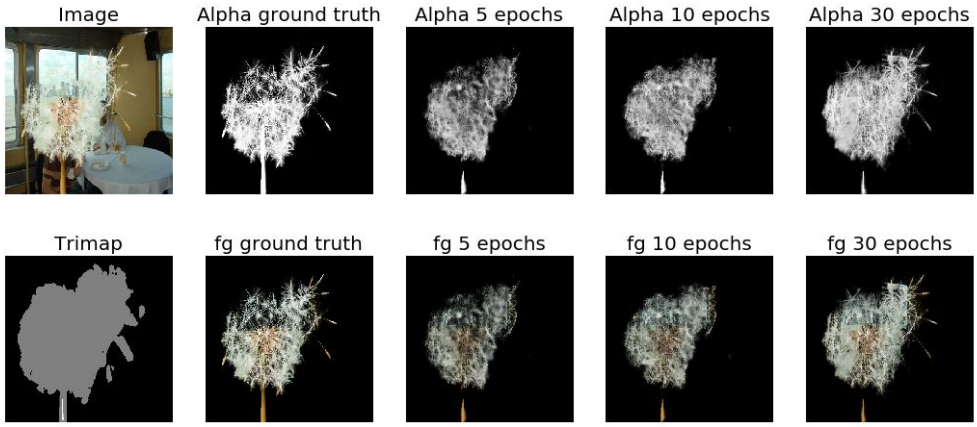


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

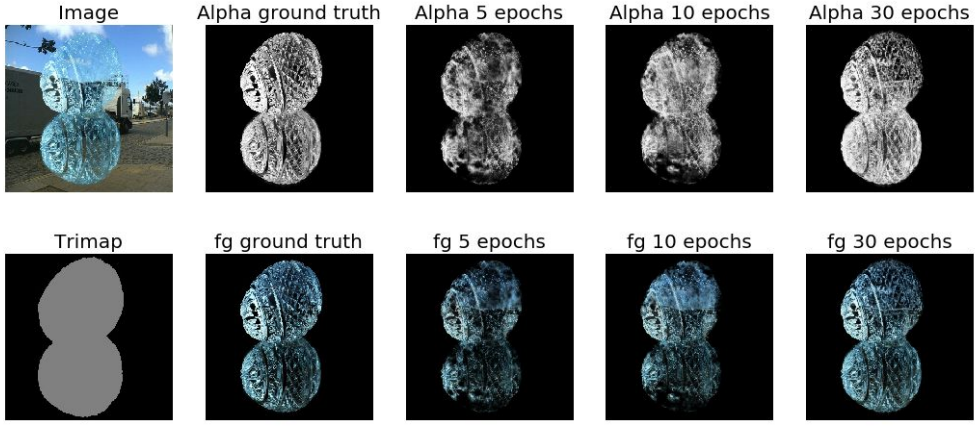


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

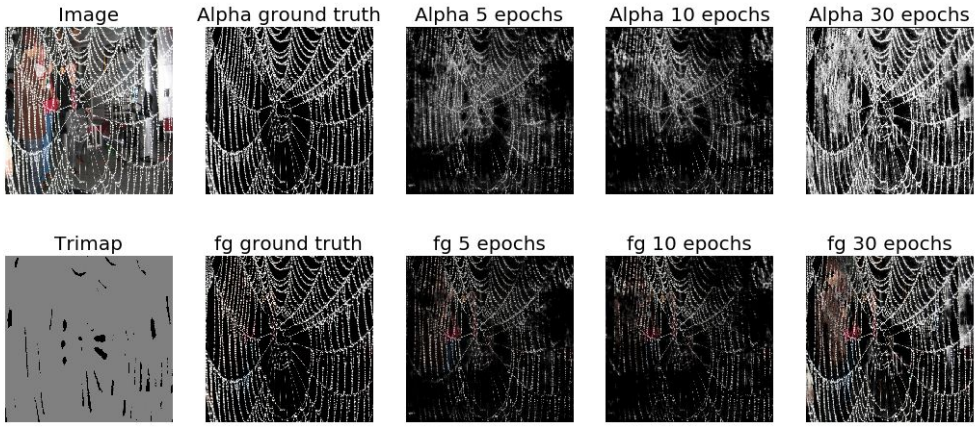


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

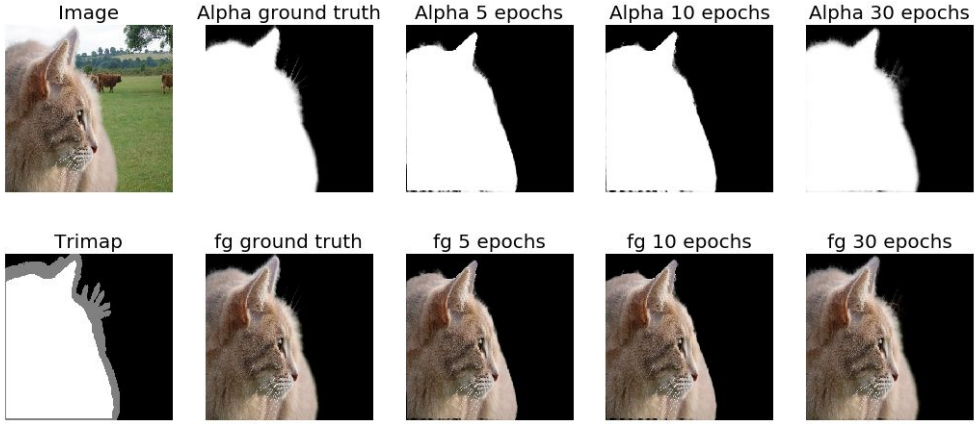


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

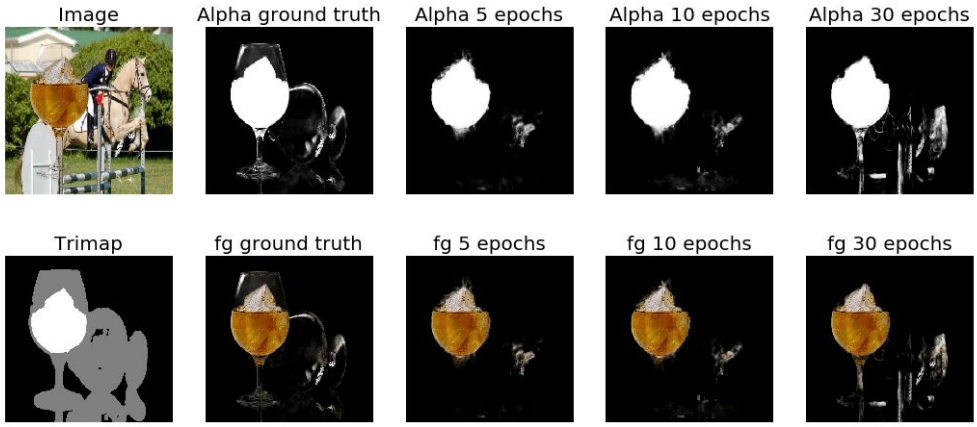


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

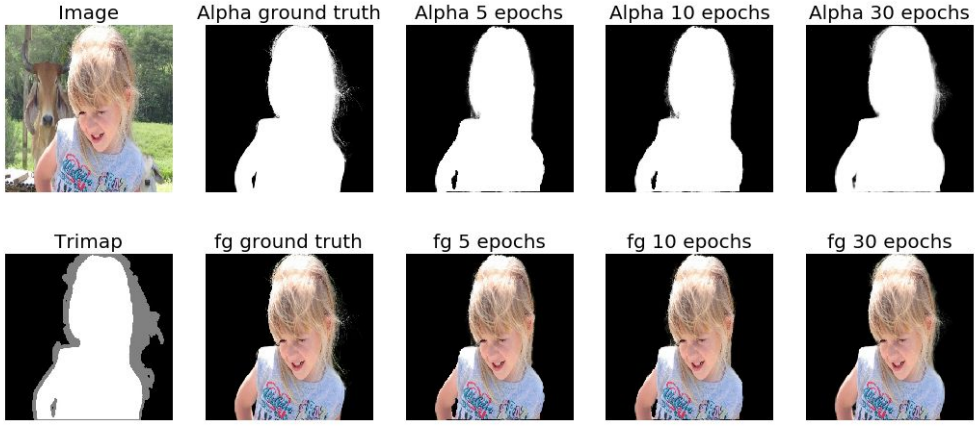


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

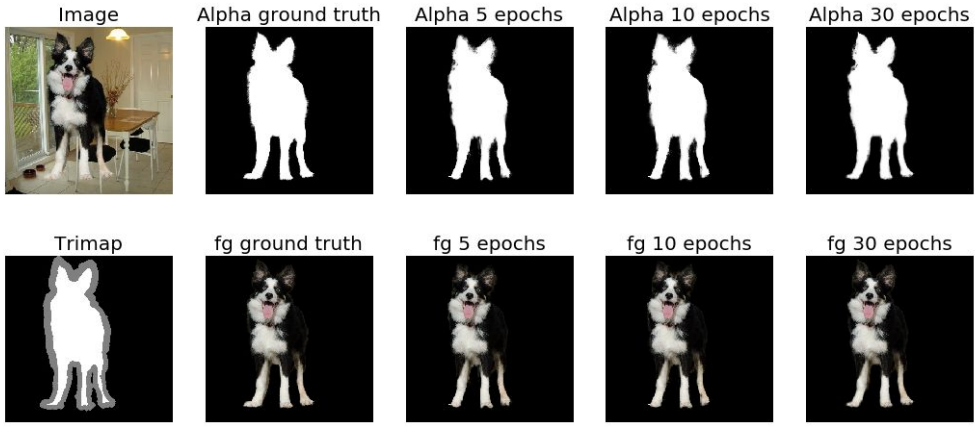


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

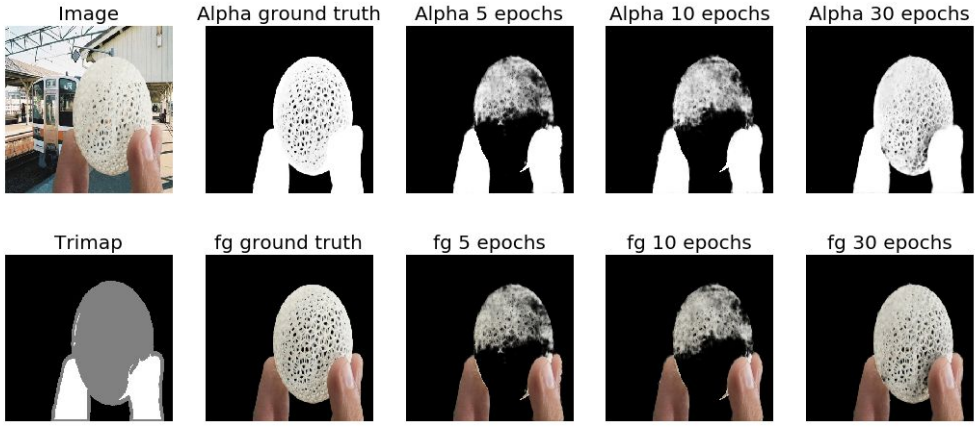


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

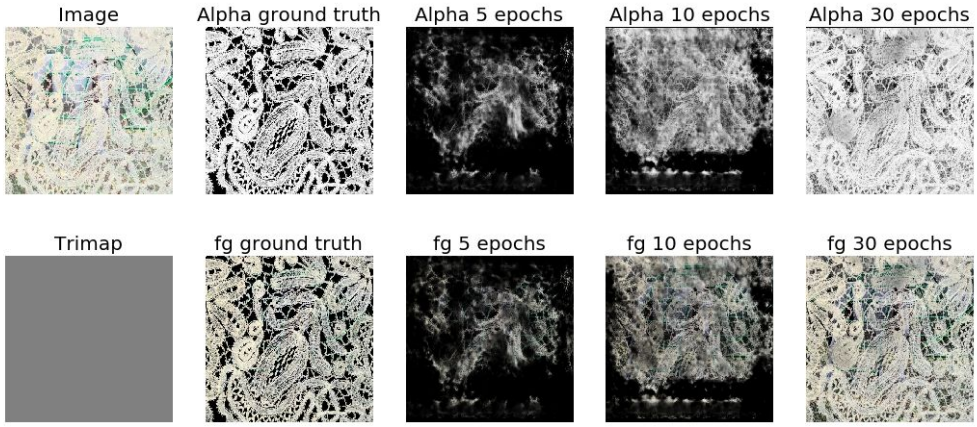


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

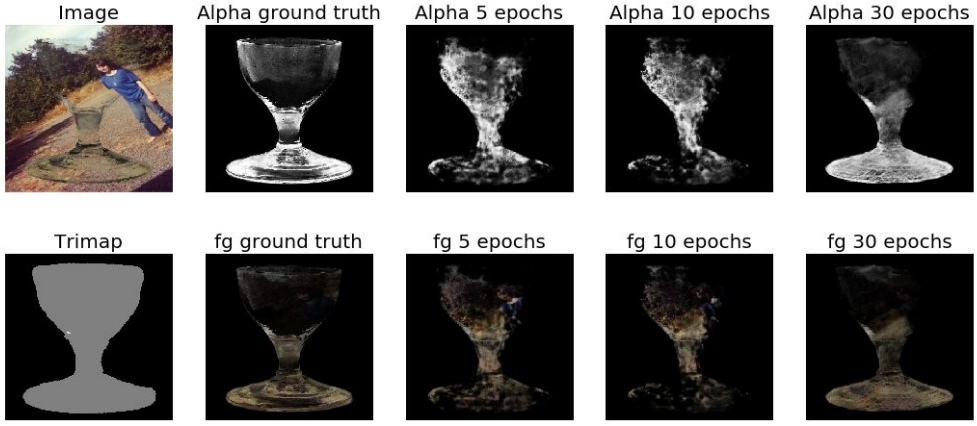


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

MILESTONES LEFT AND FUTURE WORK

- **Train the Neural Network with 40 epochs**
    - Currently we trained the neural network with 30 epochs.
- **Refinement network training and Fine-tuning the overall network once again**
    - Though the alpha predictions from the first part of the current network are much better, because of the encoder-decoder structure, the results are sometimes overly smooth.
    - Therefore, we extend our network to further refine the results from the first part. This extended network usually predicts more accurate alpha mattes and sharper edges.
- **Novelty and improving results if possible**
    - After completing the refinement stage and fine tuning we would like to focus on optimizing the network at any stage, if possible and also to improve results by changing network structure etc.