COMPUTER VISION PROJECT MID EVALUATION REPORT

DEEP IMAGE MATTING USING CONVOLUTIONAL NEURAL NETWORKS

MNS Subramanyam(20161190)

Sowrya Deep Thayi (20161029)

Appari Lalith(20161038)

Team Number: 30

Team Name: Mission Impossible

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

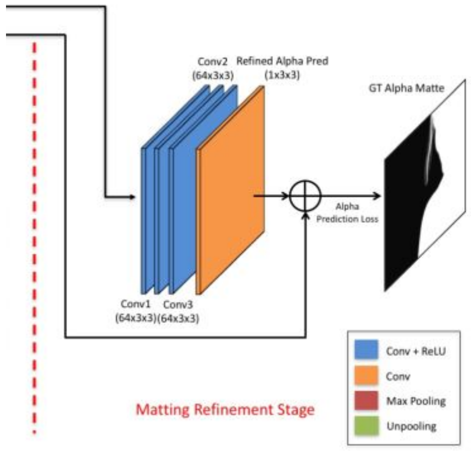
- A dataset consisting of 431 foreground images for training, 20 foreground images and 20 trimaps for testing were provided by the author of the paper upon our request.
- To train our matting network, we create a dataset by composing objects from foreground images onto new backgrounds for which we used 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.



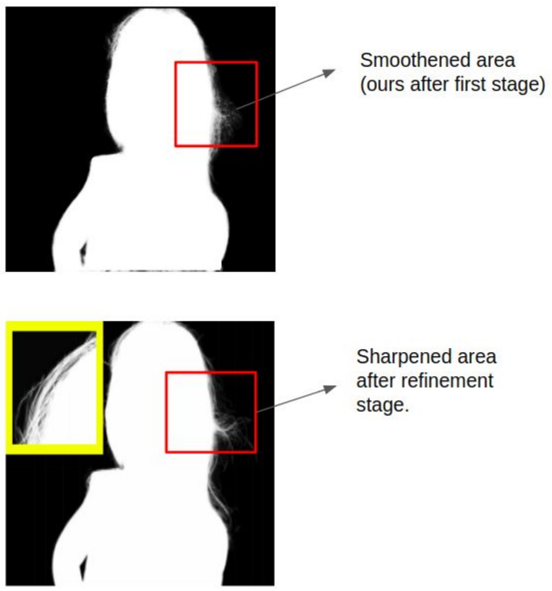
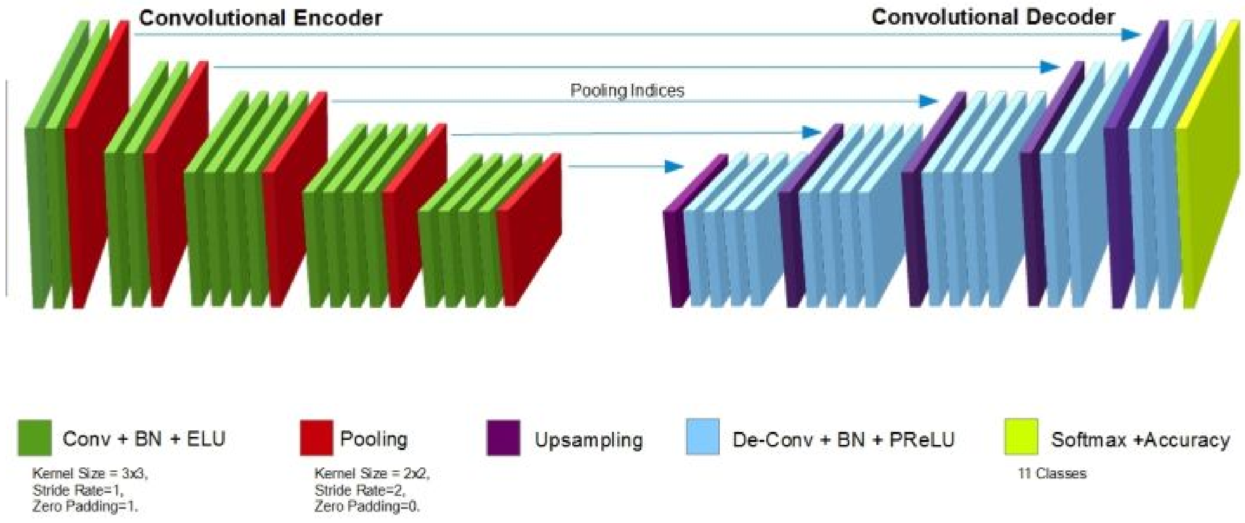


IMAGE DATASET PREPROCESSING

- 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.

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

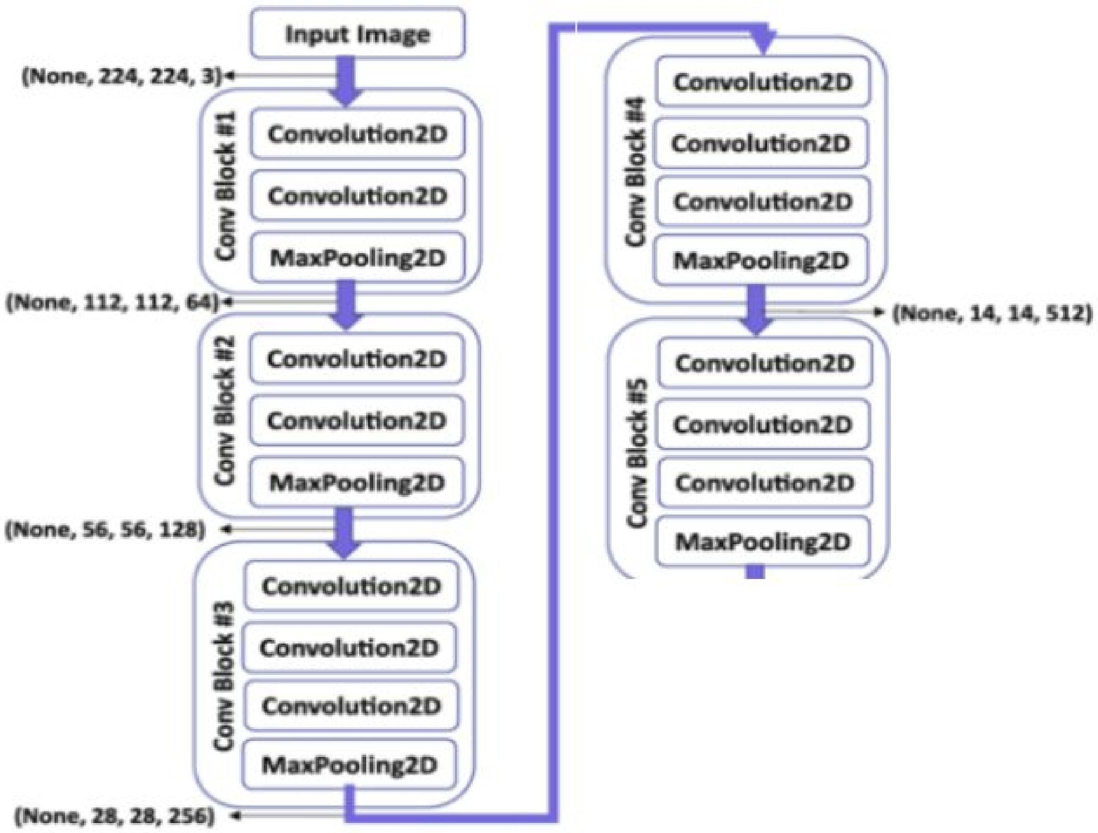
- The next loss is the composition loss which constrains the network to follow the compositional operation, leading to more accurate alpha predictions.

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

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

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

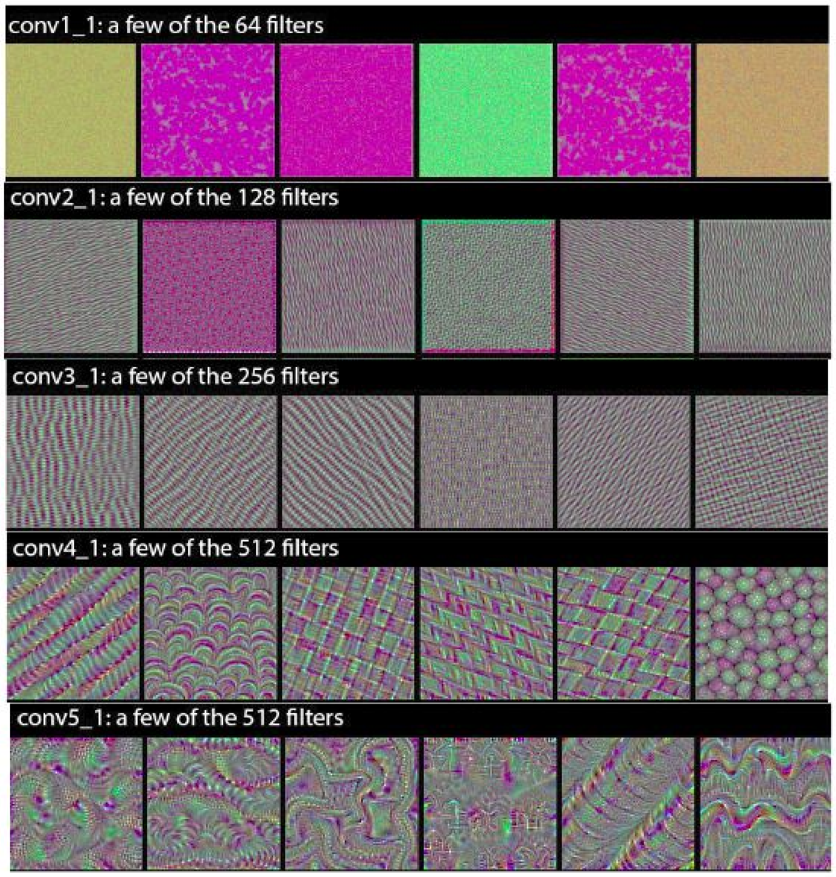
ENCODER NEURAL NETWORK



- 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))
)
```

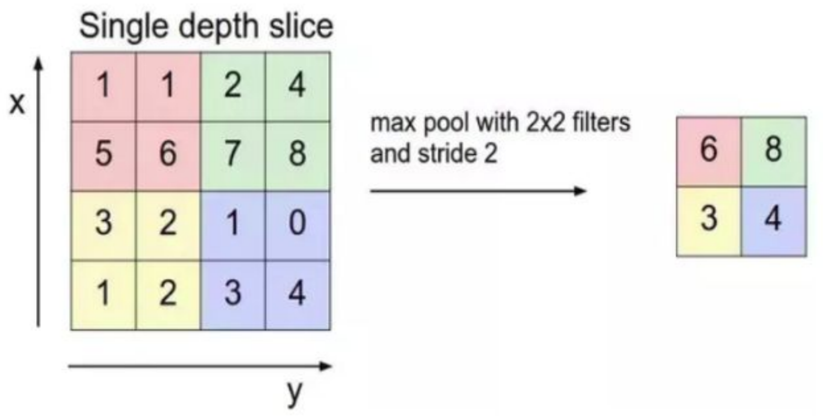
- Different layer filters in encoder neural network


conv1_1: a few of the 64 filters
conv2_1: a few of the 128 filters
conv3_1: a few of the 256 filters
conv4_1: a few of the 512 filters
conv5_1: a few of the 512 filters

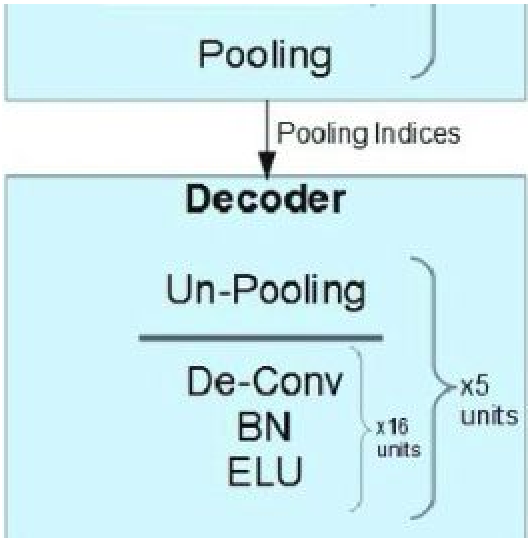
- Above are some of the visualizations of the filters used in encoder network.
- The first layers basically just encode direction and color. These direction and color filters then get combined into basic grid and spot textures.
- The textures gradually get combined into increasingly complex patterns.
- 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

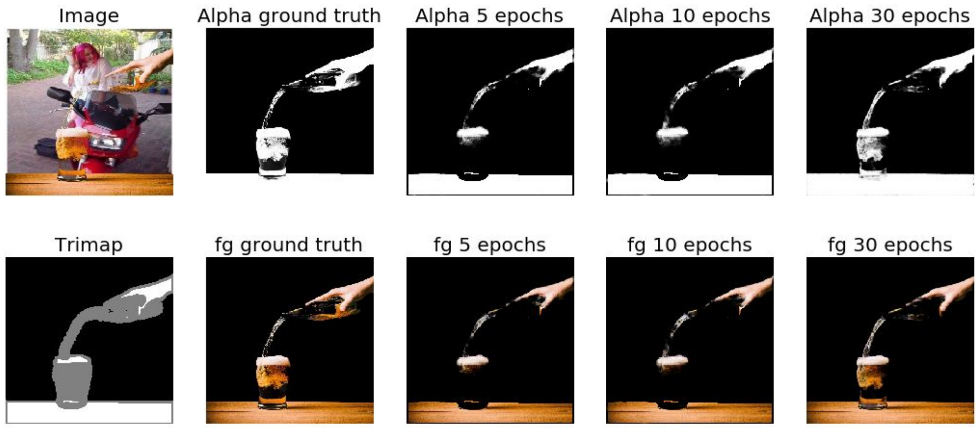


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

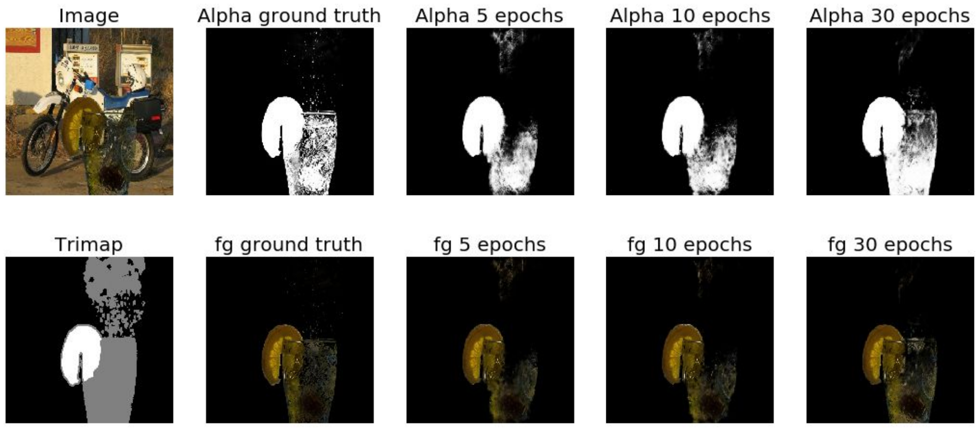


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

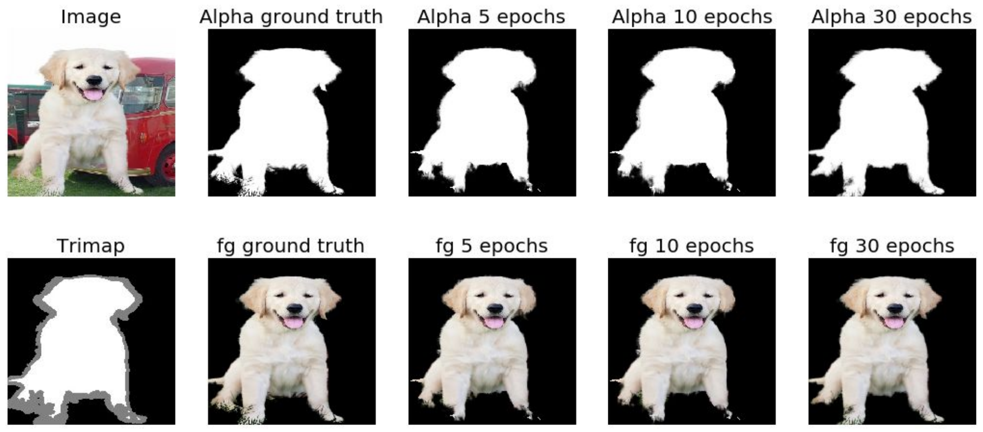


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

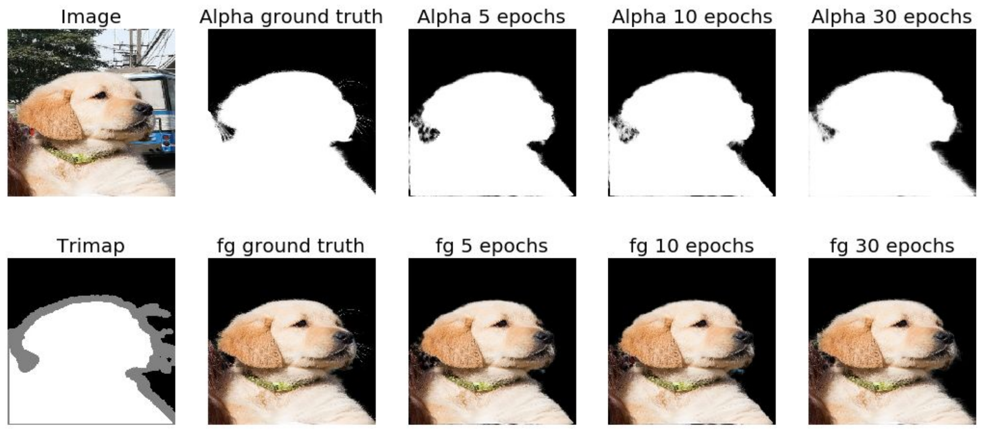


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

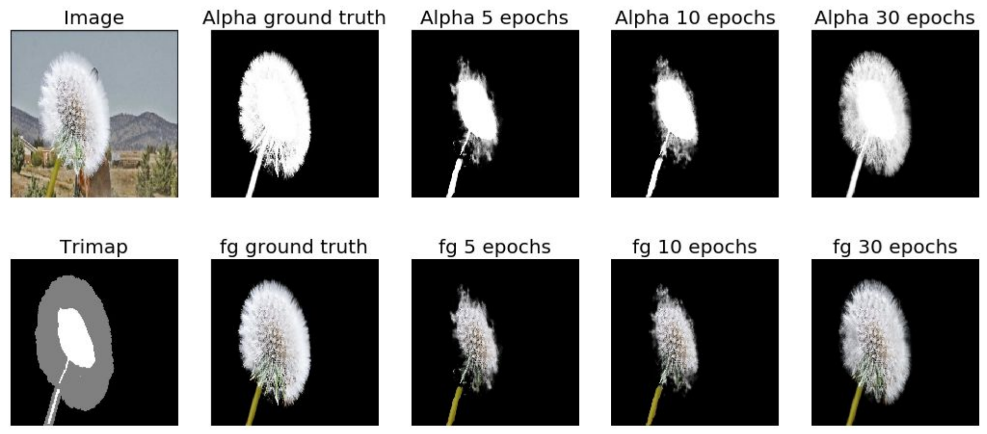


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

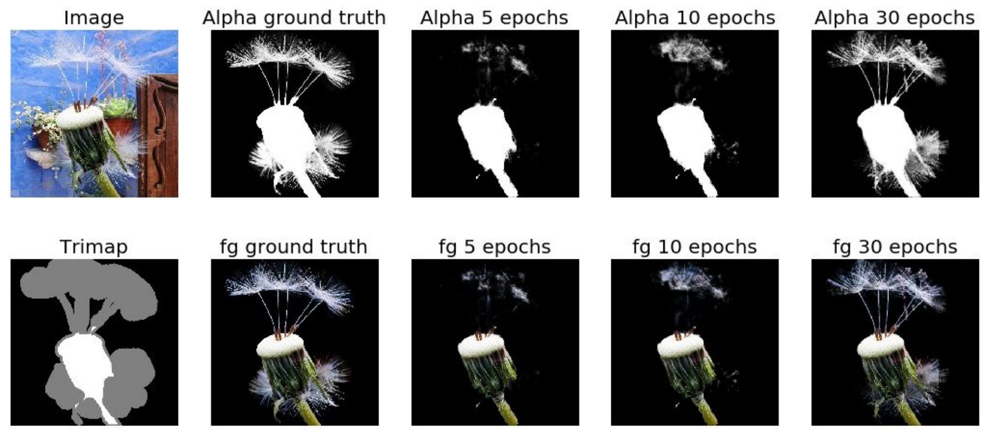


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

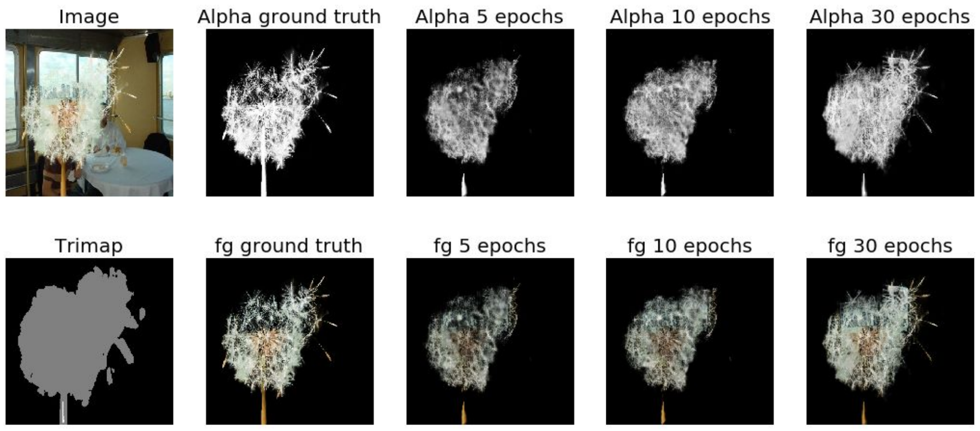


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

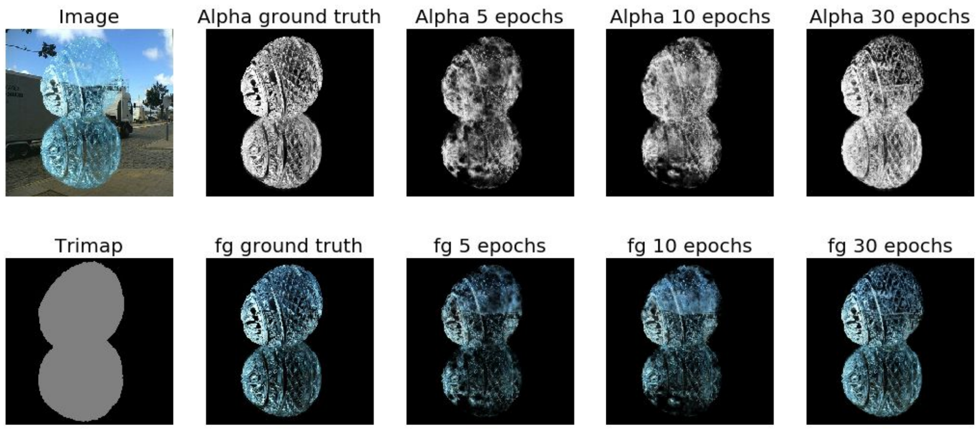


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

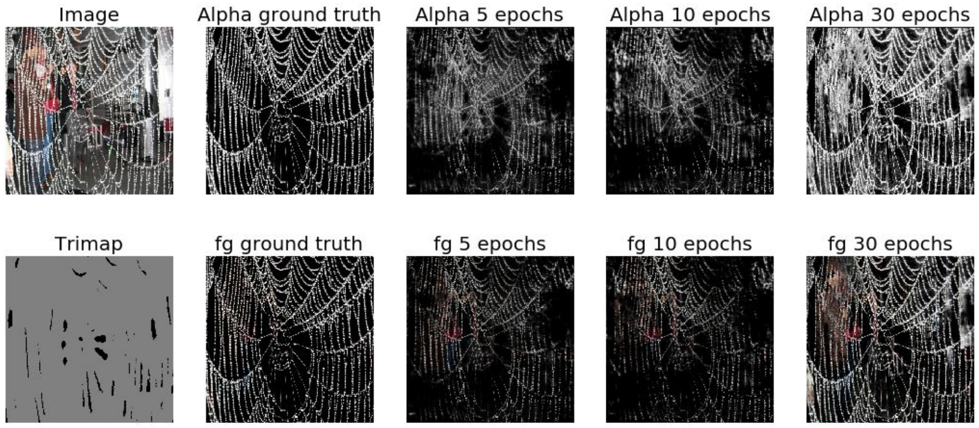


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

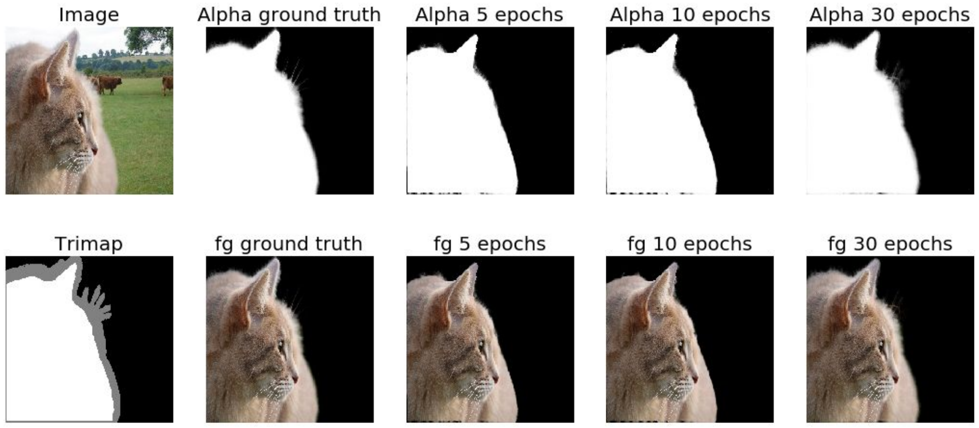


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

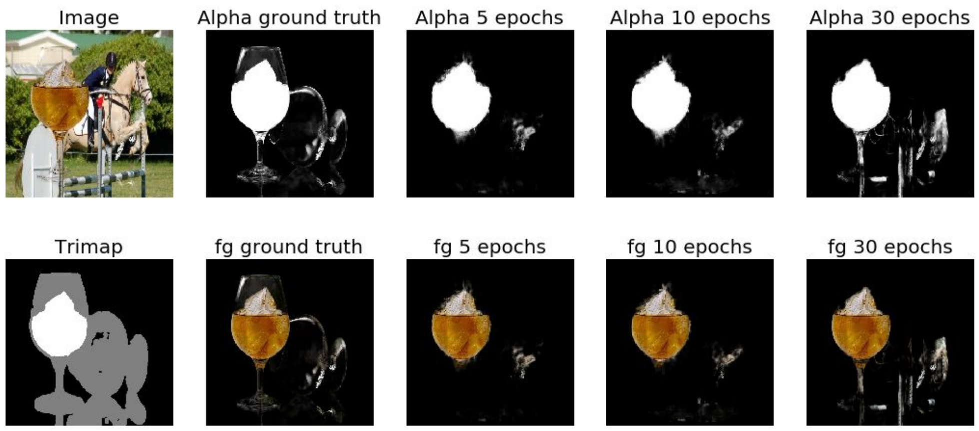


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

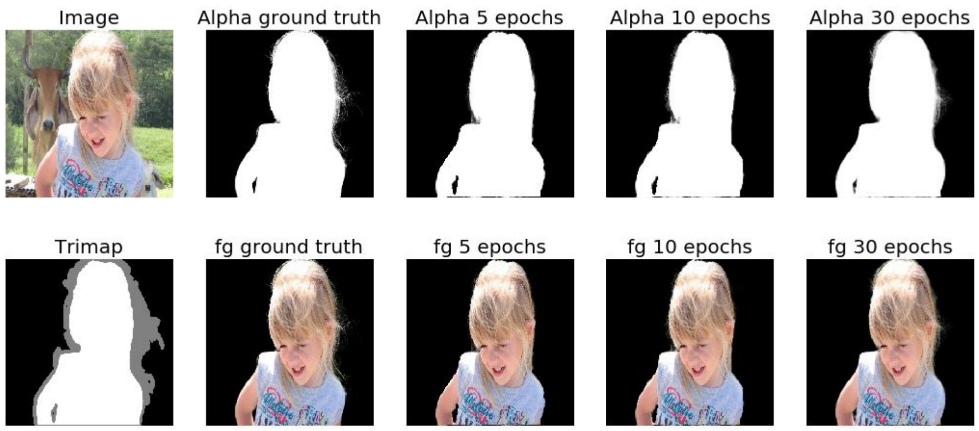


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

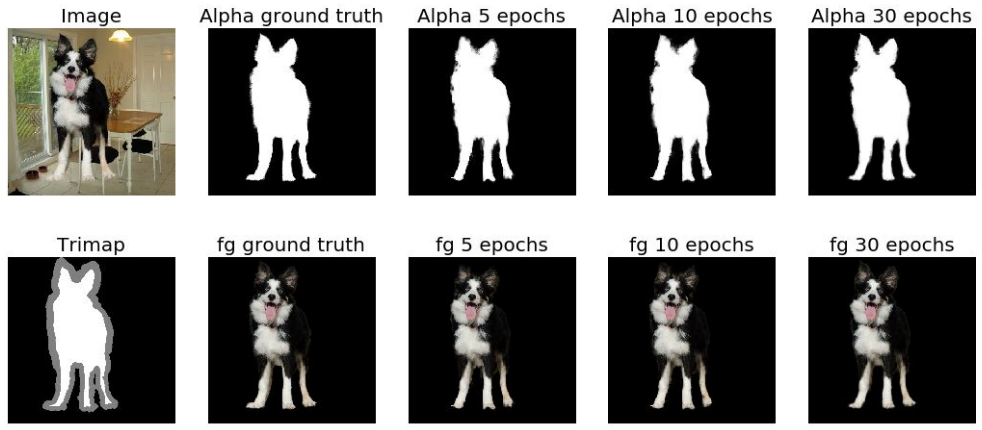


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

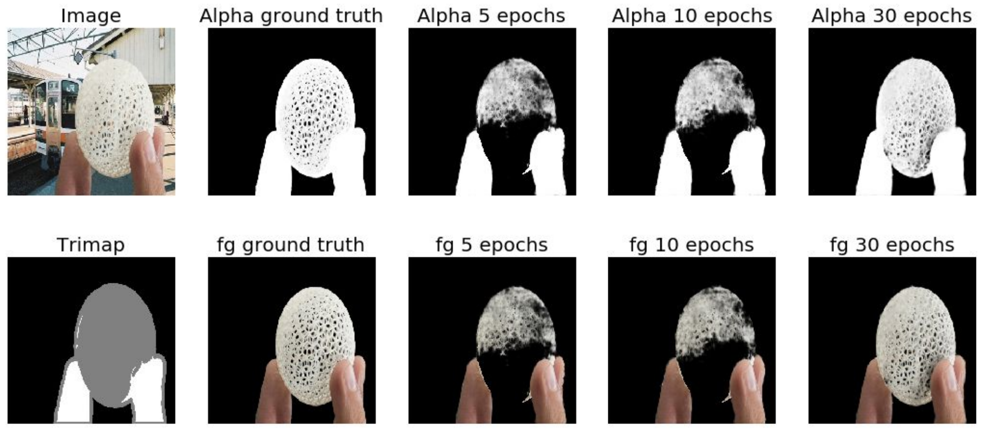


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

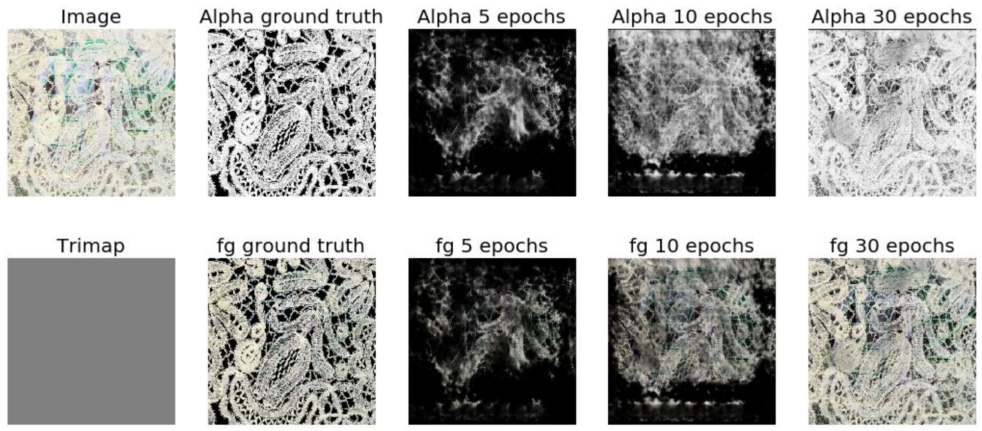


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

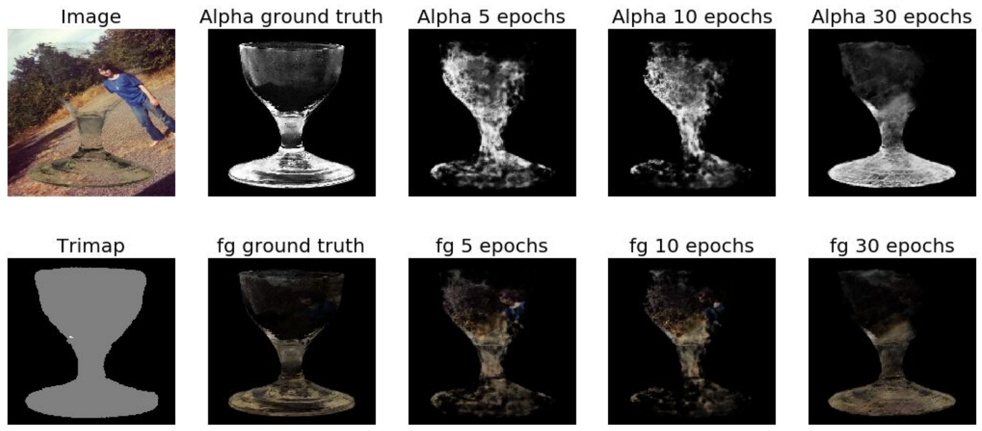
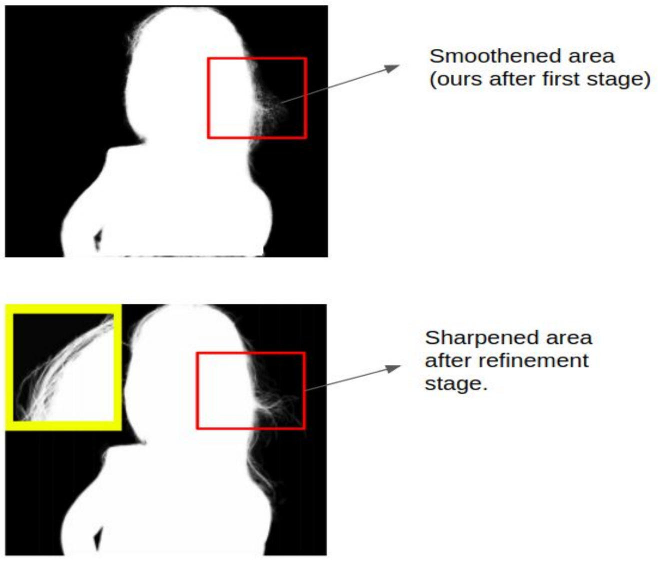


Image
Alpha ground truth
Alpha 5 epochs
Alpha 10 epochs
Alpha 30 epochs
Trimap
fg ground truth
fg 5 epochs
fg 10 epochs
fg 30 epochs

REFINEMENT NETWORK

- Because of the encoder-decoder structure, the results are sometimes overly smooth.
- Therefore the refinement stage is introduced to make the images more sharper.
- The network is extended now with the addition of refinement stage, which usually gives more accurate results i.e predicts more accurate alpha mattes and sharper edges.



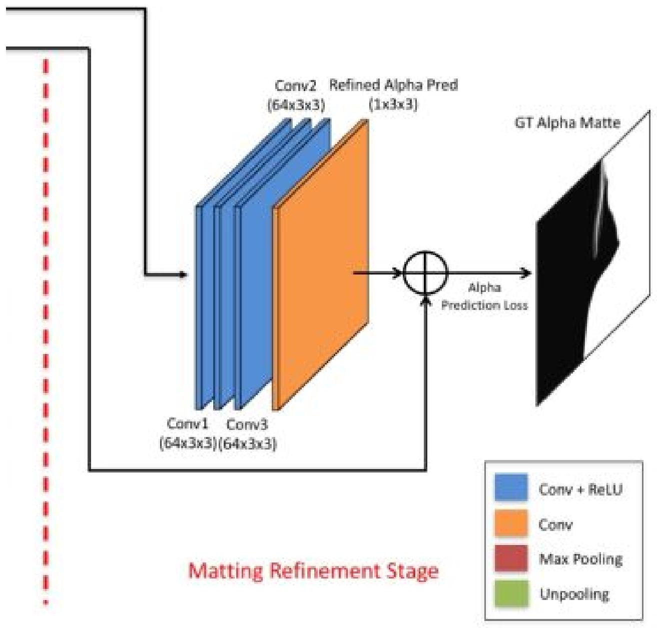
REFINEMENT NETWORK STRUCTURE

Input: Alpha matte of encoder-decoder structure.
Output: Corresponding predicted alpha matte.
Structure:

- The network contains four convolutional layers and is a fully convolutional network.
- The first 3 are followed by a non linear “RELU” layer.
- There are no downsampling layers here as we want to keep very subtle structures missed in the first stage.
- A “skip-model” structure is used where the 4th channel of the input data is first scaled between 0 and 1 and then is added to the output of the network.

<u>Training:</u>

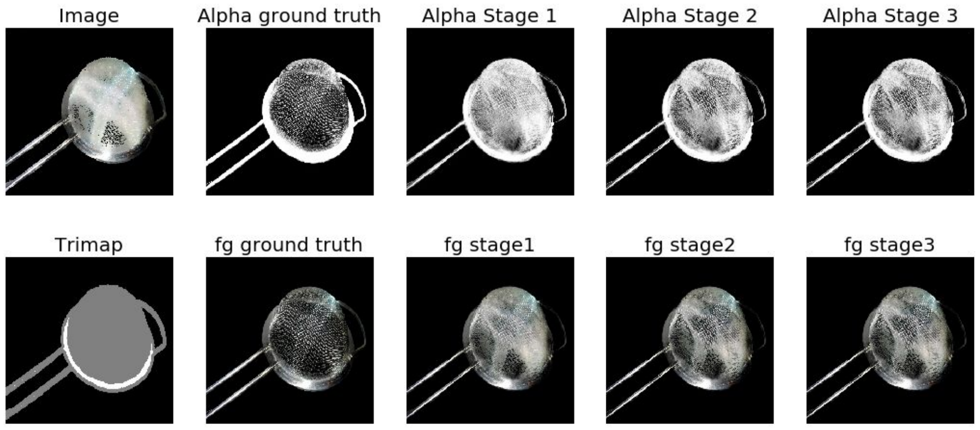
- First the encoder-decoder part is trained. Then, after the encoder-decoder part is converged, we fix the parameters and then update the refinement part.
- Backpropagation takes on the refinement part of the network in this step.
- Only the alpha-prediction loss is used to its simple structure.
- Now, the refinement part is also converged. So, we fine-tune the whole network together.
- The method used to fine-tuning is “Adam”.
- A small learning rate 10^{-5} is set constantly during the training process.

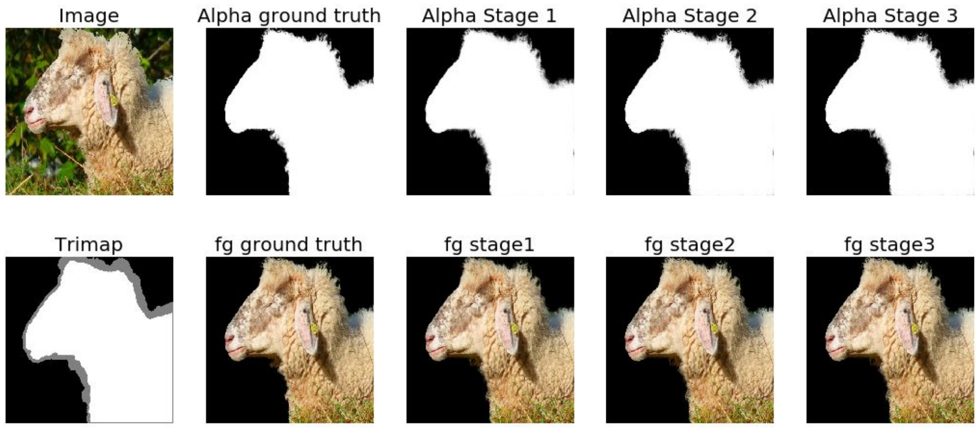


EVALUATION ACCURACIES

Stage	Epochs	SAD error	MSE error	Model Code
1	23	59.6	0.019	OURS EDNET
2	21	54.6	0.018	OURS EDNET
3	25	50.4	0.017	OURS EDNET

RESULTS OF DIFFERENT STAGES





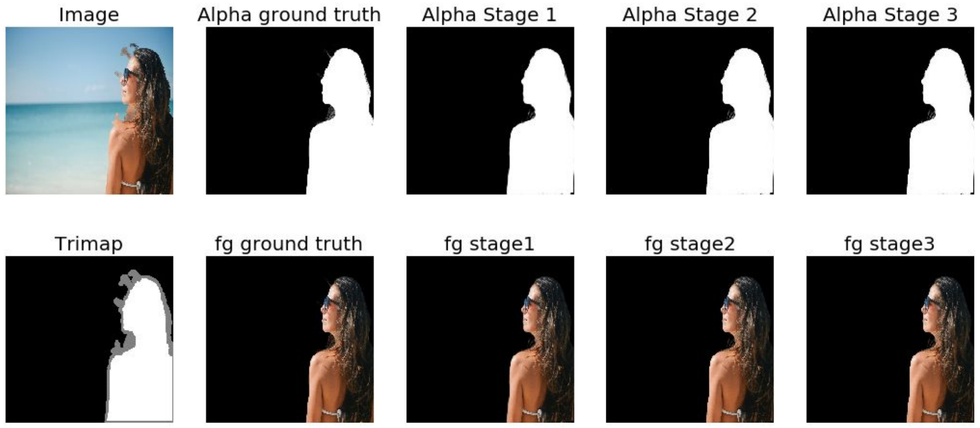


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

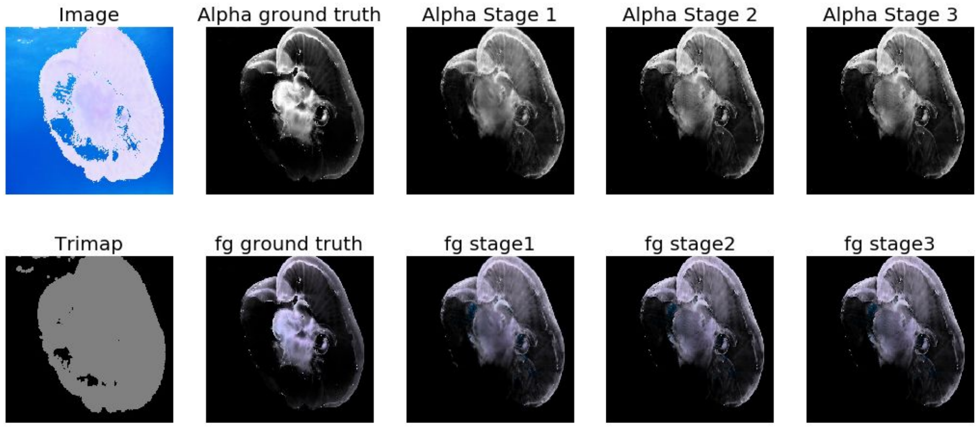


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

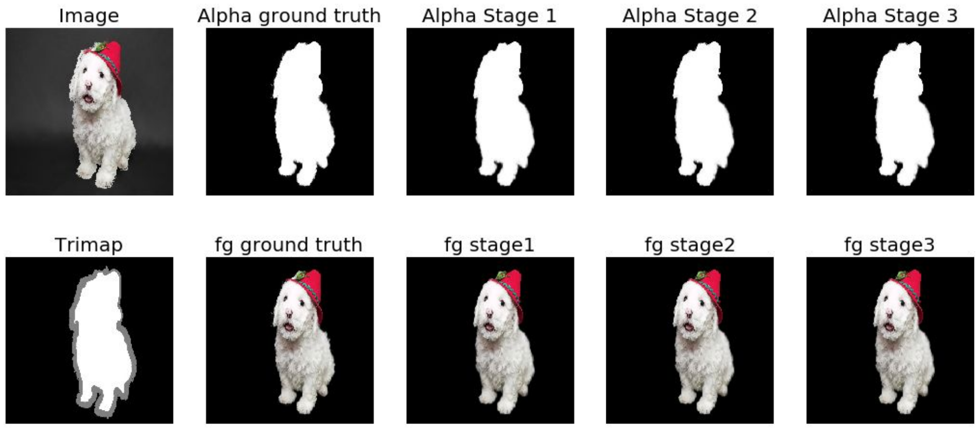


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

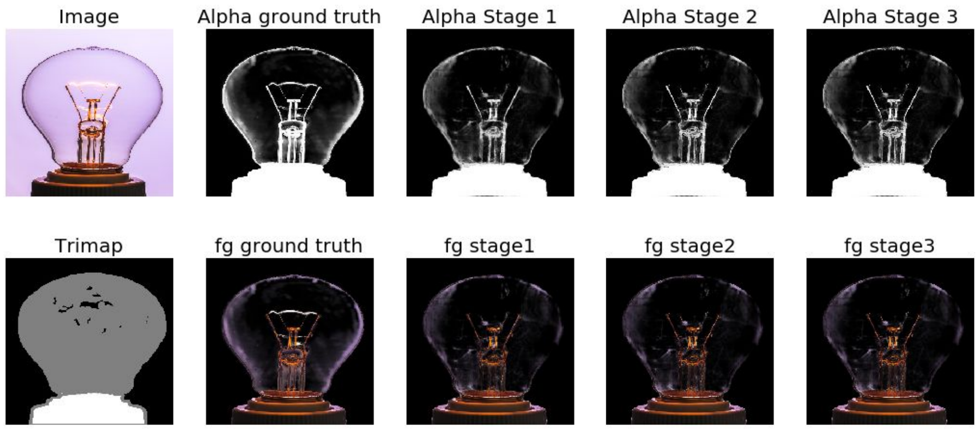


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

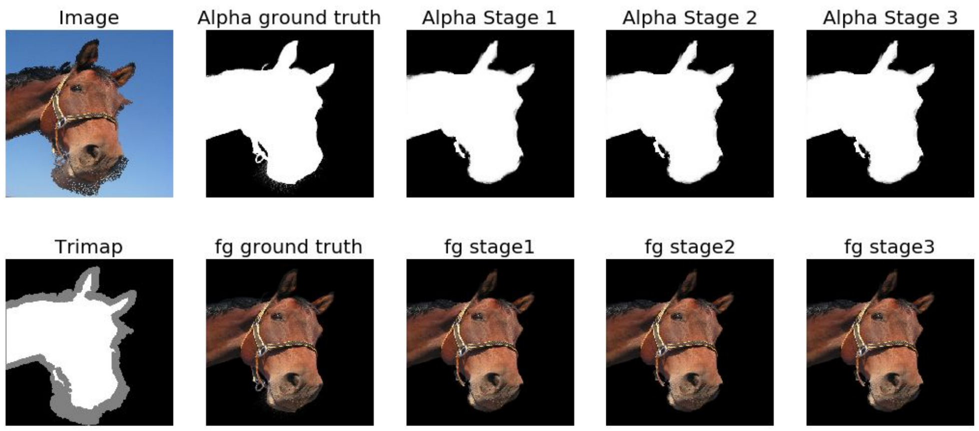


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

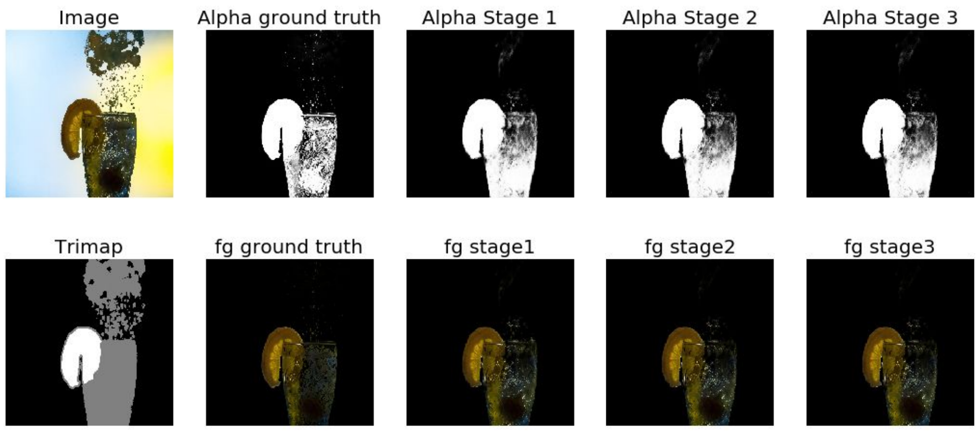


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

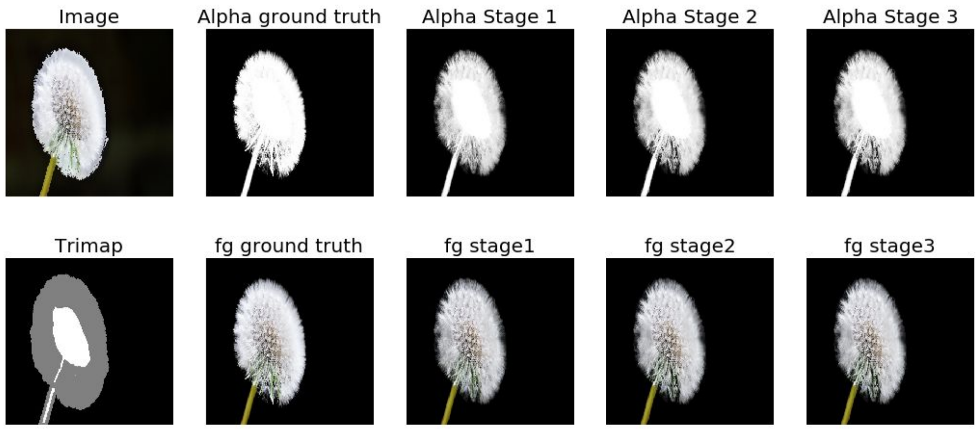


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

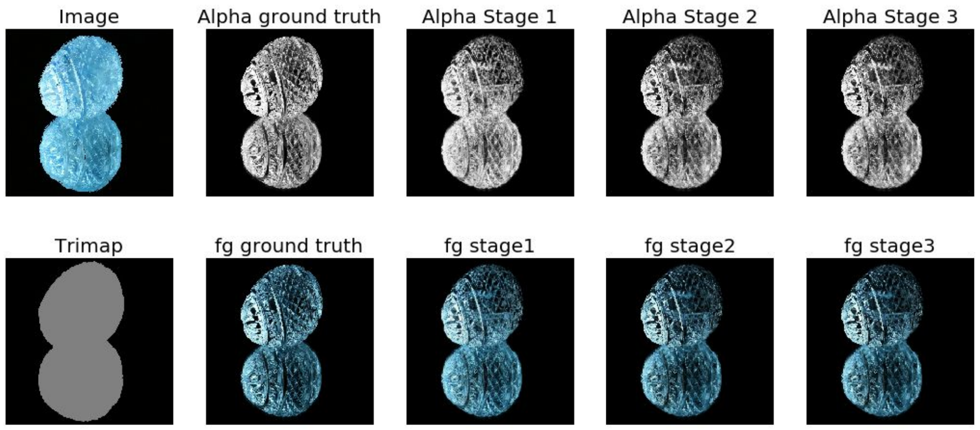


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

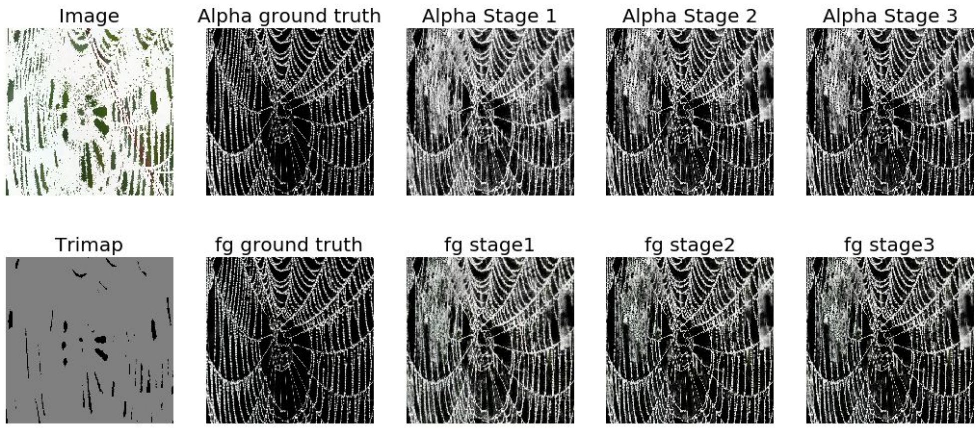


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

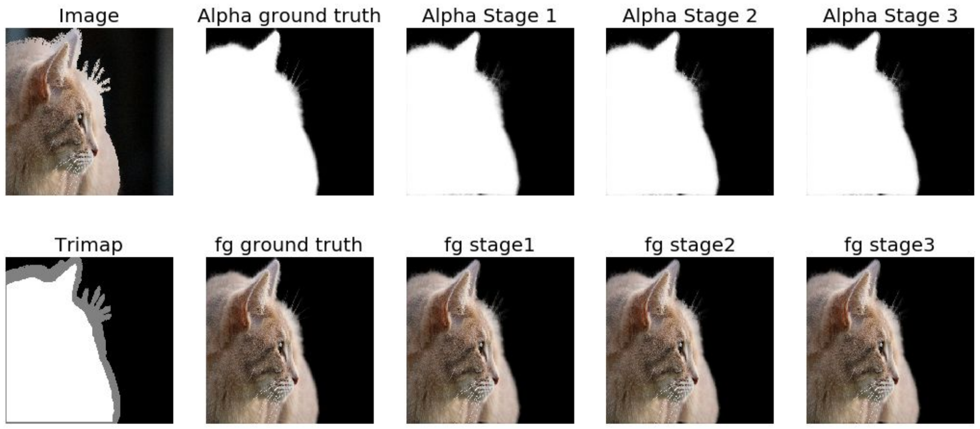


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

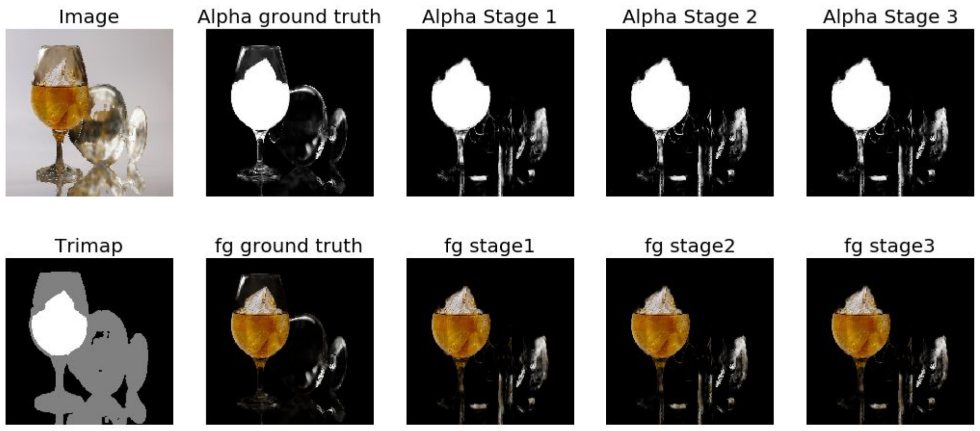


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

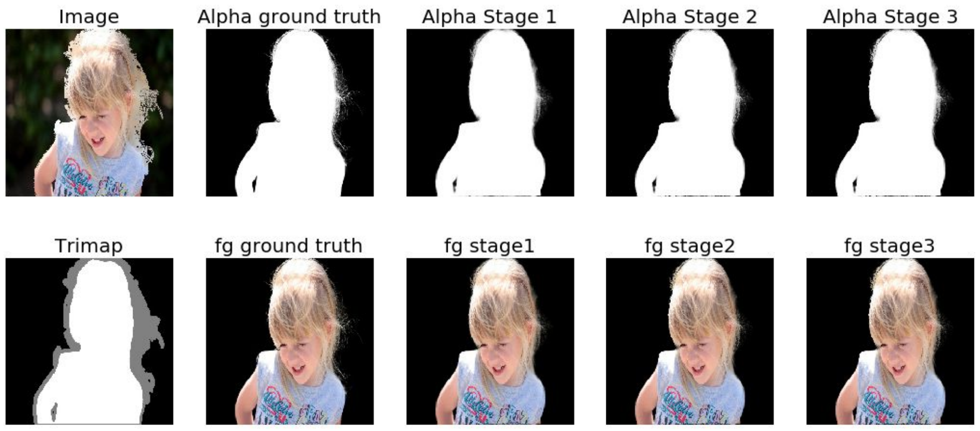
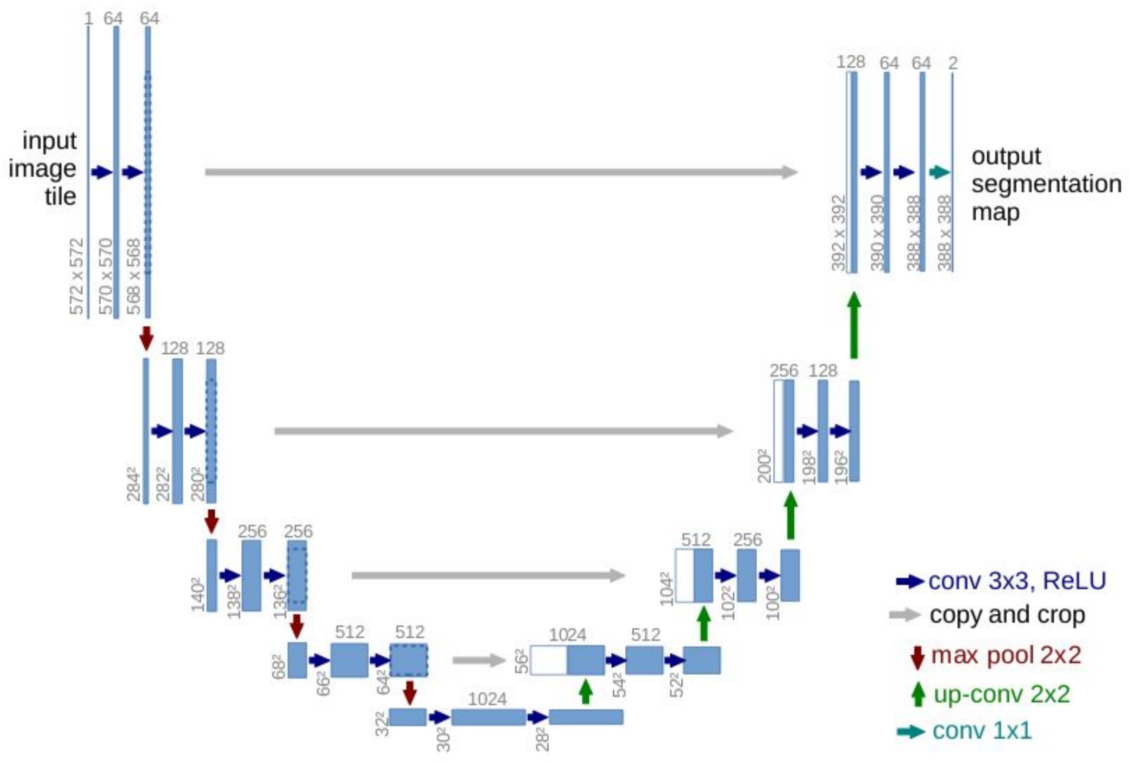


Image
Alpha ground truth
Alpha Stage 1
Alpha Stage 2
Alpha Stage 3
Trimap
fg ground truth
fg stage1
fg stage2
fg stage3

EXPERIMENTS USING DIFFERENT ENCODER-DECODER NETWORKS

U-Net: Convolutional Networks

- Sliding window setup to predict the class label of each pixel by providing a local region around the pixel as input.



Here each blue box is a multi-channel feature map.

PROS:

- The localization of this network i.e this network can localize.
- The training data in terms of patches is much larger.(than the training images)

CONS:

- It is quite slow as the network is run separately for each patch.
- There is a lot of redundancy due to overlapping of patches.
- Trade-off between localization accuracy and "use of context".

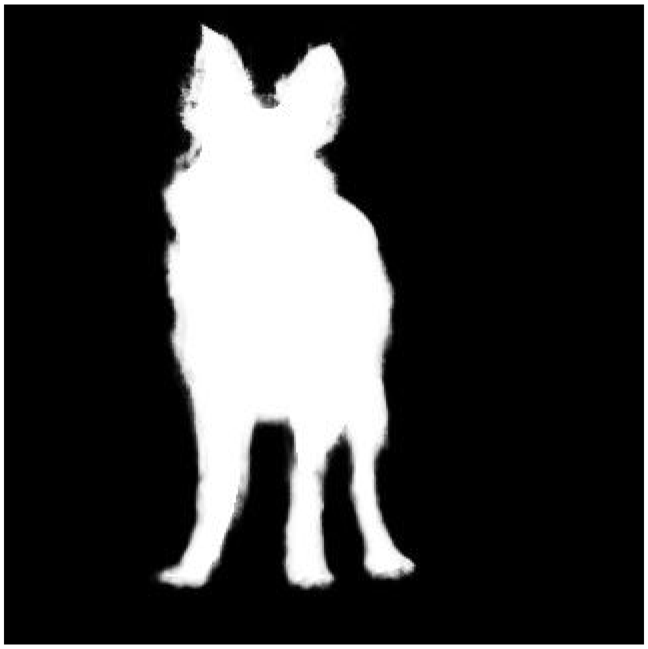
- Larger patches require more max-pooling layers that reduce localisation accuracy.
- Smaller patches allow network to see only little context.

U-NET Architecture:

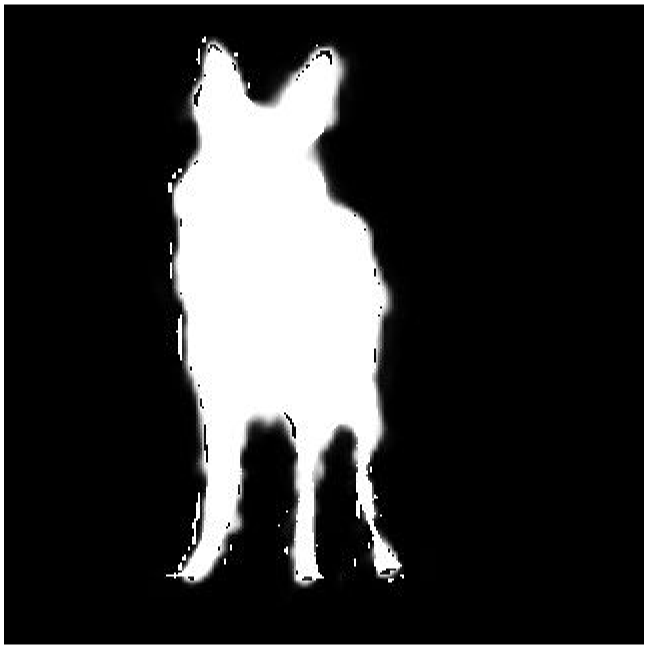
- The main idea behind this model is to “supplement a usual contracting network by successive layers, where pooling operations are replaced by upsampling operators”. This increases the resolution of the input.
- For localization, high-resolution features from the contracting path are combined with the upsampled output.
- In this network, in the upsampling part because of the large number of feature channels the context information is able to propagate to higher resolution layers.
- Thus, the expansive path is more symmetric to the contracting path, thus the name “U-Net”
- The network doesn’t have any fully connected layers and the segmentation map only contains pixels for which the full context is available in the input image.
- A weighted loss is used, where the separating background labels between touching cells obtain a large weight in the loss function.

U-Net: Convolutional Networks vs Our Encoder-Decoder Network

Using our Encoder-Decoder Network



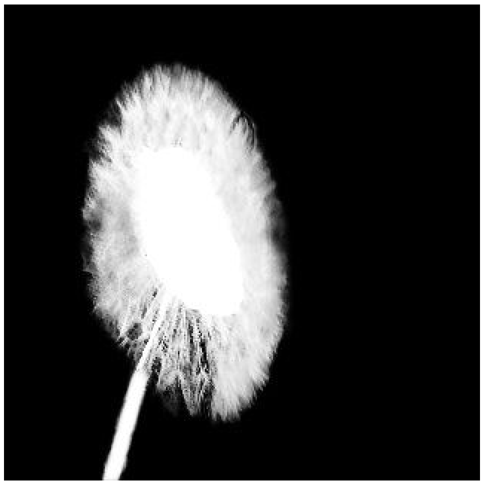
U-Net encoder-decoder network

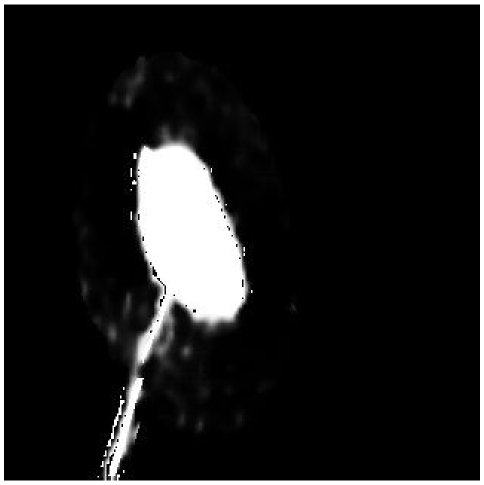


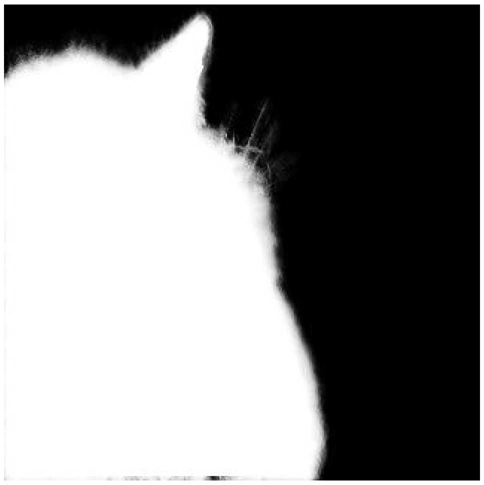
- One of the main observation from this experiment is that network trained using U-Net architecture is not able to predict the border of the image properly.

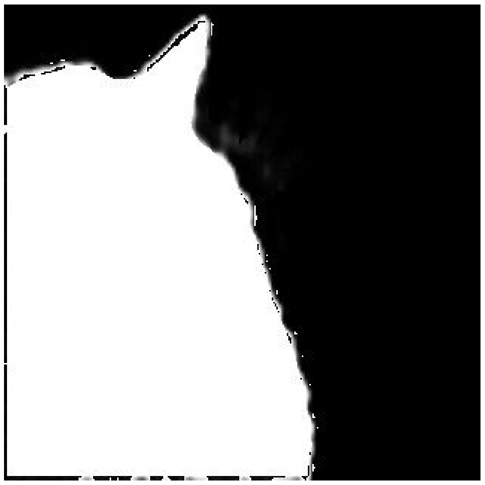
RESULTS USING U-NET ARCHITECTURE

OURS vs UNet



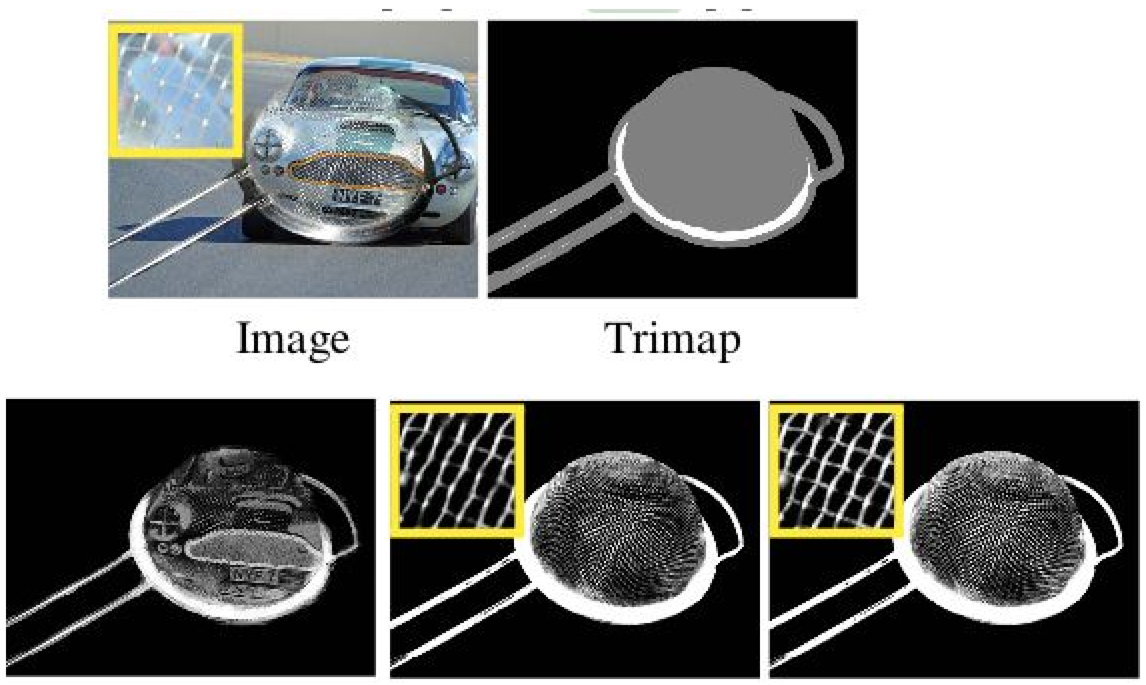






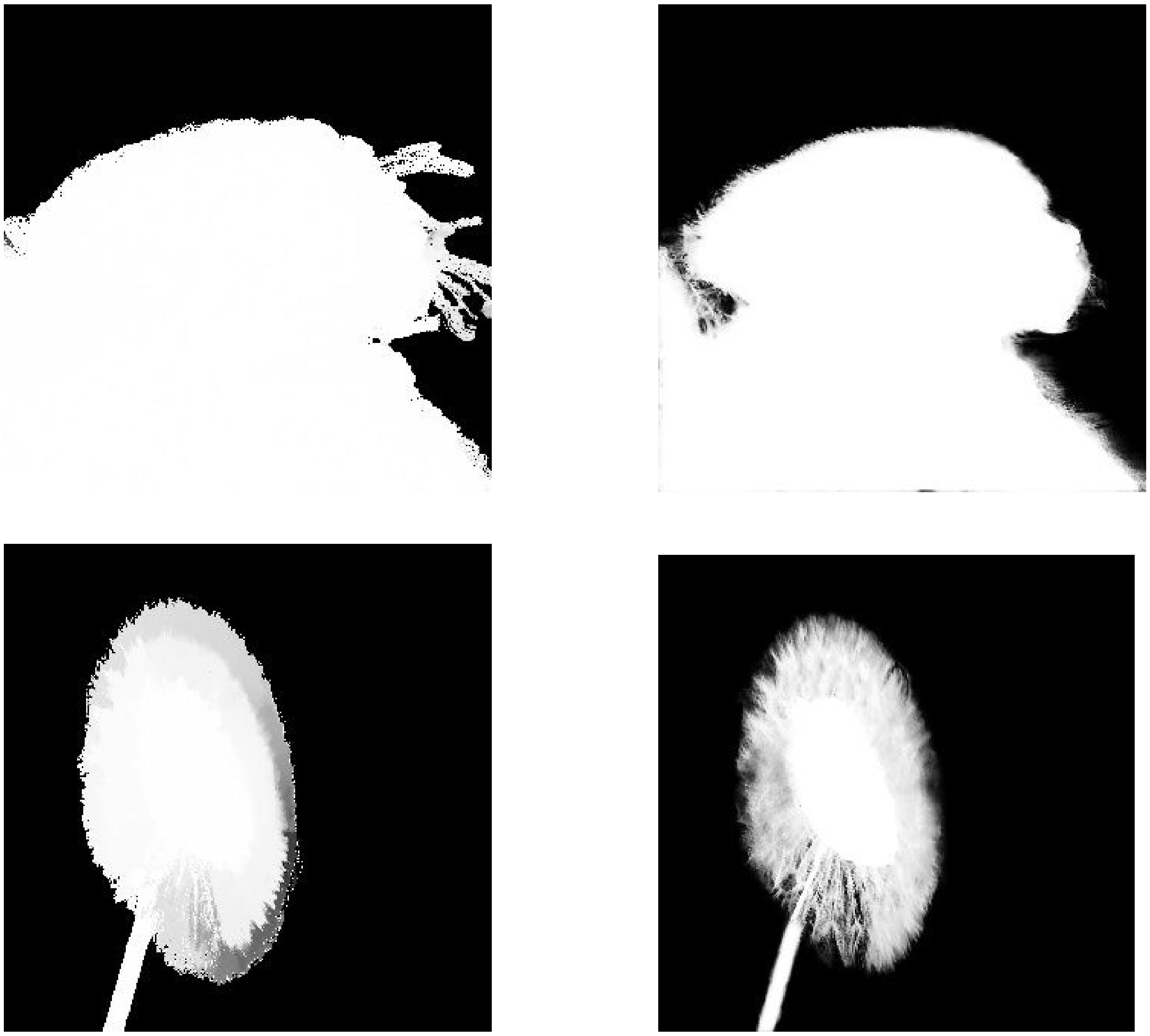
KNN-MATTING

- This approach uses K-nearest neighbors (KNN) searching in the feature space for matching, and uses an improved matching metric to achieve good results with a simpler algorithm than other architectures.
- The local 4D color-line model which is widely adopted by other matting approaches is not adopted here. Thus resulting in a better generalising in any color space in any dimensions.
- It does not require a large kernel to collect in defining the Laplacian, nor does it require good foreground and background sample pairs.
- However it still lacks much when compared to our encoder-decoder network. This is shown in the image below.



KNN **Ours-Raw** **Ours-Refined**

KNN-MATCHING VS DEEP-IMAGE MATTING



OBSERVATIONS

- The results of KNN-matching lacks refinement even when compared to just the encoder-decoder network.
- It is also observed that as in the first image in the above results, KNN matching is predicting some part of background as foreground.

Blue Screen Matting

- This method addresses matting problem using Triangulation method.
- Triangulation method takes two composite images and their respective backgrounds separately. Then the alpha values are obtained using these.
- It forms a matrix equation with alpha values of the image as an unknown, next it uses the concept of pseudo inverse to compute the alpha matte.

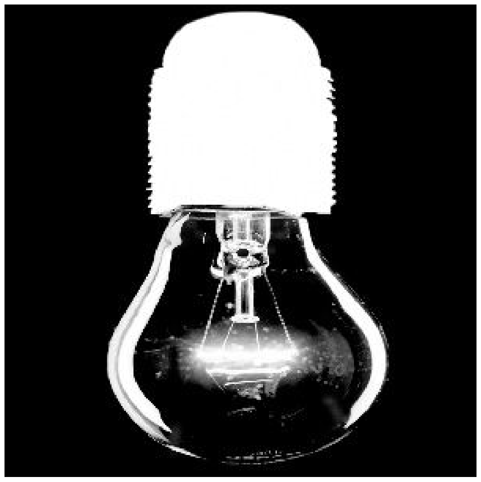
$$C_o \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ -R_k & -G_k & -B_k & t_4 \end{bmatrix} = \begin{bmatrix} R_\Delta & G_\Delta & B_\Delta & T \end{bmatrix}$$

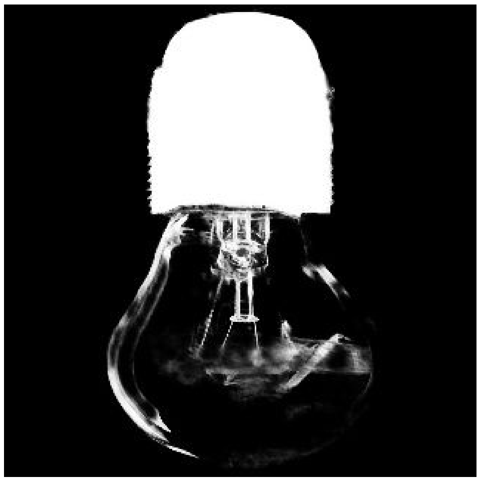
<u>COMPARISON OF THIS METHOD WITH OURS</u>

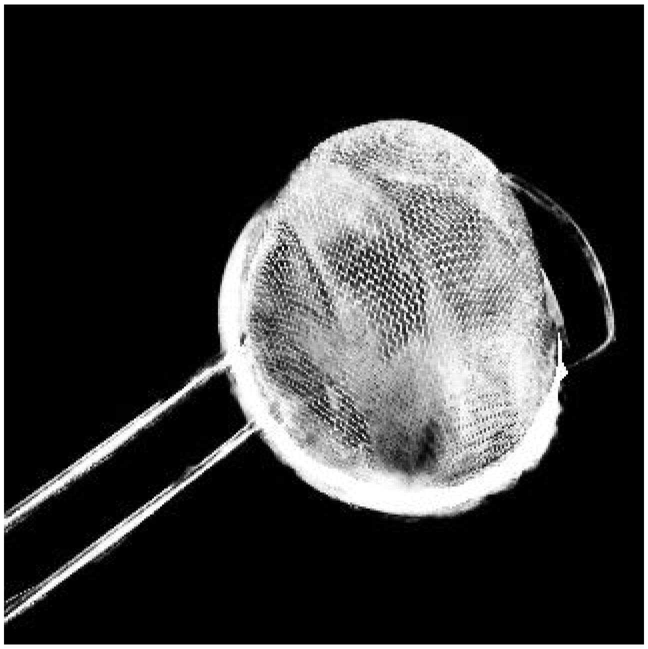
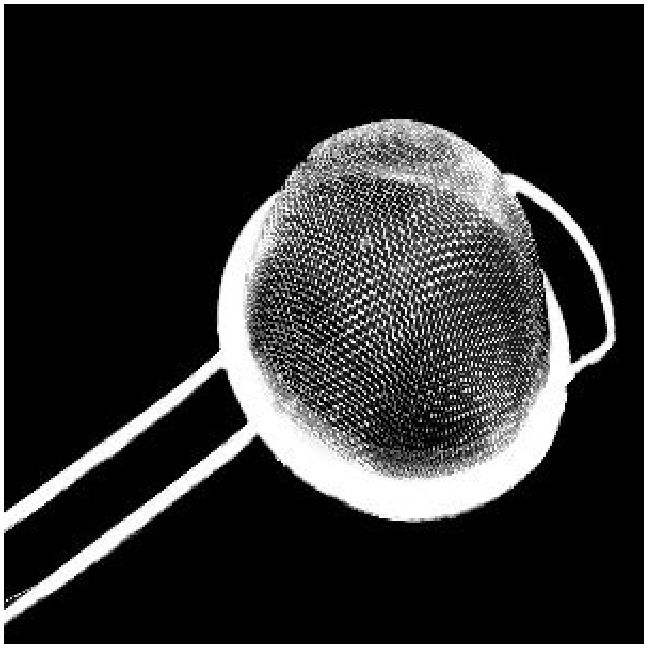
- The main limitation of this method is that this method requires the images of backgrounds separately while obtaining alpha matte whereas our method does not need any such input.

RESULTS

<u>OURS</u> <u>VS</u> <u>BLUE SCREEN MATTING</u>





OBSERVATIONS AND GOODNESS OF OUR METHOD

- Blue screen matting has almost close to 100 accuracy. It can be seen that our method is able to achieve results just closer to this.
- Deep-image-matting is able to get significant results even without the use of background images which is the case in Blue-screen matting.