Deep Image Matting

CV Project Evaluation-0

MNS Subramanyam(20161190)
Sowrya Deep Thayi (20161029)
Appari Lalith (20161038)

Introduction

- Image matting involves extracting foreground object from an image.
- It is a key technology in image editing and film production and effective natural image matting methods can greatly improve current professional workflows.
- Matting tasks usually produces a "matte" that can be used to separate foreground from the background in a given **image**.

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i \quad \alpha_i \in [0, 1].$$

Conventional Methods

- 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.

Approach to overcome these problems

- 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.

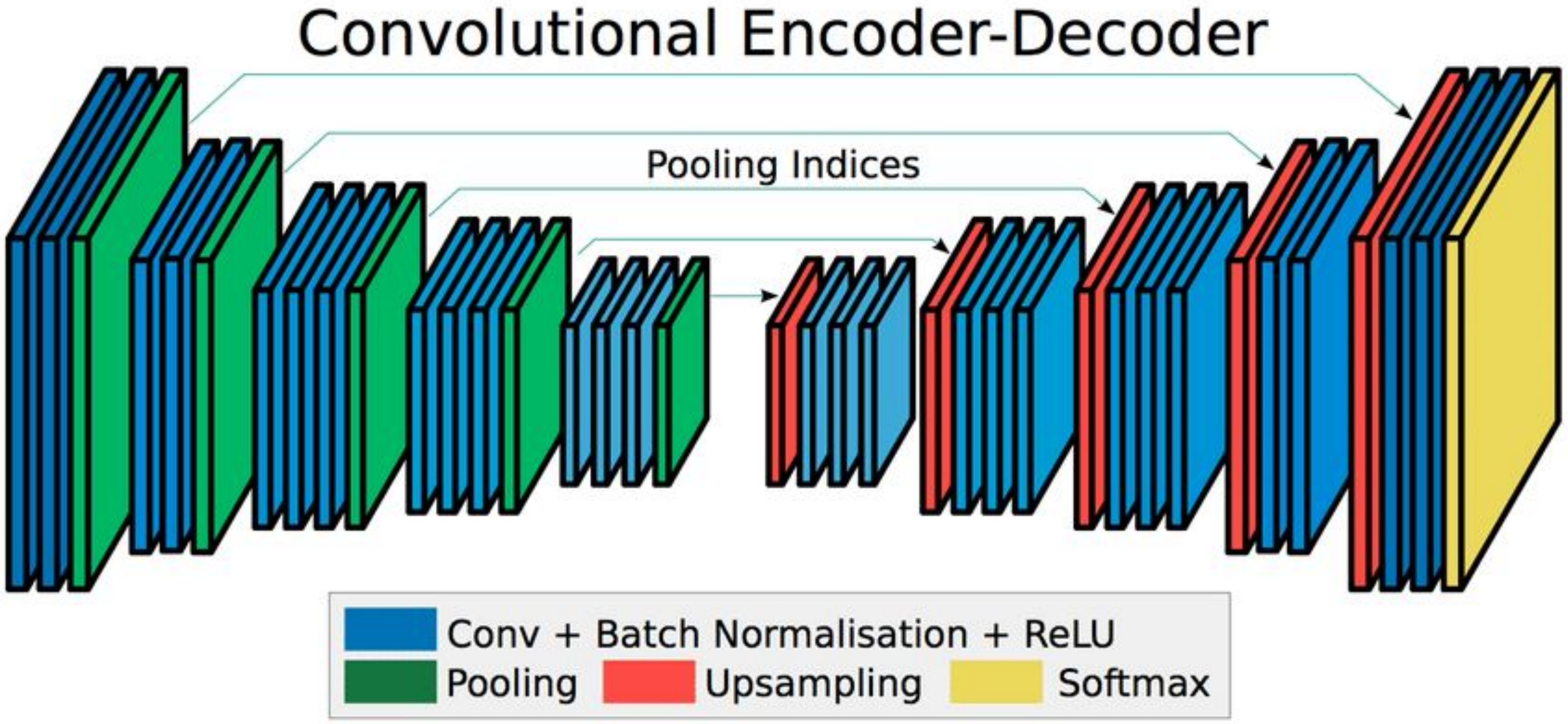
Deep learning approach

- This deep learning model has two parts.
- The first part is a deep convolutional encoder-decoder network that takes an image and the corresponding trimap as inputs and predict the alpha matte of the image.
- The second part is a small convolutional network that refines the alpha matte predictions of the first network to have more accurate alpha values and sharper edges.

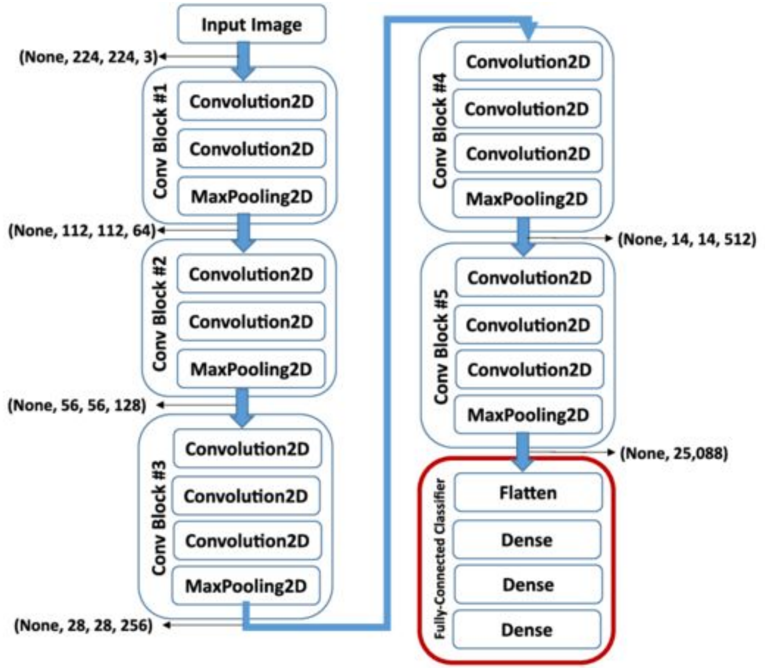
Encoder-decoder network (First part)

- 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.

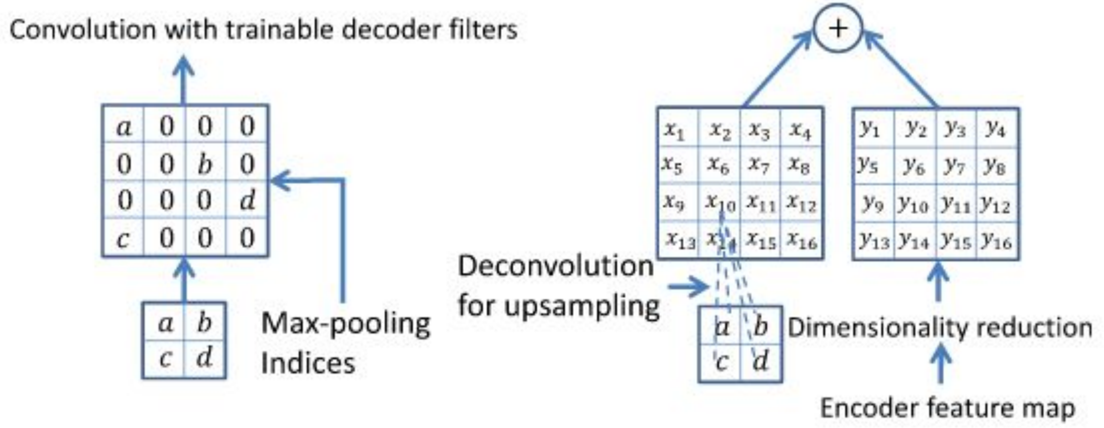
What is Encoder-Decoder network

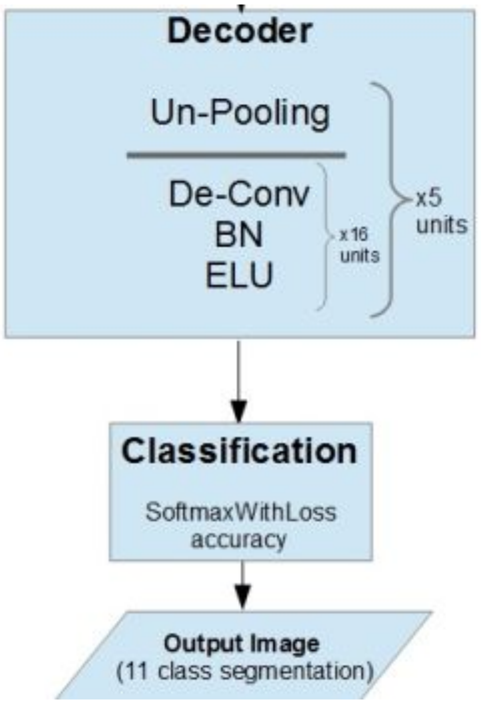

Convolutional Encoder-Decoder
Pooling Indices
Conv + Batch Normalisation + ReLU
Pooling
Upsampling
Softmax

Encoder Network

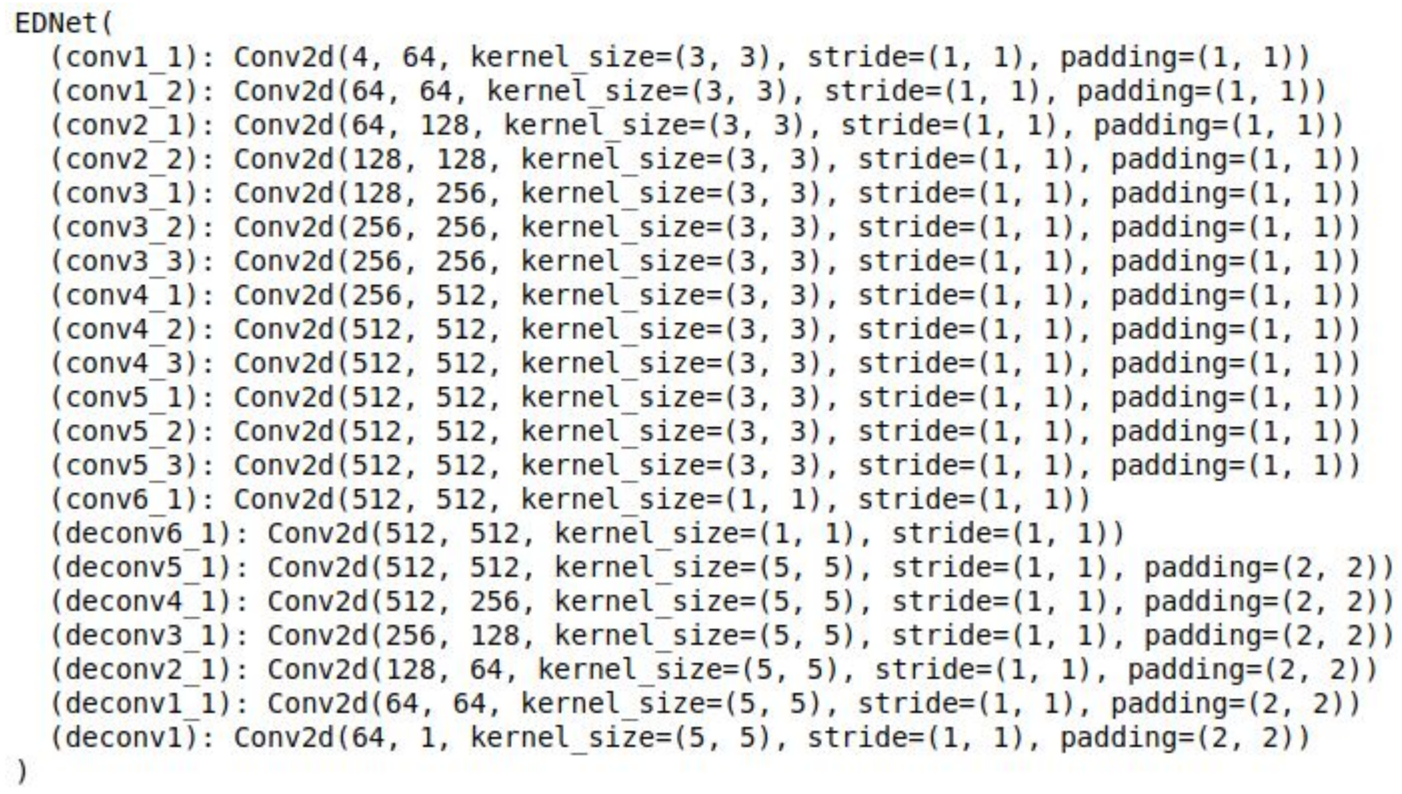


Decoder Network


Convolution with trainable decoder filters
a 0 0 0
0 0 b 0
0 0 0 d
c 0 0 0
a b
c d
Max-pooling Indices
Deconvolution for upsampling
+
x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16
y1 y2 y3 y4 y5 y6 y7 y8 y9 y10 y11 y12 y13 y14 y15 y16
a b
c d
Dimensionality reduction
Encoder feature map

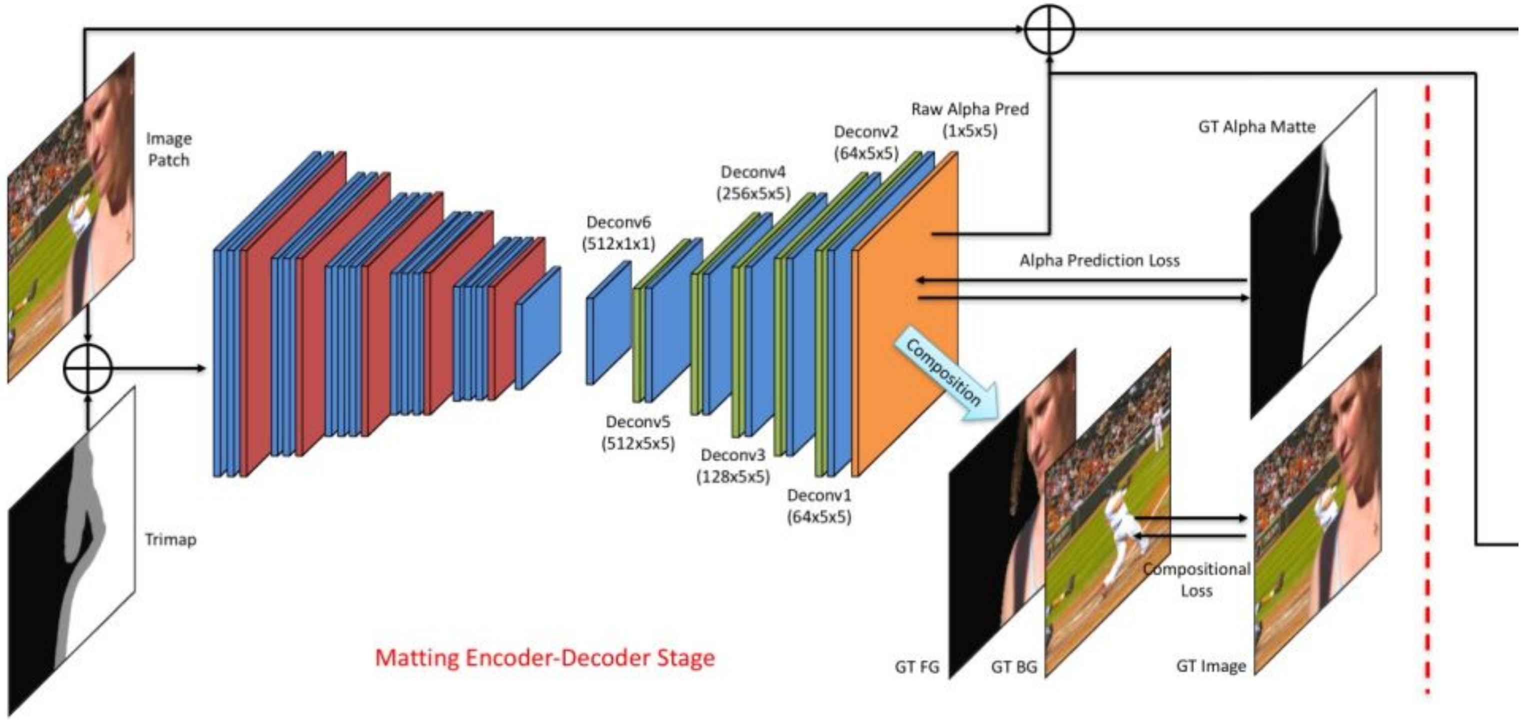


Layers in the encoder decoder net



```
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))
)
```

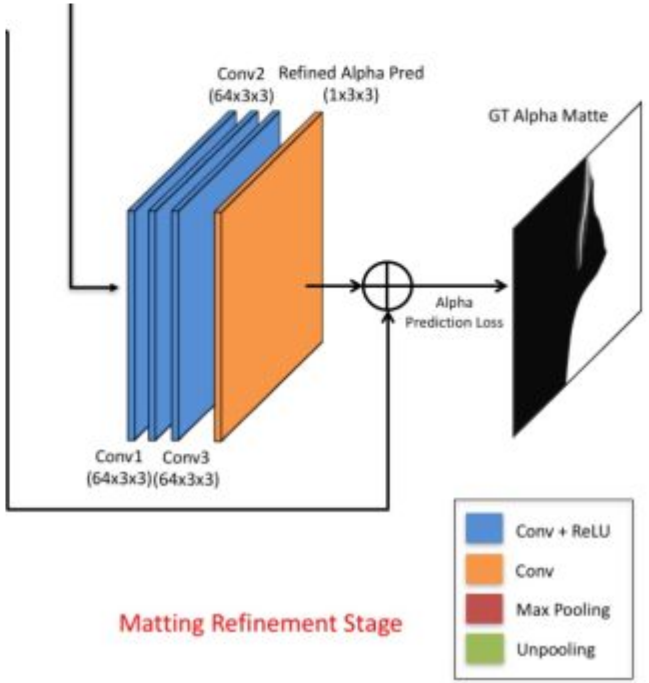
Encoder-decoder network (First part)

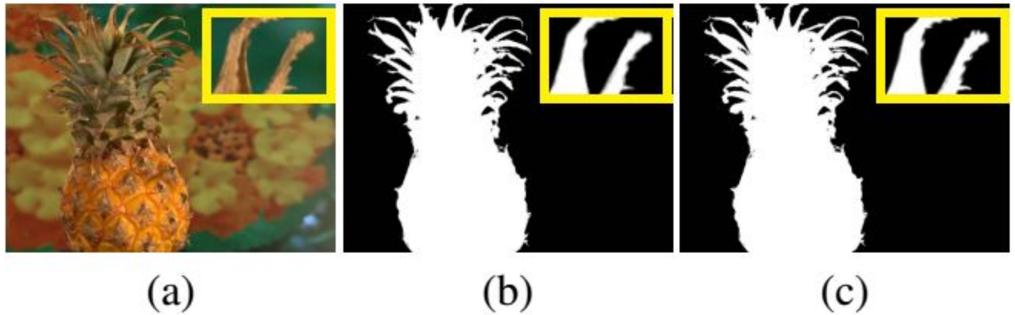


Refining the output from first part. (Second part)

- Due to encoder-decoder structure the results are usually smoothened.
- The second part network usually predicts more accurate alpha mattes and sharper edges.
- This network has 4 conv layers with 3 of them containing ReLU as activation function.
- After the refinement part the overall network is fine-tuned. Adam optimizer is used to update both parts.

Refining the output from first part. (Second part)





Datasets and dependencies

- http://www.alphamatting.com/datasets.php benchmark dataset with only 27 training images and 8 testing images.
- https://sites.google.com/view/deepimagematting dataset containing 49300 training images with 493 unique objects and 100 background images. 1000 testing images with 50 unique objects and 20 background images.
- Dependencies: pytorch, GPUs (already have ada accounts) and other python packages.

Thank You