Image Matting

Jiedong Lang
Hongyang Wang
Zhengyan Shi

December 10, 2021

Abstract

Image matting is a fundamental computer vision problem and has many applications. Many previous works focus on applying machine learning algorithms to image matting. However, most of them have poor performance when an image has similar background colors or complex textures. This project will use the unsupervised learning algorithm Kmeans and GMM to achieve matting as the benchmark. Moreover, we compare the results with the deep learning model. In addition, we use a large-scale image matting dataset including 28000 training images and 6000 testing images. We evaluate our deep learning and achieve a relatively good result.

1 Introduction

1.1 The Matting Problem

Extracting foreground objects from images or video sequences is important in many images and video editing applications. We are interested in exploring the cutting-edge technology of image matting. Combined with what we have learned before, traditional machine learning has been developed in image recognition and matting, and deep learning plays an important growing role in this field. Therefore, we plan to use deep learning models to realize image recognition and matting and then compare results with traditional machine learning models.

As machine learning has more in-depth research on how to divide the image into the foreground and background, the focus on the technology of image matting has also increased. VGGNet improves performance by continuously deepening the network structure. Increasing the number of network layers will not increase the number of parameters because the number of parameters is mainly concentrated in the last three fully connected layers. Because of more non-linear operations, the former is more vital to learning features.

Currently, there are two limitations. First, current methods formulate matting problems as a linear combination of two colors. It is susceptible to situations where colors overlap foreground/background color distribution. Unfortunately for these methods, it is common for natural images, often leading to low frequency 'smearing' or high frequency 'chunky' artifacts depending on the method. Another limitation is focusing on small datasets. At some point, methods will overfit the dataset and no longer generalize to real scenes. Image matting is also called pull mask or digital mask, which accurately separates the foreground object from the background, including determining full and partial pixel coverage. This problem was established mathematically in 1984 by Porter and Duff.[4] They introduced the alpha value to act on the relationship between the foreground and the background to achieve the purpose of anti-aliasing. The formula can be expressed as $I_z = \alpha_z F_z + (1-\alpha_z)B_z$, where I_z is the image, F_z is the foreground image, B_z is the background image, and α_z is a coefficient used to balance the relationship between foreground and background. Alpha can be any value in the range 0 to 1. When Alpha equals 1, the image will be the foreground image. When Alpha equals 0, then it will be background only.

Our approach is mainly inspired by the paper, Deep Image Matting, to predict the alpha value in the mathematical problem mentioned above. The model is a VGGNet 16 based SegNet Model. Due to the time limitation, we only reproduce the first stage - build a convolutional encoder-decoder network. This network takes an image and the corresponding trimap as inputs to predict the alpha value of the input image. Also, we try data augmentation methods to improve the model's performance.

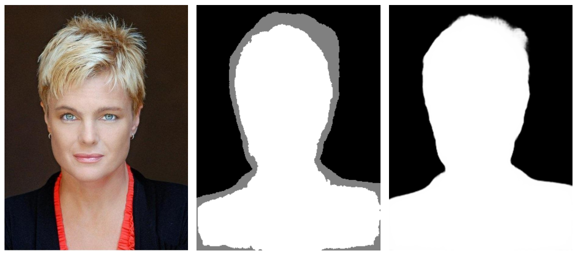
1.2 RELATED WORKS

There are several famous works about applying deep learning in image Matting. DIM (Deep Image matting) explains for the first time that given images and auxiliary information trimap, Alpha can be learned end-to-end. The network is divided into two stages. The first stage is a deep convolutional encoder-decoder network. The second stage is a small convolutional neural network to reduce the loss of detail caused by the encoder-decoder network and improve the accuracy and accuracy of Alpha prediction. NVIDIA's Deep Image Matting is one of them. Researchers use low-level and high-level information, take the original image and trimap as input to get a rough alpha channel map and then use small convolution to optimize the matting result to get the mask's alpha loss.[2] AlphaGAN natural image matting is a matting work based on generating a confrontation network after deep image matting. The generator is used to generate the alpha channel, and the discriminator is used to determine whether the synthesized image is true according to the generated matting result. In order to better solve the spatial positioning problem of CNN, the encoder captures global feature information by expanding convolution, avoids down-sampling the feature map, and overcomes the problem of spatial information loss in deepmatting.[1] BGMv2 (Background Matting v2) changes the way of thinking. It uses background pictures instead of trimap to assist the network in making predictions, effectively avoiding the time-consuming and laborious problem of trimap acquisition. The network is divided into two parts: Base network and Refiner. In the Base network stage, many low-resolution preliminary predictions are made. In the Refiner stage, the corresponding slices of the high-resolution image are Refined through the Error Map. In this way, real-time prediction of high-resolution images is realized[3]

2 PRELIMINARIES

2.1 Dataset and Features

We draw our dataset from Kaggle. It is currently the largest portrait matting data set, including 34,427 images and corresponding matting results. The dataset can be separated into two-part: 1) the first part is the original images; 2) the second part is the matting results developed by the AISegment company, and we can get the alpha matte by transforming matting results as labels. (y-value). For all original images, they are half-length portraits of 600*800. Due to the limited computation, we resize all the images to 28×28 and get the corresponding trimap through the given function. We can see one example in Figure 1 to see input result and output result.



(a) Input. (b) trimap. (c) Alpha.

Figure 1: Example of input image (left), trimap (middle) and target (right).

2.2 Experimental Setup

We split the data into a training set and a test set with 80% observations in the training set and 20% observations in the test set, so we have roughly 28,426 images for the training set and 6,000 images for the test set.

For the project model, we mainly focus on the performance of the VGG-16 network and then compare the VGG-16 model with the classical unsupervised machine learning.

For the unsupervised model, we implement the Kmeans or GMM model to find the alpha of images. The specific procedure is : 1) first of all, we read the original picture into ndarray; 2) then clustered the pixel's value by using K-means or GMM; 3)finally, turn the processed ndarray into an image.

Table 1: Hyper-parameters table

Part		
Parameter Name	Description	Value
num_epochs	epochs	10
batch_size	batch size	64
learning_rate	learning rate	0.01

For hyper-parameters of the VGG-16, we can see the default hyper-parameter value in table 1. In the meantime, we also try experiments on data augmentation methods (flip and rotate) to improve the performance of our deep learning model.

Due to the constraints on GPU resources, we can only use the CPU to train our machine learning model.

2.3 Problem Setup

2.4 Unsupervised Machine Learning Model

The Kmeans clustering aims to partition n observations into k clusters. Each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a cluster prototype. The initial positions of k centers are randomly decided.

The GMM model is a probabilistic model to represent the presence of a sub-population. Formally a mixture model corresponds to the mixture distribution representing the probability distribution of observations in the overall population. However, while problems associated with "mixture distributions" relate to deriving the properties of the overall population from those of the sub-populations, "mixture models" are used to make statistical inferences about the properties of the sub-populations given only observations on the pooled population, without subpopulation identity information.

2.4.1 Deep Learning Model

In this project, we reproduce the first stage of the paper. It is a deep convolutional encoder-decoder network to predict the alpha matte of input images. It roughly predicts the boundary foreground objects.

For the model architecture, we can see it from Figure 2. For input images, they are image patches and the corresponding computed trimap, which are concatenated along the channel dimension, resulting in a 4-channel input. Therefore, we build a convolutional layer that transforms the 4-channel input to 3-channel

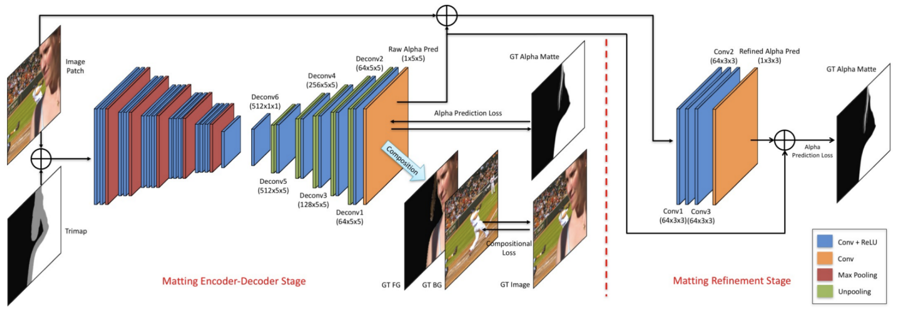


Figure 2: Encoder-Decoder network is the left network of the red dotted line

input. The encoder uses the first 13 convolutional layers of VGG-16 and four max-pooling layers as the encoder. Moreover, the encoder portion has the same number of parameters as VGG-16 (the conventional layers with 3×3 kernel size, one stride, one padding, and max pooling operation with size two, one stride, one padding). The decoder consists of six convolutional layers and four unpooling layers, followed by a final alpha prediction layer.

2.4.2 Metrics

We use the loss called the alpha-prediction loss that is introduced in the paper, $DeepImageMatting$. It is the absolute difference between the ground truth alpha values (label alpha) and the predicted alpha values at each pixel.

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

where α_p^i is the output of the prediction layer at pixel i. $alpha_g^i$ is the ground truth alpha value at pixel. Epsilon is a small value.

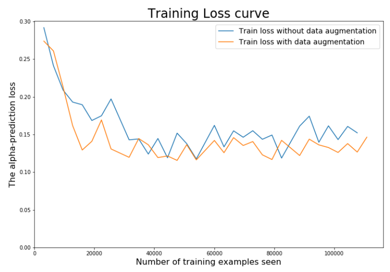
3 Results

We implemented three models: unsupervised machine learning (Kmeans GMM), VGG-16 without data augmentation, and VGG-16 with data augmentation. Figure 3 and Table 2 show the alpha-prediction loss curve of the training set and the test set from the VGG-16 without data augmentation. We represent our several comparisons in Figures 4, 5, 6. After model training, the alpha prediction loss can be reduced to about 0.15. However, the test loss is about 0.3, although our model might overfit. Our prediction result can roughly predict the human body cut with a rough edge when we see our prediction result. We guess that the reason is that we set a small number of epochs so that our model might not be trained very well, resulting in a higher loss in the test set.

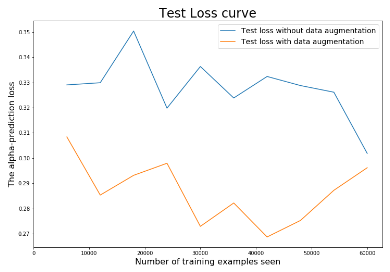
Table 2: Comparison of model performance

Model Name	Training Loss	Test Loss
VGG-16	0.169581.	0.301838
VGG-16 (with DA)	0.146492	0.296174

We randomly choose 50% of images to flip and rotate for data augmentation methods. We can see that both models have similar performance, but data augmentation methods improve the performance of our model. We guess that the encoder-decoder network predicts the raw alpha matte of input images so that the boundary of images may be crudely predicted. Therefore, although with data augmentation methods, our model cannot predict alpha matte with a sharp edge. Suppose the next refinement stage is added to our model. The data augmentation methods may have better performance.

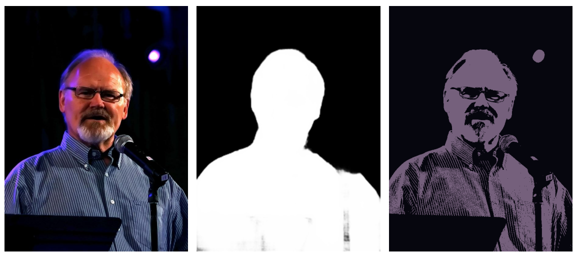


(a) Training Set Loss.



(b) Test Set Loss.

Figure 3: Example of input image (left), trimap (middle) and target (right).



(a) Input. (b) VGG16-DA. (c) ML.

Figure 4: Comparison 1 of result

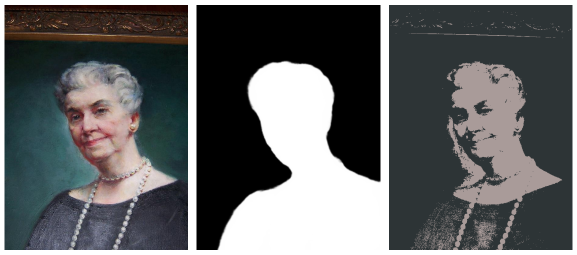

(a) Input. (b) VGG16-DA. (c) ML.

Figure 5: Comparison 2 of result

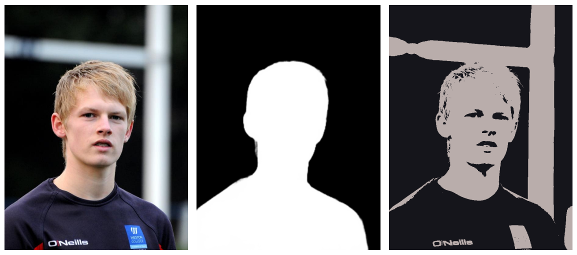

(a) Input. (b) VGG16-DA. (c) ML.

Figure 6: Comparison 3 of result.

4 Discussion

Based on our results, we think our result is pretty good (although not good enough). The reason for 'pretty good' is because we have compared it with the unsupervised method and find that the effect is way much better, which is a bonus point. As for 'not good enough, ' our model only reproduces the first stage of the paper to predict the alpha matte of the image, which is 'rough.' Of course, the good part is that we use data augmentation to improve the model's performance, which reduces the loss of the model. We can further optimize our model's refinement step if we have more time and more resources.

5 Conclusion

Here we are going to summarize our project. We reproduced the first stage of the paper: Implementing Deep Image Matting by using VGG-16. Plus, we also implement Kmeans and GMM to use as a benchmark. We use the alpha-prediction loss as the metric. The result shows that the deep learning model performs better than the unsupervised machine learning, and within VGG-16, the VGG-16 with data augmentation has a more remarkable performance.

For future work, we intend to extend this project in several ways. In particular, we intend to (1) implement the refinement stage neural network, (2) explore the hyper-parameter for VGG-16, (3) explore more effective ways to pre-process our input images (4) explore more structure of deep learning models.

References

[1] David Cochard. *Deep image matting.* URL: https://medium.com/axinc-ai/deep-image-matting-a-machine-learning-model-to-improve-the-accuracy-of-image-matting-2ff98e0b47d6.

[2] Vedanta Jha. *Understanding AlphaGAN matting.* 2019. URL: https://medium.com/vedacv/understanding-alphagan-matting-1dfae112a412.

[3] Shanchuan Lin. *Robust High-Resolution Video Matting with Temporal Guidance.* 2018. URL: https://arxiv.org/pdf/2108.11515.pdf.

[4] T. Porter and T. Duf. *Compositing Digital Images.* Vol. 18. 3. 1984, pp. 253–259. URL: https://keithp.com/~keithp/porterduff/p253-porter.pdf.