# Edge-Guided Image Inpainting of Human Faces: Project Report

Team name: The Avengers SKS
Team members: Shuhao Li, Chenyue Wang, Keying Chen

## Abstract

Image inpainting is a task of filling in missing irregular regions of an image. One of the most popular uses of it is to conceal the blemish on the face of humans or restore damaged pixels on the face image. Based on the EdgeConnect[1] paper, we present a two-stage generative model, edge generator and an image completion network, to reconstruct the face image in this project. Some modifications are applied to the original model in order to achieve better model performance. Instead of using regular convolution operators, we used Gated-Convolutional presented in the paper[4] presented by Jiahui et al. As this modified convolution can provide us with a built-in contextual attention module which should help us get better inpainting results. Another major change to the model we made is that we replaced the original network structure with a Unet[5]. This change in the network structure helped us greatly during the training process due to our limited access to GPU servers while Unet can provide us with faster model convergence speed for the model. Code and models are available here *https://github.com/Shuhaoalex/CSC420Project*.

## 1. Introduction

Image inpainting is a task of synthesizing alternative contents in missing regions such that the modification is visually realistic and semantically correct. It allows people to remove the unwanted object from the background of the image. The usage of image inpainting can also be extended to super resolution, image stitching and many other tasks. Image inpainting is increasingly being used on people's face images in our daily lives. It is useful for people who want to remove wrinkles, acnes or moles from their faces or restore the damaged face image.

In general, there are two categories of approaches nowadays for doing image inpainting: patch matching using low-level features in the local region and feed-forward generative models with deep convolutional networks[4]. The former methods can usually give reasonable infilling to the missing region while often fail on complicated scenes. The later approach can take the advantage of the semantic exploit ability of deep neural networks obtained by training on large datasets. Therefore, the later approach can usually over perform the former approach on images with complicated scenes. People's face images, which is the kind of images we care about in our project, are having complicated scenes, as there are a great amount of feature points on human faces.

In our project, we chose to design our algorithm based on the overall algorithm architecture presented in paper EdgeConnect[1]. In this paper, a two-stage image inpainting model is presented. The two stages are edge generation and image completion respectively.  Applying edge detection algorithms over the unmasked region of the image would provide us with an incomplete edge map. The edge generation step is to make reasonable predictions of edges in the masked region of the image. Image completion step is to inpaint the missing regions of the input image with the help of the edge map predicted by the first stage,  then generate the final inpainting result. Compared to an end-to-end deep learning model for image inpainting, this two stage algorithm can let users have the chance to participate in the inpainting procedure by modifying the edge map prediction generated by the first stage before it is passed into the second stage of the algorithm. This can help users to avoid getting a result that might not satisfy their demands caused  by simply passing their images through a magical black box. As the only information the image completion stage can have in the masked region is the edge map, the model used in the second stage should heavily rely on the information provided by edge map while doing inpainting. In other words, the inpainting result should follow the edge map in the masked area of the image.

## 2. Literature Review

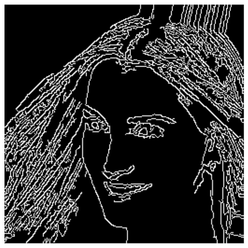
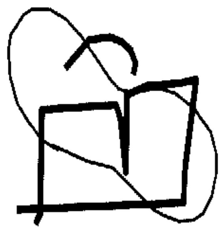
Edge-Connect method proposed in EdgeConnect Paper[1] divides image inpainting into a two-stage process: edge generation and image completion. Edge generation only focuses on predicting the edge in the missing area, and image completion uses predicted edges to estimate the color pixel in the missing region. It forms an end-to-end adversarial model that is reconstructed based on dilated convolution and residual blocks. However, the image inpainting results is highly reliable on the edge information. In case of relying too much on previous edge prediction errors, we decided to train our edge generation and image completion model separately and introduced a strong learner, gated convolution[4], for both parts of the model. The final output of gated convolution is  a multiplication of σ(gating) and φ(feature), where σ is a sigmoid function and φ is any activation function. Compared to regular convolution with dilated rate, gate convolution can learn features for each channel at each spatial location across all layers, which can  improve the consistency  of image pixels. Furthermore, in order to achieve accurate and precise segmentation of images efficiently, we use the U-net convolutional network[5].  U-net consists of two parts. The first part is a contracting network by successive layers similar to encoder, which can capture context information. The second part is a symmetric expanding path similar to a decoder, which allows precise localization. This second upsampling part uses a large number of feature channels, which can propagate context information to feature channels and retain spatial information. By combining the high resolution features with the upsampled output, we can obtain a precise result. Moreover, this network is computationally efficient. Overall, U-net helps us obtain the model with faster model convergence.

## 3. Methodology

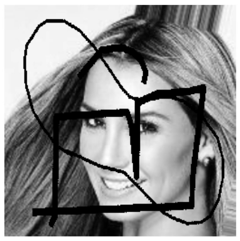
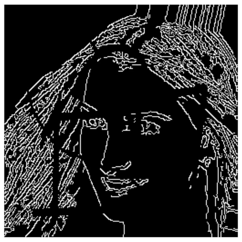
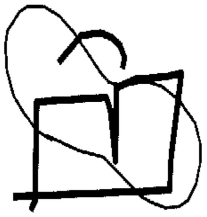
## 3.1 Dataset

We can start from our data pre-processing part. We use celebA[2] as the dataset for images, and Quick Draw Irregular Mask Dataset[3] as the dataset for irregular masks. We firstly list all training images' paths into a file before reading the images, and we then read that file and convert files' path into a list for further preprocessing data. The script for generating the file list and the function for converting it into a list is from the public code by EdgeConnect paper[1]. Then we start our data preprocessing part. Let C be the ground truth color face image we get from the celebA[2] and M to be the mask with white background and black foreground we get from Quick Draw Irregular Mask Dataset[3]. During training, color images C and masks M are randomly paired together to avoid overfitting. For the ground truth edge map, we generated it by canny algorithm. Before passing the training data to GPUs, we did data preprocessing on the CPU side to maximize the training efficiency and make them ready to be the input dataset of our model.

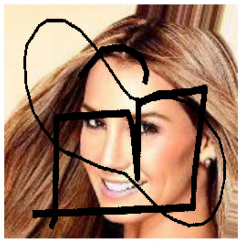
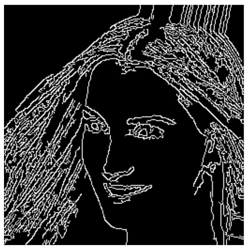
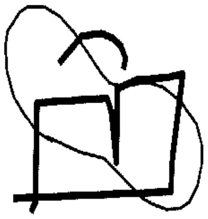
- Ground-truth Set(ground truth color image, ground truth gray image, ground truth edge image, mask)

- Edge Completion Model Dataset
  For the edge completion step, we will take the masked grayscale image, masked edge map, and the mask as an input dataset. The grayscale image here is used for providing contextual hints to the model for completing the missing edges in the masked regions. The training targets for this part are the ground truth edge maps.

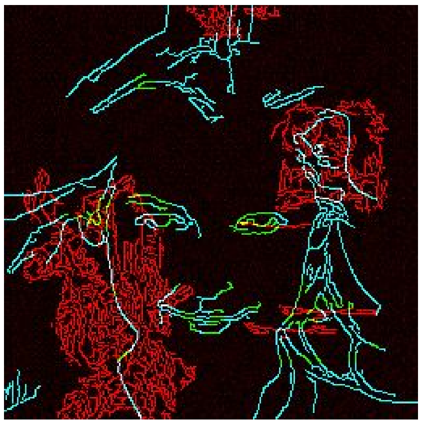
- Image Completion Model dataset
  For the image completion step, the input dataset of the model would be the masked color image, complete edge map and the mask. The training targets for this part are the ground truth color images.

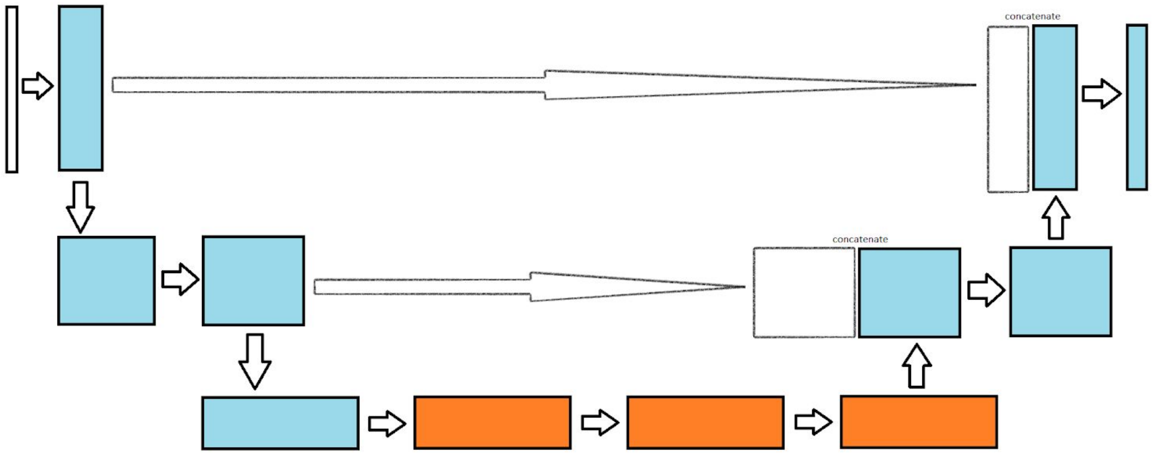
## 3.2 Networks and Training

From our perspective, we believe that the two stages of our model can be trained separately during the training process, as the overall targets for the two stages inside the model are independent to each other. We chose to complete the edge prediction network first before proceeding to the image completion stage of the model. The methodology of implementing those two steps are really similar. The only difference is the size of the inpainting generator is larger than that of the edge generator, as we consider the inpainting job harder compared to the edge predicting job.

The very first version of our implemented network was simply a series of Gated Convolutional blocks concatenated one after the other. However, this network was not even able to reconstruct the unmasked part of the ground truth edge map in the final result after several epochs of training. An example of the nonsense results produced by the model at this stage is shown below.



Due to limited access to GPU resources, though we believe that we can have the model producing reasonable results after spending more time training, we just cannot afford spending more time on waiting for such indeterministic results. In this case, we begin searching for methods to modify the model so that the model can converge to the result that we want with a faster speed. After having several days of trials and errors, we decided our model to be a Unet while having several residual blocks at the bottom of the Network.



The image above briefly shows the architecture of our network, orange blocks represent residual convolution operations while blue blocks represent the gated-convolutional blocks. Thanks to the help of residual blocks and jump connections that are happening in a typical Unet, the training process converged much faster than before. The network was able to rebuild the unmasked part of the input image in its result within the first two epochs of training. Then the training process should begin to struggle on rebuilding the masked area of the image.

Another modification we had done to the model compared to the paper EdgeConnect is we modified the input to the discriminator of the edge generation stage. In the original paper, the inputs to the discriminator while training the discriminator itself are the ground truth edge map and the predicted edge map generated by the generator. In this case, an easily identified difference between the ground truth and the generator prediction would be ground truth edge maps only having 0 or 1 as its pixels' value. However we could have real numbers between 0 and 1 as pixel numbers in our generator prediction, since we used sigmoid function as activation function at the end of the generator. If the discriminator finds this major difference and the generator tries to hide this kind of defect in its output by only outputting values close to either 0 or 1, this will make the training process of our model suffer. Specifically, this will make the gradient we get from the backpropagation step extremely small as values close to 0 or 1 can only be generated by the 'flat' region of the sigmoid function.. In order to encounter this possible issue, we also introduced small noises to the ground truth edge map before it is put into the discriminator for training. All 0s are replaced with the absolute value of a random number generated by TensorFlow's truncated_normal function with mean equals to 0 and standard deviation equals to 0.1. Similar processes are applied to all 1s in the ground truth edge map too. In the end, we will have random numbers with range [0,0.2] representing all 0s and random numbers with range [0.8,1] representing all 1s.

Overall, for the edge generator model, we take masked gray image, masked edge image and mask as input. For the image generator model, we take masked color image, predicted edge image and mask as input. Both models start with a few downsample gated convolution. Gated convolution can learn soft masks from data. It splits the convolutional filters into two parts. The first part of the output is called gating (Og), we will apply a sigmoid function σ to restrict it's value between zeros and ones. The second part of the output is called feature (Of), we apply an activation function φ (ELU) on it. The output of gated convolution is to multiply the two part together, defined as Output = σ(Og) * φ(Of).

After the downsample gated convolution, we use the residual blocks and unet network structure to save training time. By achieving this, we keep the record for the downsample convolutional output for each output size. When we are doing upsample gated convolution, we concat the current upsample gated convolution result with the previous downsample gated convolution result which has the same image size.

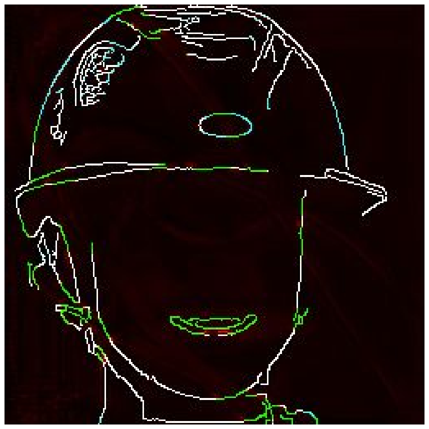
Finally, we will have several loss functions to train our model's parameters.These loss functions are feature matching, styles, perceptual and reconstruction loss functions, which allows our results closed to the real image by measuring the differences between covariances of the activation maps and comparing the intermediate layers of the discriminator. Their implementation can be found in the EdgeConnect[1] paper. As our goal is to give reasonable infilling to the missing area of the image instead of reconstructing the image accurately, the reconstruction loss is only a small part of our loss function.

## 4. Results and Experiments

## 4.1 Results

### 4.1.1 Edge Generator

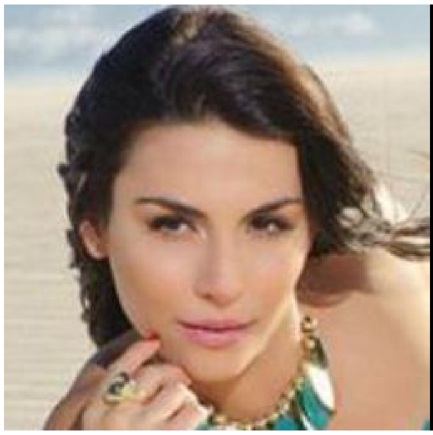
After all the efforts we have tried, we did not really succeed in the edge generation stage of the algorithm. The model did not predict reasonable edges in the masked region of the image even after 40 epochs of training on the large dataset. We think the reason for this to happen should be the generator gets trapped into a local minimum during the training process, the generator is satisfied by simply reconstructing edges in the unmasked region.

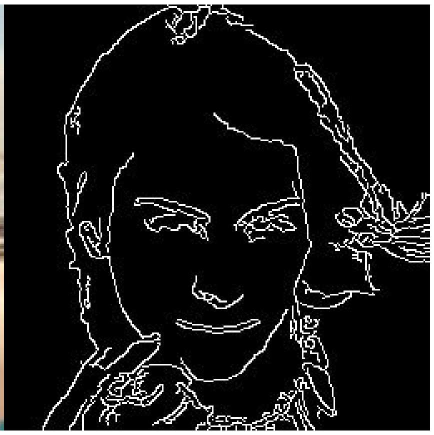


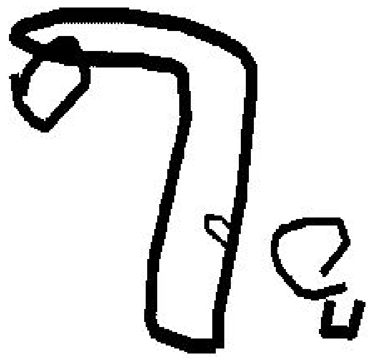
Edges predicted by the generator are represented by the red channel in the image, ground truth edges are represented by the green channel in the image, edges in the unmasked regions are represented by the blue channel in the image. We can see that the edge generator does not predict the masked edges well as those green lines should be yellow in this case. Here is a brief demonstration on what we believed had happened.

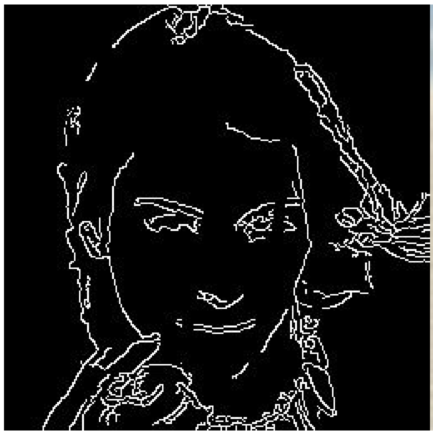
To make the generator predicting edges in the masked area, the Feature Matching loss(FMLoss)[1] should play a big role here. As the  adversarial loss should only help to make the edge generator predict reasonable edges. However, after being able to reconstruct edges in the unmasked area of the image, the FMLoss will not give enough incentive to the generator to make predictions in the masked region of the image. We think that this issue might be able to be addressed if we modify the discriminator in the edge generation stage somehow to make the adversarial loss also provide such incentive to the generator. For example, we can make masks also be the inputs for the discriminator while now discriminator can discriminate ground truth edge maps from generated edge maps by looking whether there are edges in the masked area. However, we do not have enough time to verify our guess at this point, so this can be considered as one of the further works for this project.

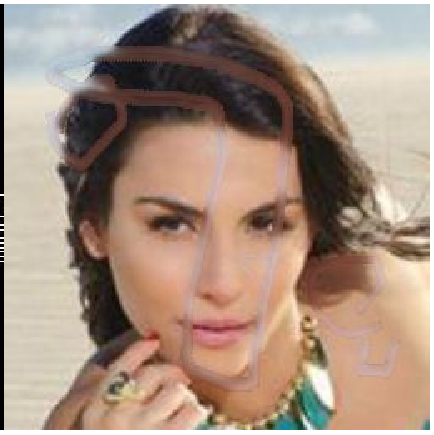
## 4.1.2 Inpainting Generator

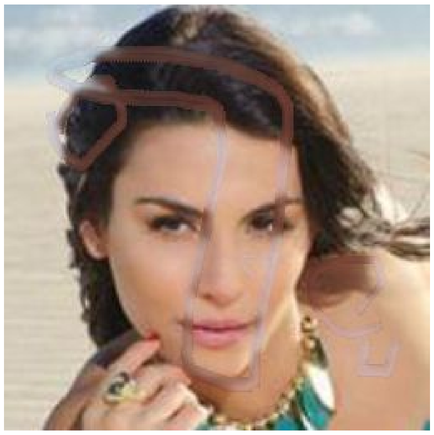

ground truth image


ground truth edge


mask

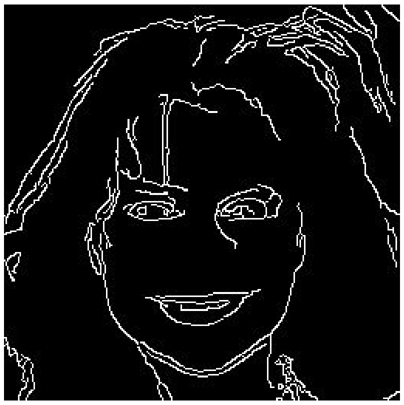

edge predicted
by edge generator

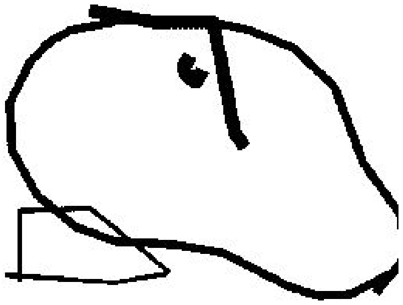

inpainting using
predicted edge map

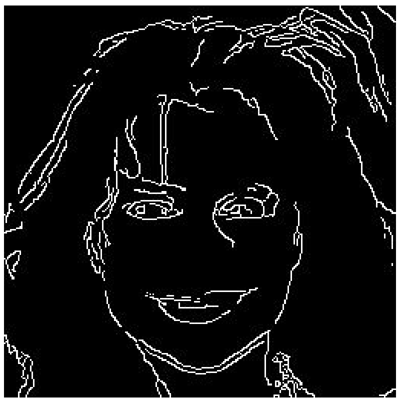

inpainting using
ground truth edge map

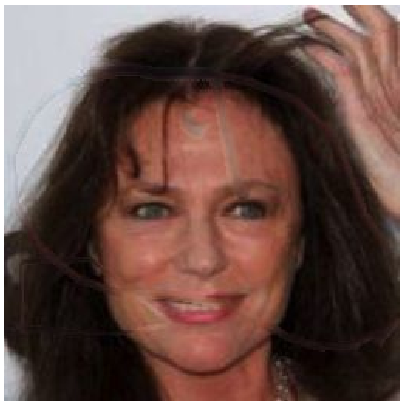
By zooming in to the nose region of face, we can see that the information provided by the edge map does get used by our inpainting generator, as the inpainting result generated by the ground truth edge map do have clearer details in that region. In other words, our inpainting model will be able to do a sketch guided inpainting after all.
However, the effect of edge map to the final prediction does seem to be minor in most cases, and we think this issue can eventually be addressed by a longer training time and more training epochs. However, we do not have enough chance to further verify our theory at this point. Here are also some other results generated by our inpainting generator.

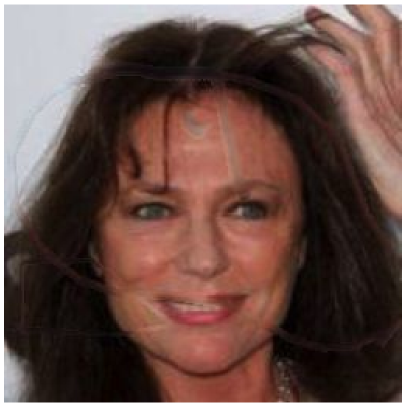
ground truth image


ground truth edge

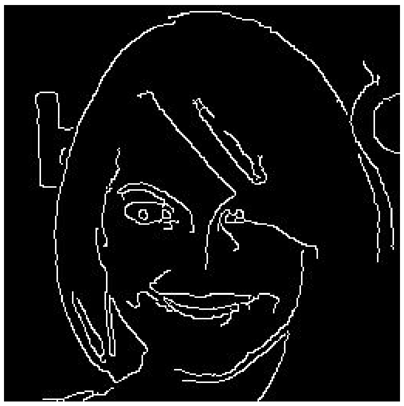

mask

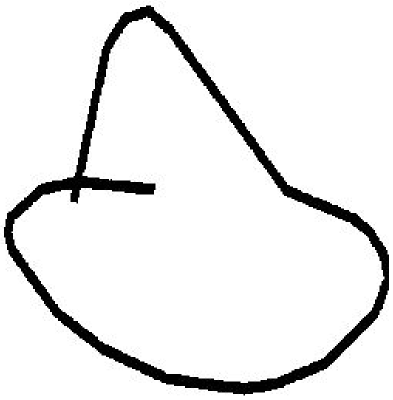

edge predicted
by edge generator

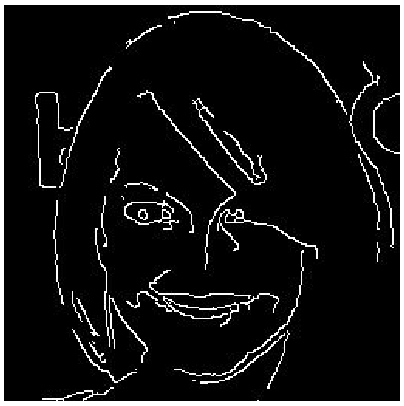

inpainting using
predicted edge map

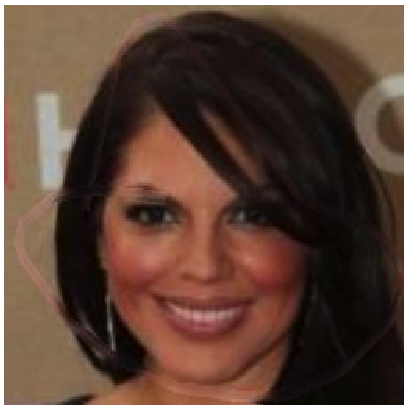

inpainting using
ground truth edge map

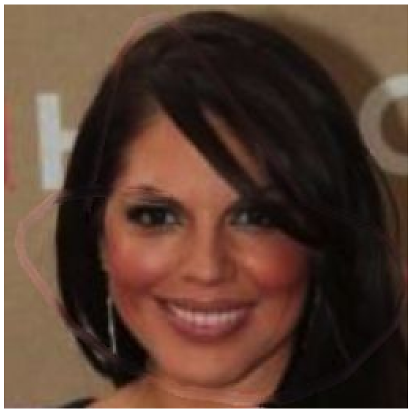
ground truth image


ground truth edge


mask

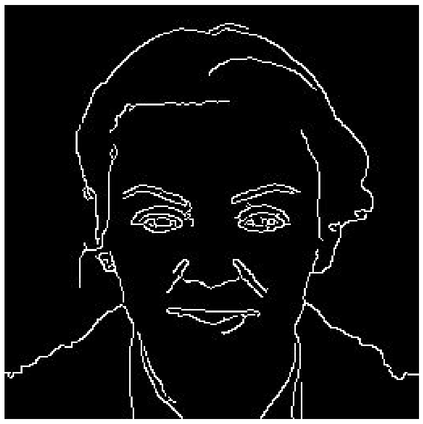

edge predicted
by edge generator

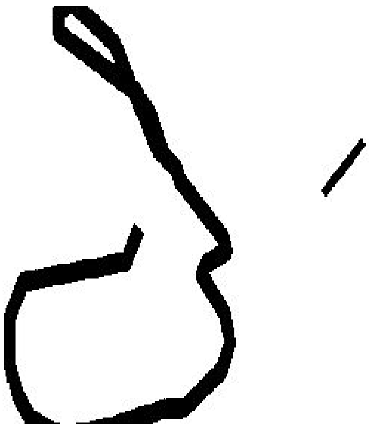

inpainting using
predicted edge map

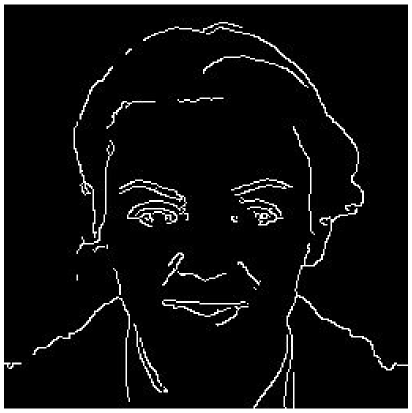

inpainting using
ground truth edge map

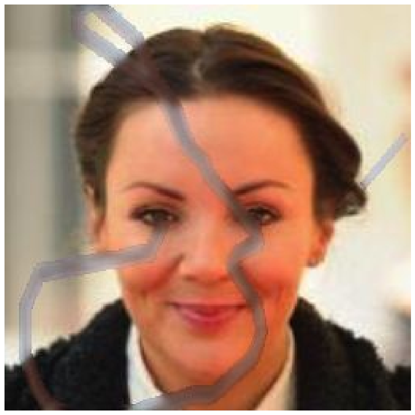
We tried many different ways to modify the parameter and the number of gated convolution, but we still can't get good predictions for some pictures. Here are some bad examples that we didn't predict successfully at the end. We think this is due to we don't have enough training time and enough gated convolution to study the feature for each channel. For example, the below prediction can't capture the color of the faces.

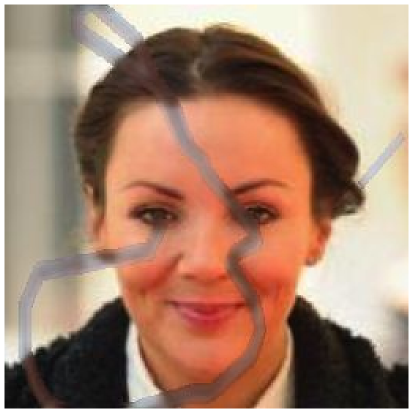
ground truth image


ground truth edge


mask


edge predicted
by edge generator

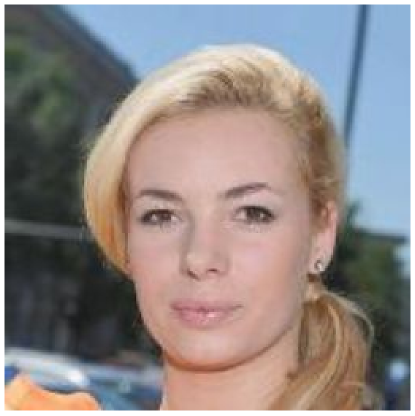

inpainting using
predicted edge map

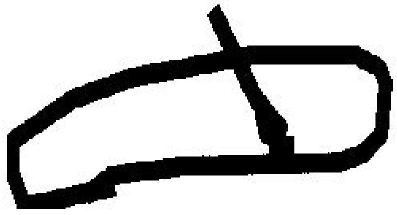

inpainting using
ground truth edge map

## 4.2 Experiments

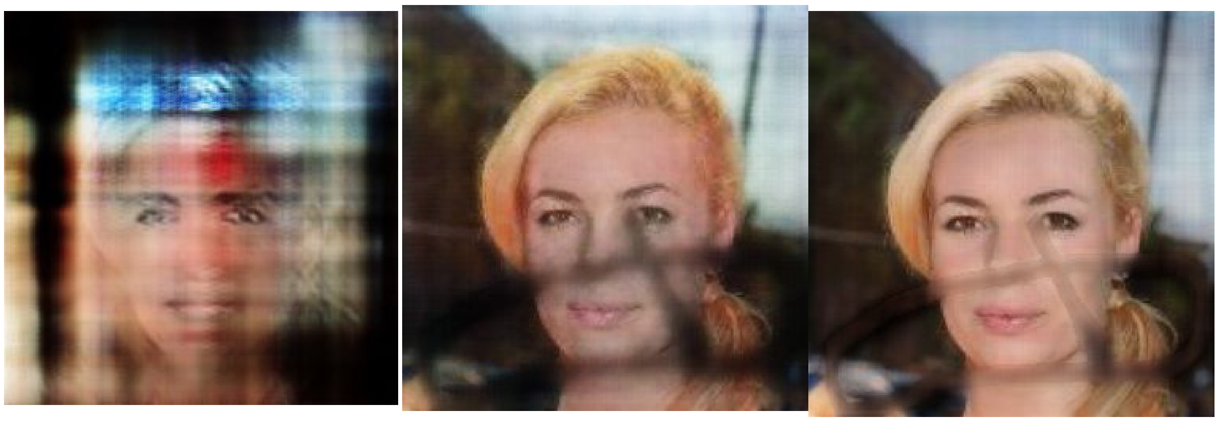
To demonstrate the effectiveness of jump connections during the training process of a neural network, we constructed three networks for the image completion stage of our algorithm. The first one is a series of gated convolutional blocks concatenated on after the other. The difference between the second one and the first one is that all gated-convolutional blocks at the bottom of the network (blocks with smallest height and width) are replaced with residual blocks. The third network has even more jump connections in the network, which formulates the original sequential network to a Unet. We built each one of the three kinds of networks above and trained those networks for one epoch over the dataset.

- With the same input image and mask below,





- Following three images from left to right are the results produced by the three networks described above respectively.



- We can see that the regular sequential network was producing nonsense results that look nothing similar to the original input image. Comparing between the second and the third image, we can see that though they both did a good job reconstructing the unmasked part of the original image, the third network began fitting the masked region of the image earlier than the second network, as the masked region in the third image began imitating the color of the surrounding region of the original image.

## 5. Conclusion and Future Work

In our project, we achieved image inpainting for any irregular shapes masked on people's faces using edge-prediction model and image-inpainting model. For both models, we use gated convolution operators since it gives us a built-in contextual attention module for accurate inpainting results. We also use U-net for our network structure, which helps us obtain our model with fast convergence and precise image results. Even though we generate reasonable inpainting image results in our project, we could still tune different parameters for our models to achieve a better result in the future. Moreover, due to the limitation of GPU, we don't have a chance to train our model with a different dataset Places365[6]. In the future, we can train our model by combining different datasets to see if we can obtain a more general inpainting model not only focused on faces.

## Authors' Contribution

Chenyue Wang is responsible for finding datasets, preprocessing data and generating dataset needed for the edge generator and image generator model respectively. Keying Chen is responsible for constructing the edge generator and image generator model. Shuhao Li is responsible for the whole training process, the loss function part and modifying the code for the model part. Everyone is responsible for the report and the video presentation.

## References

[1] Kamyar Nazeri, Eric Ng, Tony Joseph, Faisal Z. Qureshi, and Mehran Ebrahimi. “EdgeConnect: Generative Image Inpainting with Adversarial Edge Learning” arXiv:1901.00212v3 **[cs.CV],** ***11 Jan 2019.*** ***https://arxiv.org/abs/1901.00212v3***

[2] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. “Deep learning face attributes in the wild” arXiv:1411.7766v3 **[cs.CV],** ***24 Sep 2015.*** ***https://arxiv.org/abs/1411.7766***

***[3] Karim Iskakov. “Semi-parametric Image Inpainting” arXiv:1807.02855v2 [cs.CV], 13 Nov 2018. https://arxiv.org/abs/1807.02855***

[4] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. “Free-Form Image Inpainting with Gated Convolution” arXiv:1806.03589v2 **[cs.CV],** ***22 Oct 2019.*** ***https://arxiv.org/abs/1806.03589***

[5] Olaf Ronneberger, Philipp Fischer, Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation” arXiv:1505.04597v1 **[cs.CV],** ***18 May 2015.*** ***https://arxiv.org/abs/1505.04597***

[6] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, Aude Oliva. “Places: A 10 Million Image Database for Scene Recognition,” **in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.40, no.6, pp.1452-1464,** ***1 June 2018,*** **doi: 10.1109/TPAMI.2017.2723009.**