CSCI 1430 Final Project Report: Image Segmentation and Layer Replacement

Team Sun Segmentation: Cameron Fiore, Colin Jones, Jiaqi Zhang, Martin Gil del Real.
Brown University

Abstract

In this project, we propose a novel automatic photo editing tool. This tool takes in visual data (i.e. a photo or a video) and, using a deep-learning model, automatically segments the input by the different objects that it recognizes. The system then separates the original input into "layers", with each layers corresponding to a recognized object in the input. The user can then select which object in the input to modify, and our tool automatically replaces the selected object with an image of user choice.

1. Introduction

In this project, we essentially tackled the process of "masking" found in many image manipulation programs, and we do it via layer-replacing and image segmentation. Through semantic segmentation, we can automate (albeit with a little bit of trust) the process of selecting an object in an image or video (this is very tedious and hard to do with video) and with the obtained area of where an object is meant to be, then replace it with another image or video.

In principal, via the segmentation, we obtained a matrix of size 320, 320 (the size of the resized input image) which is populated with different integers representative of the object that occupies that pixel, and for every different integer, we create a "layer" that can be replaced with another 320, 320 image (which we refer to as our secondary inputs). For video, it is the same process, simply with multiple frames, and the ability to properly input videos as secondary inputs.

To build on this program, we could modify it so that besides layer-replacing with another image/video, one could layer-replace with a style transfer of the original image with a input image which dictates the style. So image a picture where the only two recognized objects are the person and the background, with this new addition, only the person (or the background, depends on you) will get style transferred, whilst the background (or person) stays the same.

2. Related Work

Our project relies heavily on the computer visions task of semantic segmentation. Semantic segmentation is the task of taking an image and classifying each pixel with the object class that it captures. Recently, deep learning models have been demonstrating state-of-the-art performance for this task. One such model that we first attempted to train is called U-Net [3]. U-Net was originally designed for segmenting biomedical images. Another model with which we experimented is called Seg-Net [1]. Seg-Net was designed for more common, every day image segmentation tasks. Both models are comprised of a decoder (convolutional and down-sampling layers) followed by an encoder (convolutional and up-sampling layers). Both of these models require a lot of time and data to train, so we ended up using pretrained weights for our project taken from here.

3. Method

Our method is split into two parts: a backend algorithm to segment out objects and a frontend algorithm to replace objects with user-specified images.

3.1. Backend: Image Segmentation with SegNet

For the backend, we chose a deep learning network, called SegNet [1] to help us segment objects by classifying pixels. The architecture of SegNet is shown in Fig. 1 (adopted from [1]). Specifically, given an image, the SegNet uses an encoder-decoder structure to classify every pixel with a label (e.g., person, cat, etc.). The encoder contains five convolutional blocks, while each block is a sequence of convolutional layers, a batch normalization layer, and a max-pooling layer. Activation functions are selected to be the ReLU function. Then, for the decoder, it reverses the encoding process by upsampling and convolutional computations. The only difference is that the last activation function is the softmax function since the network should classify each pixel. After one forward pass, the network outputs the label of each pixel.

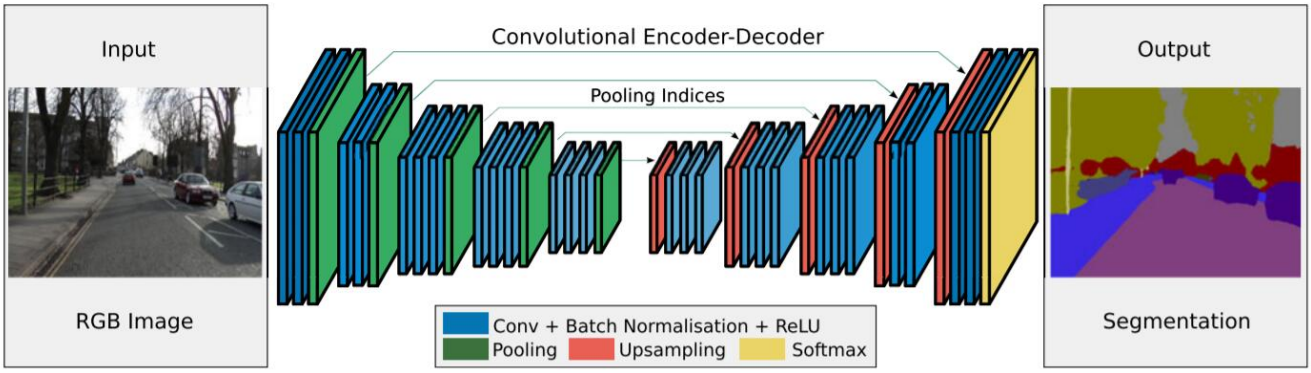

Figure 1. The architecture of SegNet (adopted from [1]).

3.2. Frontend: Layer Replacement and GUI

For the frontend, we created a GUI to allow the user to interactively implement our layer replacement algorithm. From the GUI, the user can upload an input file and segment it. They are then given a listing of the different layers which have been recognized by the backend, and they have the option to upload images or videos to replace any of these layers. The layer replacement algorithm works by using the segmentation from the backend to isolate a 2D matrix for each layer. For example, in a picture with a person, the isolated 2D matrix for the person label would have 1s where the person is and 0s where the person isn't. Then, for each layer selected from the GUI, we multiply the inputted "swap" image by this 2D matrix, and add up all of the layers to get the output image, as seen in Fig.

4. Results

4.1. Image Segmentation

We implemented a SegNet network with Tensorflow and train it on the ADE Challenge Data [4]. The dataset contains 20210 training images and 2000 validation images. There are 151 labels in total for the dataset. We train the model on the RectifiedAdam optimizer with learning rate$= 10^{-3}$ and loss function as sparse categorical cross-entropy.

The model reached 40% accuracy after 40 epochs. To further improve it, we freeze some layers during training. Specifically, train all the layers for 40 epochs, then freeze the first three encoding blocks and train for 40 epochs, then unfreeze all the layers. Repeating this process, we improve the accuracy to at most 54 %. The accuracy seems fine for over 100 labels, but the segmentation result doesn't look good.

So we moved to a pre-trained SegNet model that was trained on the PASCAL VOC dataset [2] with 21 labels. This pre-trained model achieved over 90%. We planed t fine-tune this model on our dataset, but we had no time to do this. So this would be the next step to do in the future. With this pre-trained model, the segmentation is more stable and reasonable.

4.2. Layer Replacement

The layer replacement algorithm ended up working very well. It does its job as expected, and is fully dependent on the accuracy of the semantic segmentation. If specified, a specific layer will not be replaced by an alternative secondary input, but instead will leave the original image intact (in said specific layer). For a video result, please follow this link In this result, the label "background" was successfully replaced with another video, and all other recognized labels (there were many besides person), were left the same.

4.3. Technical Discussion

What about your method raises interesting questions? Are there any trade-offs? What is the right way to think about the changes that you made?

Answer: For the back-end model, we actually trained a UNet first, but it has low accuracy. So we moved to the more complicated SegNet. It needs more time to train but achieves higher accuracy. Also, we froze some layers during training to avoid overfitting.

In the layer replacement algorithm, we had to face a decision of what order to "layer-replace" for a video: [1] either looping through each layer, and do all frames for said layer at once, or [2] looping through each frame and layer-replace each layer for said frame at once. Looping through each frame gave the benefit of allowing us to immediately compress the many np arrays representing images for each "layer", into one final image for the frame, but we'd have to continuously open and close image and video data for the secondary inputs for each frame. Whilst looping through each layer, allowed us to simply open one image/video file at a time, go through all its frames, then close it, but we'd have to remember those results until the very end, when we can compress into the final frames. We were unsure

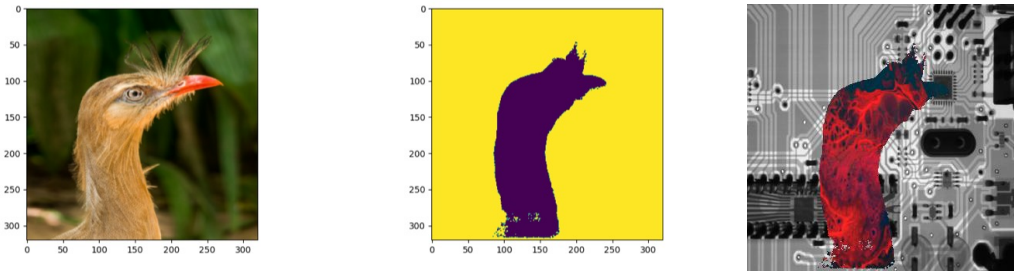

Figure 2. A process through the segmentation, first is the inputted image, then is the ”layer” matrix of 1s and 0s, finally is the layer replaced image

of which one would take up more memory and slow the algorithm. Ultimately, this could probably be solved with some smart matrix operation, but we opted for looping to ensure it worked first.

4.4. Societal Discussion

Please respond to the following questions. Different projects will have different scales and qualities of impact; we ask you to think creatively and consider the broader implications of your project rather than just the more narrow current technical capability. Responses should together take up roughly one page in your final report.

1. Describe the socio-historical context of your project to identify three broad societal factors that could affect your data, goal, and/or hypothesis. These factors might include current or historical policies, events, social conditions, and larger societal systems. Cite at least one outside source.

   A1: We aim at developing an application for public use. So the user requirement definitely will affect our goals. Replacing some parts in images is a kind of manipulation. It will produce a powerful tool in creating fake photographs. In our Hw1, we've discussed the problem of manipulating images regarding this topic. So the public concerns on this manipulation may also affect us on the algorithm design.

2. Who are the major stakeholders in this project? What is your relationship to these stakeholders?
   Stakeholders are those who may be affected by or have an effect on your project topic. Some examples of stakeholders are a particular demographic group, residents of a particular geographic area, and people experiencing or at risk for a particular problem.
   Consider the following questions to help identify stakeholders:

   - Who does this project topic currently affect? It is difficult to say, since it is a visual art based program.
   - Who might be harmed by your research findings?
   - Who might benefit from your research findings? In terms of the layer-replacement, a very specific artist will benefit from the architecture of this program :)

   A2: The stakeholders in this project are any person or group that wishes to take an existing image and modify it in some way via layer replacement. However, depending on the intent of a particular user, the result may vary. For instance, an artist can use our model to generate stylish pictures. A potentially malicious user will use our GUI to make fake images for unethical purposes. If our model were to be deployed as a public app, we may have to add certain safeguards to our algorithm in order to prevent these malicious acts by use of our project.

3. Research or journalism on your broader project topic may have already been conducted. What was the societal impact of existing research? Discuss the implication of this research on your project and consider the following questions to help identify at least one implication. It may affect:

   - How you should frame your goal,
   - How you should design your algorithm,
   - How you should analyze your data,
   - How you should interpret your findings, and
   - How you should present your results.

   A3: There are countless examples of deep learning algorithms, especially object detection and labeling systems, producing discriminatory output, as did this google AI system. Classification models must be heavily supervised, screened, and tested before they can be released for public use. For our model, the existing research emphasized the fact that we should 1) ensure

our train data is non-discriminatory and 2) we must adequately test our output not only for accuracy but also for fairness and inclusion. We made sure our data was as diverse as possible and rid of any observable bias. We also tested our model on images of people from different racial backgrounds to make sure the outputs were inclusive and fair.

4. How could an individual or particular community's civil rights or civil liberties (such as privacy) be affected by your project?

   A4: Our project does not directly affect a particular individual's or community's rights. We do not save any user-provided information. All our system does is take user input and give it right back to them, only slightly modified. However, the technology that our project is built with, i.e. a deep semantic segmentation model, may be used in a way that can affect user rights. For instance, if used as a surveillance technique, semantic segmentation can identify any object in surveillance video, a notion that American citizens may not feel comfortable with.

5. If you are using data, what kind of biases might this data contain? Do any of these represent underlying historical or societal biases? How can this bias be mitigated? Consider the following questions to help you:

   - Were the systems and processes used to collect the data biased against any groups?
   - Is the data being used in a manner agreed to by the individuals who provided the data?

   A5: Our data is taken from open sourced data-sets [2] [4] used by many other researchers in the field. In addition to our inspection for bias, these public data-sets have been vetted and tested by many other researchers, and a large number of published papers have cited these data-sets. We believe that these data-sets are therefore as unbiased as possible due to the amount of testing being performed not only by our project group but also by the countless others in the field. The providers of the data designed the public data-sets for use in object-recognition and segmentation challenges, so our project is very much in line with this intended use.

5. Conclusion

In this project, we created an app which segments images and allows you to replace any given layer. While it is not perfect, our project has a lot of potential to create some really cool art pieces, which was one of our goals from the start. Semantic segmentation is a powerful emerging technology, and we only scratch the surface in this project. However, we have successfully combined deep-learning, computer vision, and photo-editing techniques into one app that is easy to use and that hopefully demonstrates to not just computer scientists but all users the usefulness and versatility of the technologies involved.

We hope the impact of the tool we designed in this project is on the whole positive. We like to believe that we have created a user friendly way of taking visual data and easily modifying it for artistic and creative purposes. Perhaps this technology would be best utilized in already existing apps like Illustrator or Zoom, where users could easily layer replace existing photos or even their video streams. While there is always a chance that certain individuals may take a tool designed for good and use it in a malicious way, we believe our project entices only those with creative intentions to augment their digital portfolio.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017. 1, 2

[2] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015. 2, 4

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[4] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 2, 4

Appendix

Team contributions

Martin Gil del Real I worked on the bulk on the layer replacement algorithmn, alongside some help parsing through things with Colin. Me and Colin collaborated and crunched how we would obtain information from the front-end GUI and properly process it into both the segmentation back-end, and the layer replacement back-end.

Colin Jones I worked mainly to build the GUI using Flask. This involved writing the server.py file and the index.html file. I worked with Martin to connect the back-end, namely through the layer replacement algorithm, to the frontend.

Jiaqi Zhang I worked with Cameron to reimplement the backend deep learning model, the UNet and SegNet. This involved coding and debugging with Python and Tensorflow, and running codes on GCP.

Cameron Fiore I worked closely with Jiaqi on the "backend" of our project. This involved conducting research on the best image-segmentation architecture for our deep learning model, implementing the architecture via tensorflow, debugging, and training the models on GCP.