Topic: Background separation with Deep learning.

Members: Tarik Koric (koric1@illinois.edu), Pranav Velamakanni (pranavv2@illinois.edu)

Course: CS445 - Computational Photography (Spring 2020)

The topic chosen for the final project builds upon various key subjects in the course while also using cutting edge learning technologies to provide a comprehensive solution to finding an object. In this case the objects of a human, cat, and dog are isolated from the rest of the image. A deep learning solution was implemented with a couple thousand images as trained data to create a model sufficient enough for use. Similar approaches like Graph Cuts and Intelligent Scissors that were discussed before rely much on user input to isolate an object in an image. With the new approach, an image containing these objects would be inputted into one of the learned models and an image with the isolated object would be outputted without any background. A final command line application was created that uses the trained models to provide an outputted image given an input. Various softwares like Photoshop can be used with a similar approach with very little user input. This would allow for faster working environments with minimal human error for image editing softwares. Another important aspect of this tool would be for classifying images. Being able to first isolate an object in an image would allow for easier and more accurate classification models to be created. These kinds of applications can be used in things like self-driving cars for object detection.

Before the actual model was trained, training data needed to be compiled and cured. A dataset of humans, cats, and dogs was downloaded from Kaggle. Each image was then preprocessed to a size of 256x256 pixels. An output image with a mask for the detected object was also needed to be created. In this case, the select object tool in Photoshop was used and automated among all the training data to create the specific masks. The 256x256 non altered image would be used as the input data and the masked binary image would be used as the output. Figure 1 shows an example input and output. Tools that were used throughout the project largely consisted of python programming and its various packages. Keras and Tensorflow were used to create and test the models. Other packages like numpy and pandas were used to preprocess the images. Scipy was used to scale the images to the correct 256 x 256 format. Finally, Photoshop was used to create the masks for the training and test data.

The training for the project was performed by U-nets. U-nets were first proposed in 2015 in the research paper sourced below to perform image segmentation on medical data. U-nets leverage the use of convolutional neural networks (CNN). The term “U” in U-net comes from the U shape of the network architecture which is shown in figure 6. The architecture is symmetric and has 2 parts, shown in figure 6. The section on the left is a general convolution process which handles the image downsampling. The section on the right contains transposed 2D convolution layers which handle the image up sampling. Max pooling layer at each convolution step halves the image which reduces the size of the image to half at each step. This step is performed three times on the left to down sample the image. The goal of the downsampling process is to capture as many features of the source image as possible. After this process, the middle section is reached where the data is passed through two additional convolution layers before the up sampling process starts. Because the goal of U-nets is to produce an output that matches the source image, up sampling is performed to re-create the image to match the shape of the source image. Simply put, the downsampling process gathers the “WHAT” aspect of the process that is required to identify the classes present in the image. The “WHERE” aspect of the process is handled by the up sampling process which converts the low resolution image to high resolution to ensure that this information is

retained in the process. Since the goal of this project is to isolate backgrounds, the final image produced by this model is a mask which shows the predicted background marked with a pixel value of 0 and the object with a value of 255. A bitwise operation is performed on this predicted mask and the source image to result in the final RGB image where the background is removed. The app we designed for this project loads the trained models and provides an interface for users to process new images. For information on how to use the app, refer to the readme.md file included in the project.

Accuracy was calculated by splitting the original data into training and testing. Once each of the models were trained, they were tested using the testing data and an accuracy score was given. The accuracy of each model was as follows: U-net with all images - 79 %, U-net with people images - 87.95 %, U-net with cat images - 78.74 %, U-net with dog images - 80.27 %. The models were also tested with images that were neither in the testing nor training data. Figure 2 shows the same image being inputted into four different applications. Two of those were the ones created using the U-net models described earlier, U-net with all images, and the U-net of just people images. The other two applications use Photoshop's select object tool and a specific website that specifically deals with removing backgrounds from images, known as remove.bg. The two models described in the paper do fairly well against the two professional applications. Figures 3 and 4 show that same comparison with the use of cat and dog images. Finally, figure 5 shows an image that did not work as well using the U-net models. This can be attributed to the similar contrast between the person in the image and the background. Figure 5 also shows the comparison against the professional software.

There were lots of challenges to the project with getting the mask of the data. Photoshop has a tool that allows for batch preparation of images that allowed for preprocessing of the output images. While decently accurate, Photoshop sometimes gives data that is not as accurate to the naked eye. The ideal scenario would have been to use masks from the Remove.bg website. However, it is a paid service and their API only works for 50 images per month. This would have been too small of a data set to train. The models were trained with around 800 images for each model—person, cat, dog, and around 2,400 images for the all model. Given more time and a larger dataset, all four models trained would have been probably more accurate. The problem with more images is that more time is needed to train the actual model. With a higher end CPU, it took up to 4 hours to train one model alone. Better equipment like a dedicated GPU would have helped.

The project at hand was very innovative and used cutting edge technologies that are the most up to date for image segmentation. The U-net model used to train the data was published recently in 2015. The project also gives people the opportunity to download a portable command line application that anyone with the correct dependencies could test out themselves for free. The project team expects to receive full marks for all of the hard work put into the project.

Figures

FIGURE 1

FIGURE 2

FIGURE 3

FIGURE 4

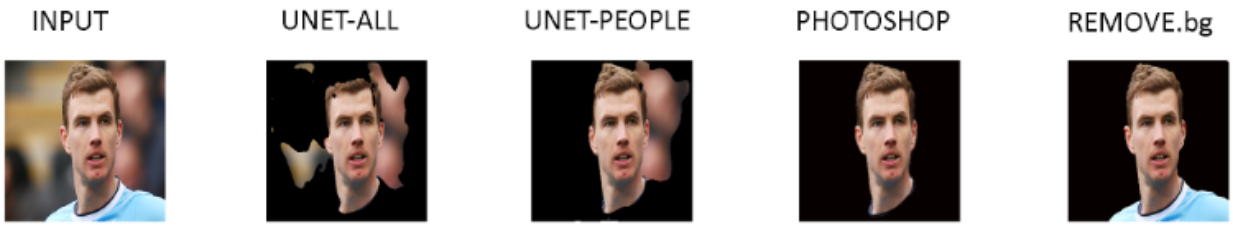

FIGURE 5

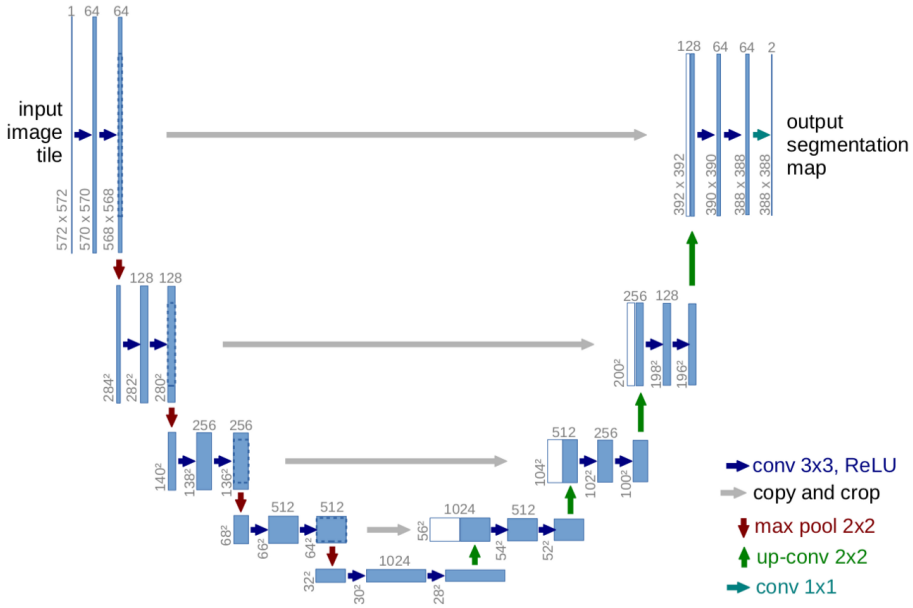

FIGURE 6

Sources

Image DATASET

https://www.kaggle.com/prasunroy/natural-images

UNET PAPER

https://arxiv.org/abs/1505.04597