CS230

Object recognition, suppression, and image inpainting

Elise Fournier-Bidoz
Department of Aeronautics & Astronautics
Stanford University
efb@stanford.edu

Alexandre El Assad
Department of Aeronautics & Astronautics
Stanford University
aelassad@stanford.edu

Abstract

Our project focuses on the operation of removing human shapes from pictures. Today this operation can be done using the commercial software Adobe® Photoshop® in about 20 min. Our approach combines two neural networks - YOLO and Context Encoders - and adapts them to our task. The first neural network is used to detect and draw a bounding box around the human(s); all pixels within this bounding box are then removed and refilled by the second neural network. We have achieved satisfactory results for both tasks separately, however, the combination of the two neural networks to create the full pipeline can still be improved.

1 Introduction

Have you ever found yourself incapable of taking a picture of a beautiful place because too many tourists were clouding the scene ? We certainly have. Ideally, automatic removal of human shapes from pictures would be performed on a user-interactive website where people would upload a picture of a partly hidden landscape and receive a clean picture in return. This would eliminate the need to ask your Photoshop ® expert friend to spend 20 minutes correcting the picture by hand.
Our project combines two interesting aspects of Deep Learning which are object (in our case human) recognition and removal, as well as picture filling. The input of our algorithm is an image, typically a landscape, on which a person appears. We then use our set of two neural networks to remove the person and refill the missing pixels to match as well as possible what the original picture should have been without the person.

2 Related Work

Several software currently enable anyone interested in photography to remove people from pictures such as Adobe Photoshop ® or Paint3D. However, these software do not completely automate the task since they require the user to draw the outline of the object to be removed. We intend to make this task completely automatic without the need for a pre-selected area.

There exists several neural networks for the image segmentation task such as Detectron [1], a state-of-the art platform for object detection research developed by Facebook, built on Caffe2. Moreover, FCN-8 [2] is a fully convolutional network for semantic segmentation. It adapts contemporary classification networks (AlexNet, VGG net and GoogLeNet) into fully convolutional networks and transfers their learned representations by fine-tuning to the segmentation task. These neural networks detect the precise contour of the object rather than simply drawing a bounding box around it.

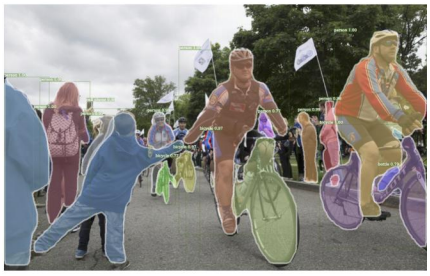

Figure 1: Example of Detectron output

Semantic Image Inpainting with Deep Generative Models [3] reconstructs the missing content by conditioning on the available data, meaning that inference is independent of the missing content's structure. Existing methods which extract information from a single image generally don't perform as well due to their lack of high level context usage.

3 Dataset and Features

3.1 Data collection

We collected data for both neural networks using the same method:

- A web scrapping script was used to download Google images based on a query such as "landscapes with people".
- Data augmentation (random cropping, flipping, change colors) was performed on these images to increase the size of our dataset.

3.2 For the Object Recognition Neural Network

The dataset for YOLO consisted of 90 images. We did not need to split the data set into train/dev/test as YOLO already performed well on our images. As explained in the results section, we did not have to retrain the neural network, the results were already very satisfactory.

3.3 For the Filling Neural Network

Our dataset consisted of 240 images, split into:

- Train set: 70 % (168 images)
- Dev set: 30% (36 images)
- Test set: 30% (36 images)

Each image was 128*128*3 (RGB colors) and we performed data augmentation to increase the size of our dataset. No labelling was necessary as the inpainting neural network was trained with unsupervised learning.

4 Method

Our approach combines two neural networks which have proven to be efficient and fast for our specific application. Furthermore, a big part of our project was the ability to combine these two neural networks, and we made the decision to couple YOLO [4] with Context Encoders [5].

4.1 YOLO9000

Ideally, we would've deleted the pixels within the contour shape of humans, but we quickly realized that retraining FCN8 would require much more data than we could possibly label. Moreover, testing our filling Neural Network showed that it had good performance on masks as large as a quarter of the

total image area, which implies that minimizing the mask size is not a priority. We therefore chose to use the YOLO algorithm, studied in class, which is already trained to do what we want with excellent performance. The disadvantage of YOLO - that we are deleting more pixels than required (the entire rectangle instead of just the human shape) - turns out to be irrelevant for our project.
We modified YOLO such that in the output images :

- only bounding boxes detecting the class "human" are kept,
- the pixels contained in the bounding boxes are set to one (turned into white pixels).

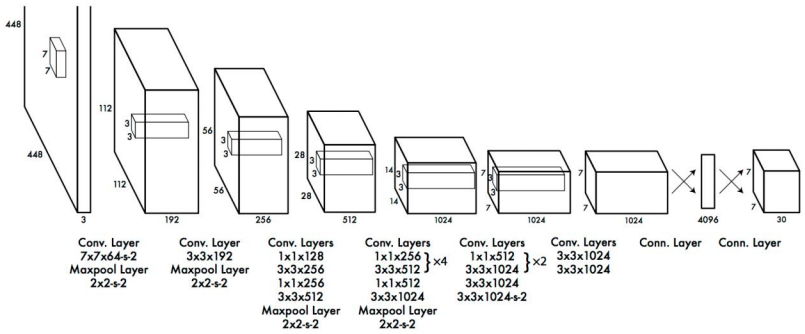

Figure 2: YOLO Architecture

4.2 Filling neural network: Context encoders

Our filling neural network, described in [5], is based on context encoders, a convolutional neural network trained to generate the contents of an arbitrary image region conditioned on its surroundings. It inputs a 128*128 image containing a mask of 32*32, where all pixels are set to 1. It outputs the 32*32 mask in which the image has been reconstructed. The encoder-decoder, which uses the L2 loss to reconstruct the image, is part of a larger GAN architecture. More specifically, the bottleneck network is the generator whose purpose is to propose an image. This proposed reconstruction is then fed into the discriminator, as well as the true image.

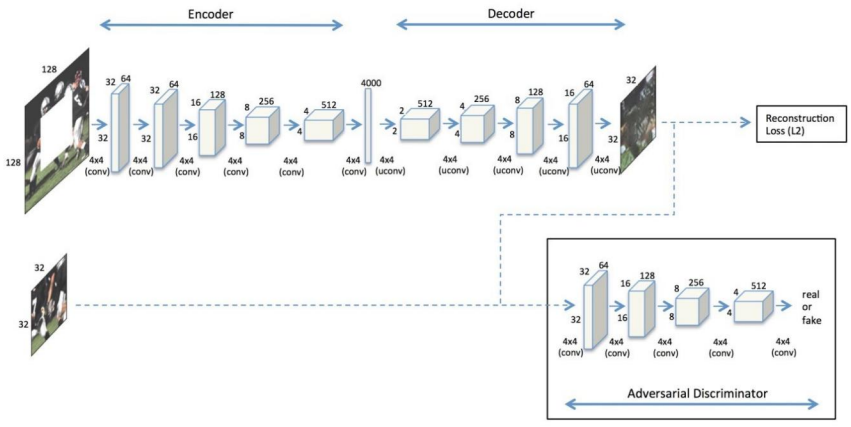

Figure 3: Context Encoder Architecture

5 Experiments / Results / Discussion

5.1 YOLO results

Before doing any transfer learning, we tested YOLO on our dataset. Results were satisfactory as shown on Fig. 11 & 12.

5.2 Inpainting Results

Most of our work was focused on training the inpainting neural network on our own dataset. The learning task was unsupervised since a random patch was automatically applied on the image before the forward pass through the neuronet. Fig. 6 and 7 show the training loss over 100 epochs and 500 epochs, both run with the following parameters:

- Batch size: 64
- Learning Rate: 0.0002

Dataset	L_2 loss
Train	0.07
Dev	0.13
Test	0.15

Figure 4: Final loss after training on 100 epochs

Dataset	L_2 loss
Train	0.058
Dev	0.08
Test	0.15

Figure 5: Final loss after training on 500 epochs

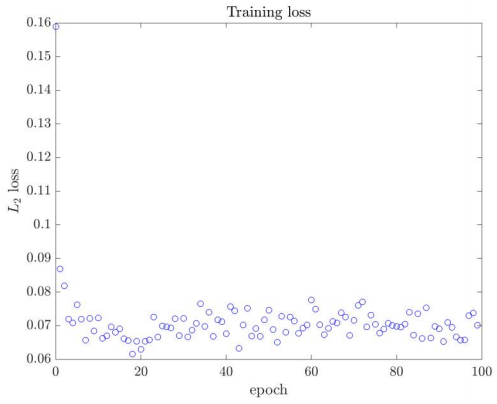

Figure 6: Filling Neural Network training loss on 100 epochs

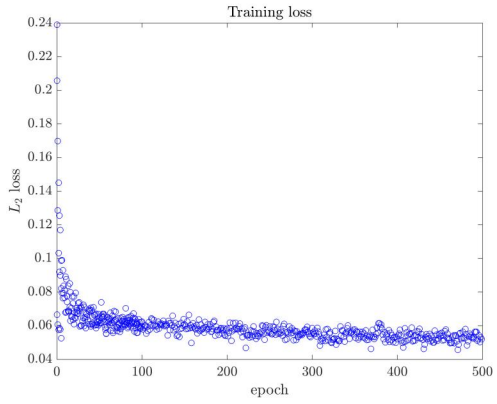

Figure 7: Filling Neural Network training loss on 500 epochs

As seen in Table 4, the dev and test losses are considerably higher than the train loss, meaning we are overfitting to the train set. Table 5 shows that the reconstruction loss on the test set is too big compared to the train and dev losses, so here we are overfitting to both the train and dev sets.

Fig. 8, 9 and 10 show the entire pipeline for an image of the dev set, passed trough the network trained on 500 epochs. The results are very encouraging on dev set images.

Figure 8: Original dev image

Figure 9: Masked dev image

Figure 10: Final dev image

5.3 Final results

Fig. 11, 12 and 13 show the entire pipeline. Filling results are far from perfect on test images.

Figure 11: Original test image

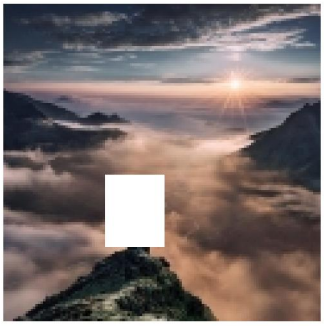

Figure 12: Masked test image

Figure 13: Final test image

6 Conclusion and future work

In conclusion, we have demonstrated the ability to automatically remove a human shape from a picture. However, better results can still be achieved.
Some possible improvements include collecting more training data or using early stopping in order to avoid overfitting to the train and dev sets, as we saw that the test loss is considerably higher than the train and dev losses. Performing human detection and removal with an image segmentation neural network such as Detectron would help reduce the size of removed areas, thus possibly improving the filling task. Retraining the filling neural network on human or random shaped masks would then be necessary in order to obtain realistic image reconstructions.

7 Contributions

- Elise: Data augmentation, filling NN code search, train loss plots and error analysis.
- Alexandre: YOLO implementation and modification, image inpainting implementation, data collection.

8 Code

Our code is available on Github at `https://github.com/aelassad/AuRePOP`

References

[1] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. `https://github.com/facebookresearch/detectron`, 2018.

[2] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[3] Chen Chen Raymond A. Yeh and Teck Yian Lim. Semantic image inpainting with deep generative models. *arXiv:1607.07539v3*, 2017.

[4] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[5] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.