Exploration Map Inpainting using Partial Convolution

Manish Saroya
Oregon State University
saroyam@oregonstate.edu

Parijat Bhatt
Oregon State University
bhattpa@oregonstate.edu

Abstract

The objective of the paper is to predict map structure across a frontier in robot exploration problem. Map structure prediction in a partially known map could then be used by decision-making algorithms which try to maximize information gain over fuel cost. This paper uses U-NET like architecture with partial convolutional layers to predict unknown areas in partially known subterranean tunnel maps. We demonstrate that our approach can successfully predict intuitive connections between unexplored tunnels.

1. Introduction

This project is motivated by the ongoing work on the DARPA Subterranean (SubT) Challenge [2]. In this challenge, robots will be exploring unknown underground environments, such as natural caves or man-made tunnels. They will be tasked with discovering points of interest, such as humans in need of rescue or incendiary devices that need to be disabled. The goal is to:

- Maximize the information gained over the travel cost
- Minimize the time it takes to find points of interest.

Efficient robot exploration is one of the fundamental problems in robotics. Lot of work has been done in predicting the map across a frontier. In [11] the authors maintain a library of map structure to infer over an unexplored region beyond a frontier and merge the inference into the robot's exploration map. This approach requires to maintain a big library and inference over it while the robot is exploring, this is computationally expensive. [5] uses a deep neural network to select frontier by avoiding computationally expensive ray casting for computing the approximated information gain. This approach, however, does not predict the region beyond a frontier leading to inefficient exploration.

Our approach predicts regions across frontiers which can be used to compute information gain. We train U-Net like architecture with partial convolutional layers to predict the unknown environment. Firstly, we procedurally generate a database of maps and then create masks over it to make it partially known map. This hidden information is then predicted by our network. While testing, the deep neural network outputs the prediction in constant time. This can be used by the robot during exploration.

2. Background

Although CNN has been around for quite some time, [9][8] made a breakthrough in machine learning by winning the Imagenet challenge. With the advent of the internet it has been possible to gather more and more data which in turn has made it possible to train CNN by using GPU's thus giving high accuracy in a variety of image recognition tasks. One such is the task of image inpainting. Often times, we come across blurred images or image with holes and we would like to restore these damaged images or improve their texture. It has been a challenging problem in the CV community for quite some time and it has only now been possible with the use of deep networks to obtain good results. It is important to note that old image techniques of denoising algorithms do not completely apply to image inpainting. Previously tried approaches involved using restoration of films, texture synthesis etc. [4] use motion estimation and autoregressive models to interpolate losses in films from adjacent frames. Other texture synthesis algorithms can be used to complete a region to be inpainted. PatchMatch[3] , one of the widely known methods, iteratively searches for the best patches to fill in the holes. While this approach generally produces smooth results, it requires domain knowledge. A limitation of previous approaches has been in using a fixed location for patches. We believe these limitations can lead to overfitting. We don't use the case of irregular holes but our hole's location is randomly selected for each image in the dataset.

3. Methods

In this section, we first explain how synthetic map database and mask for each image is generated followed by our architecture and finally the loss functions used in our model.

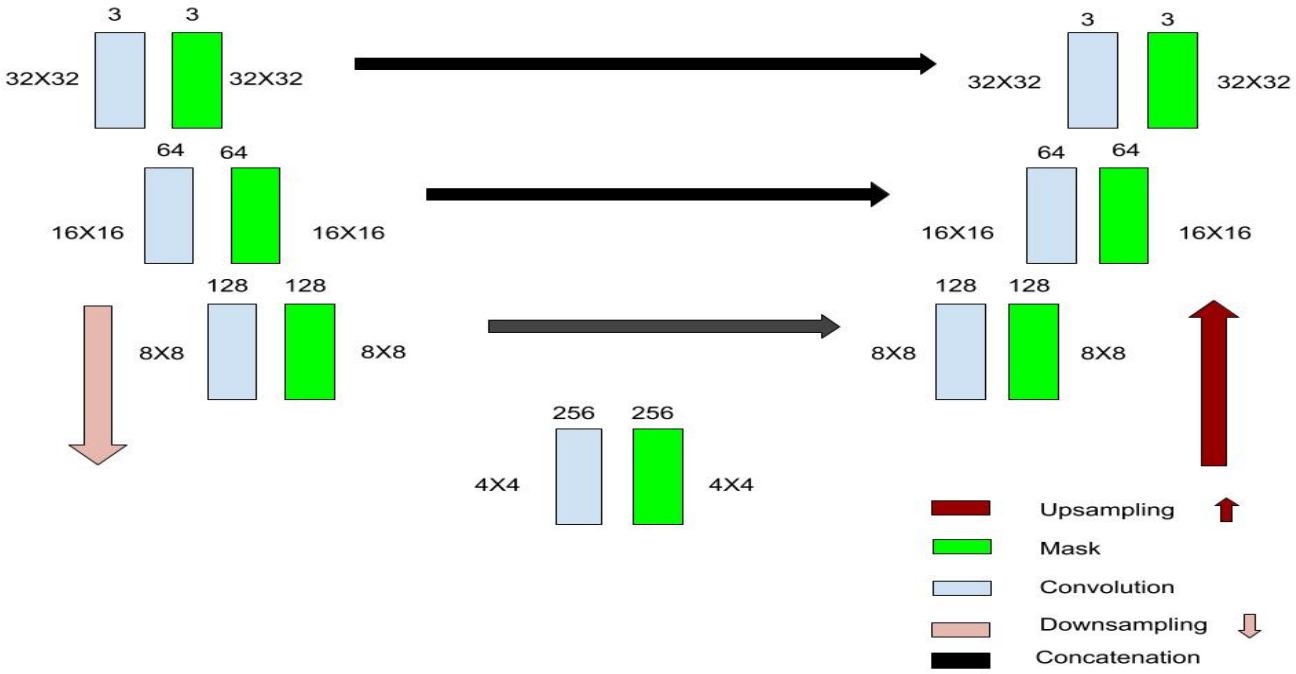

Figure 1. Deep neural network architecture for image size 32x32 pixels

3.1. Map generation

Our deep neural network needs thousands of maps on which to learn an accurate model, however there is a very limited set of available subterranean mine maps. Therefore, we must generate our synthetic dataset in order to train the network. We used the algorithm described below to generate a set of connected tunnel maps on which to train our network.

input : Map Size $S = [32, 32]$, Number of points P
output: Generated Map
$Entrance \longleftarrow (0, floor(y/2))$ Robot's Start
p_{list} = sample P points on the grid
$tree = []$
add $Entrance$ to $tree$
for $i \in p_{list}$ **do**
  find p nearest to i in $tree$
  connect p and i by shortest path
  add p to $tree$
end

Fig.2 shows sample maps in procedurally generated synthetics database. Our database consists of 50,000 maps which we use to train our deep neural network.

3.2. Mask Generation

We use a square mask of size 10x10 pixels for the images of size 32x32 pixels with the goal of keeping the mask size(the patch) to be around 20 percent of the total size. The patch is always generated along the sides(boundaries of maps) and its location is chosen randomly. For each image, we chose one of the sides randomly first(4 sides of a square) and then a starting location for a patch is chosen randomly. Choosing patch positions randomly allows us to reduce variance during training thus reducing error. One of our goals has been to make predictions for a robot that has access to a partially known map. Providing patches alongside the boundary allows us to achieve this goal. The unmasked region may be considered as the map area known to the robot or the area that it has already explored and the masked area is what it would like to have a prediction for.

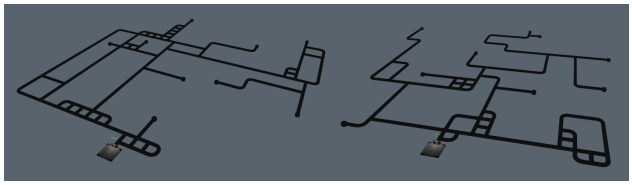

Figure 2. Set of procedurally generated world maps in Gazebo

3.3. Architecture

We use a UNET-like architecture as shown in Fig. 1. The Top left image shows the ground truth image, top right image shows the mask, bottom left shows the masked image, bottom right shows the prediction of the neural network. In this case, network is able to make multiple axis.nput to the

model is 3 images: ground truth, masked image and the mask. The sizes for all the three are 32x32 pixels. We generated only grayscale images for our dataset and converted the 32x32x1 into 32x32x3 by duplicating the 1 channel into 3 channels. This image is then passed through a convolution layer of kernel size 7, stride 2, padding 3 and outchannels being 64. So the output of this layer comes to be 16X16X64. The third dimension is the number of channels here. The output of every layer is as follows:

Type	Kernel	Stride	input	output
down	7	2	32x32x3	16x16x64
down	5	2	16x16x64	8x8x128
down	5	2	8x8x128	4x4x256
down	3	2	4x4x256	2x2x512
up	3	1	4x4x(256+512)	4x4x256
up	3	1	8x8x(256+128)	8x8x128
up	3	1	16x16x(128+64)	16x16x64
up	3	1	32x32x(64+3)	32x32x3

Down-sampling means that the size of the image or output from a CNN is being reduced and up-sampling means the size is being increased (doubled here). The padding for down-sampling layer with kernel size '7' is 3, for '5' and '2' it is 2 and 1 respectively. Padding is 1 for upsampling layers. To perform upsampling we take up the output of the previous layer and interpolate by a scaling factor of 2. Then it is concatenated with the output of a downsampling layer such that the two have the same first 2 dimensional sizes. This only increases the number of channels. So the output 2x2x512 of last convolutional layer is up-sampled to 4x4x512 and then concatenated with 4x4x256 to result in 4x4x(256+512). This data is then passed through a CNN layer to produce 4x4x256. The whole process of upsampling and concatenation is repeated again.

3.4. Loss Function

We adapt our loss function from the image inpainting for irregular holes paper [10].

$$L_{total} = 100 * L_{valid} + 10.5 * L_{holes} + 0.05 * L_{perceptual} + 120(L_{style_{out}} + L_{style_{comp}}) + 2L_{tv} \quad (1)$$

Where L_{valid} is the L1 loss for non hole regions, L_{holes} is L1 loss for the holes regions. $L_{perceptual}$ is the perceptual loss defined in [6]. L_{style} is the style loss similar to [6]. L_{tv} is the smoothing penalty total variance loss [7].

We manually tune the weight parameters for the above loss equation. We keep L_{valid} weight as high as 100 because we want to replicate the non-holes region the same as the input image. For weight of L_{holes}, we compute $w = totalPixel/whitePixel$ over our data-set. We take other loss weights from [10].

4. Experiments

Our experiments involved using a variety of training sets(different image sizes), hyperparameters and optimization algorithms, different number of CNN layers. The actual implementation of UNET in [1] starts with the image size of 512 x 512 and reduces the size after downsampling to 2x2x1024. We tried the architecture shown in the figure with 256x256 images and faced memory overflow problem for CUDA. It is important to mention that we could not use the pelican server to a large extent due to limited disk storage capacity which exceeds only after installing conda and pytorch. Our resources include a desktop in Graf Hall with 8 GB GTX 1080 and a laptop with 6 GB GTX 1060. The data generation part would take approximately 30 minutes to produce 5 GB of data for 10K images. The Cuda memory runs out after a few iteratiThe Top left image shows the ground truth image, top right image shows the mask, bottom left shows the masked image, bottom right shows the prediction of the neural network. In this case, network is able to make multiple axis.ons. We, later on, switched to a small 64x64x3 sized images which would produce results but they were not up to the mark. A reason for this may be that the width of tunnels in our synthetic map is only 1 pixel and 64 image size may be too large for the network to learn the tunnel structures.

Later, we reduced the map size to 32x32x3 and increased the dataset to 40K images. We trained the network for 512 epochs for 9 hours approximately to produce results. This was done a few times while searching for the learning rate for ADAM optimization. The final batch size used was 64 but experiments were also done with 32 and 16 for 32x32x3 image size. The loss curves for different losses used are shown in Fig. 3 where the red line is training loss and blue represents validation loss. The curves show normal trends of train and test losses. The training error is lesser than test for most losses while it overlaps with the test loss for L_{valid}.

5. Results

Our architecture is able to predict intuitive connections in the unknown regions of a map. Fig. 4 shows that the network can predict L shaped tunnel connections and the width of predicted tunnels are same as that of known parts of tunnels. Fig. 5 shows the tunnels can be extended along the axis of the tunnel, similarly Fig. 6 shows the tunnels can be extended in a perpendicular direction. Fig. 7 shows the network can predict multiple connections.

6. Conclusion and Future Work

We observe through experiments, that our model does learn tunnel structures in the map when it is trained on our dataset. It can redraw T-points and loops when trained on $40,000$ images for $400,000$ iterations with properly

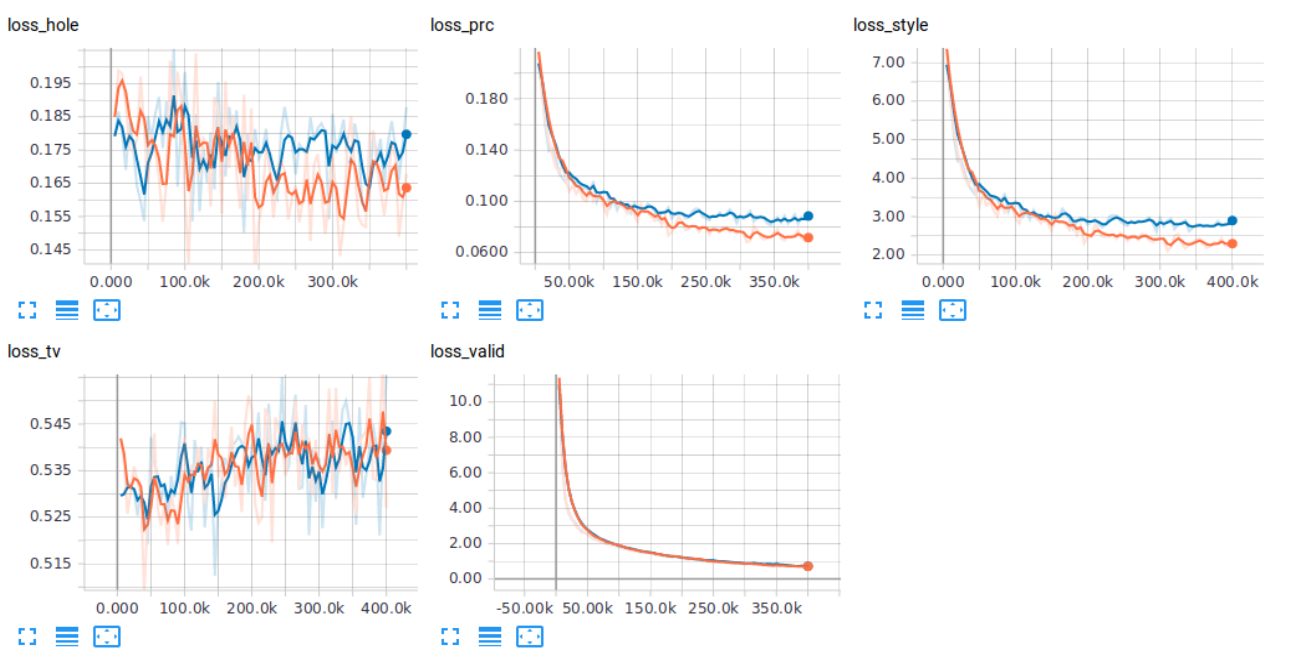

Figure 3. Training and testing loss for all 5 components are shown. The red curve represents training loss and the blue curve represents validation loss.

tuned loss parameters. This predicted map can be used by decision-making algorithms to maximize information gain over traveling cost. We tried to obtain results for synthetic maps using an online available model trained on imagenet and the results were observed to be not good. This is perhaps there is a difference in distribution between our synthetic map dataset and the imagenet dataset. We also observe that ratio between the tunnel width and the size of the image can be a very significant factor for our model to learn.

The performance of our model is owed to the robustness of U-NET architecture and efficacy of partial convolution. It was interesting to see that the model would just not fill holes but can also learn to draw T-points. Results are promising, and in future, we would like to work with more realistic maps; may be using different masks for the same image. We would like to see how would W-GAN perform on our problem.

References

[1] [1505.04597] u-net: Convolutional networks for biomedical image segmentation. https://arxiv.org/abs/1505.04597. (Accessed on 03/23/2019).

[2] DARPA subterranean challenge. https://www.subtchallenge.com/. Accessed:2019-02-22.

[3] generalized_pm.pdf. https://gfx.cs.princeton.edu/pubs/Barnes_2010_TGP/generalized_pm.pdf. (Accessed on 03/23/2019).

[4] Interpolation of missing data in image sequences - ieee journals & magazine. https://ieeexplore.ieee.org/document/469932. (Accessed on 03/23/2019).

[5] S. Bai, F. Chen, and B. Englot. Toward autonomous mapping and exploration for mobile robots through deep supervised learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2379–2384, Sep. 2017.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[7] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[9] Y. LeCun and Y. Bengio. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA, 1998.

[10] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro. Image inpainting for irregular holes using partial convolutions. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[11] A. J. Smith and G. A. Hollinger. Distributed inference-based multi-robot exploration. *Autonomous Robots*, 42(8):1651–1668, Dec 2018.

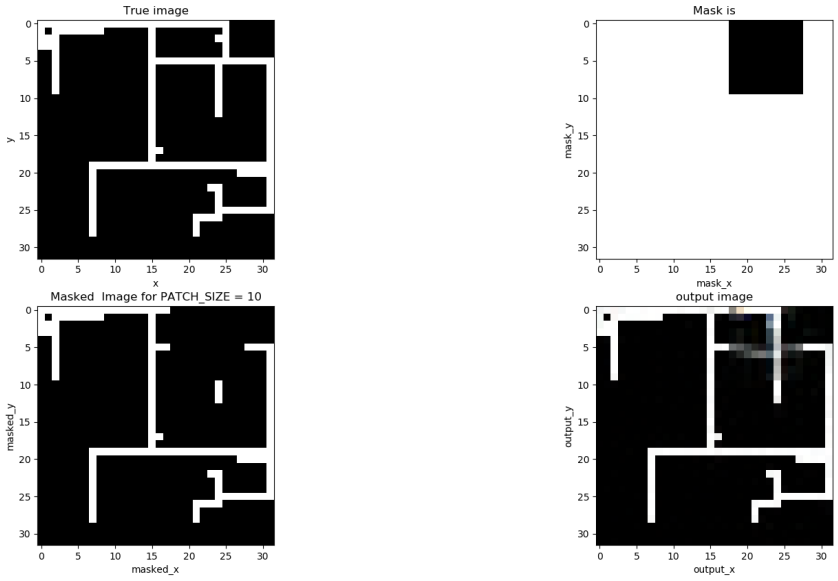

Figure 4. The Top left image shows the ground truth image, top right image shows the mask, bottom left shows the masked image, bottom right shows the prediction of the neural network. In this case, network is able to redraw a L shaped tunnel.

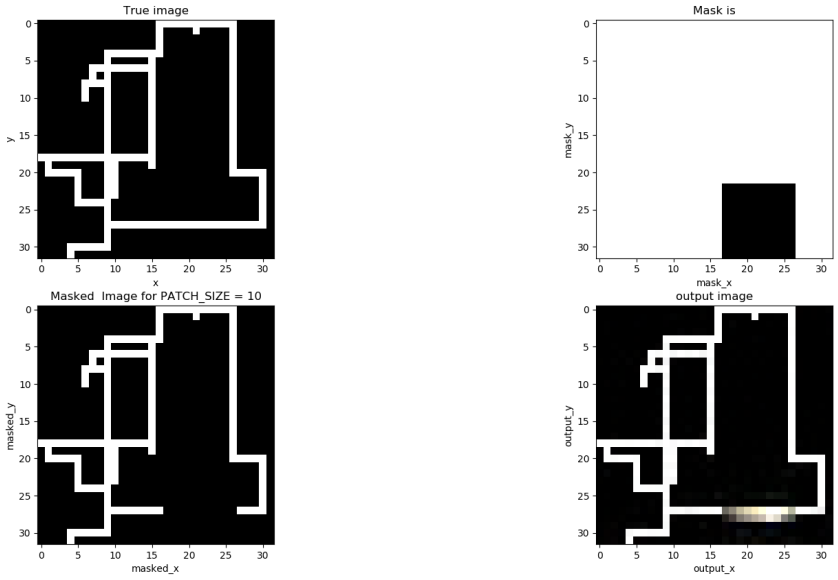

Figure 5. The Top left image shows the ground truth image, top right image shows the mask, bottom left shows the masked image, bottom right shows the prediction of the neural network. In this case, the network is able to extend the tunnel along the tunnel's axes.

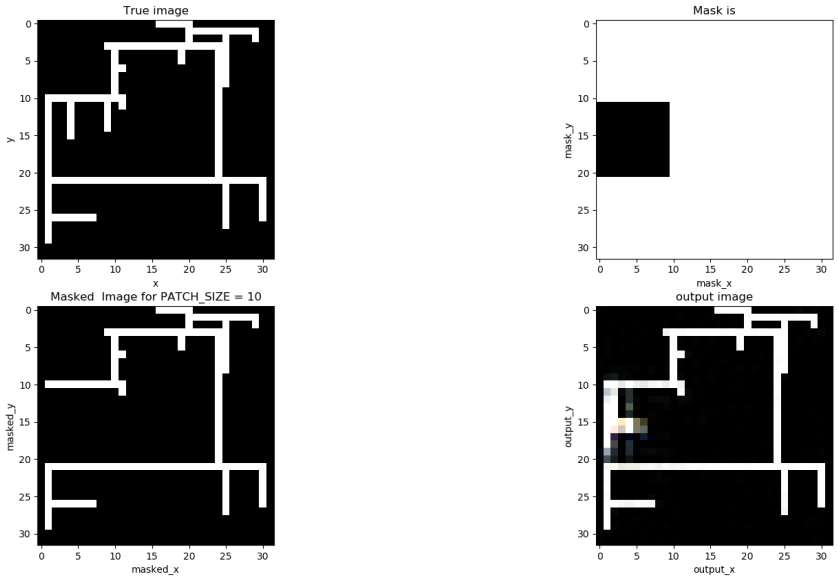

Figure 6. The Top left image shows the ground truth image, top right image shows the mask, bottom left shows the masked image, bottom right shows the prediction of the neural network. In this case, the network is able to extend the tunnel perpendicular to tunnel's axes.

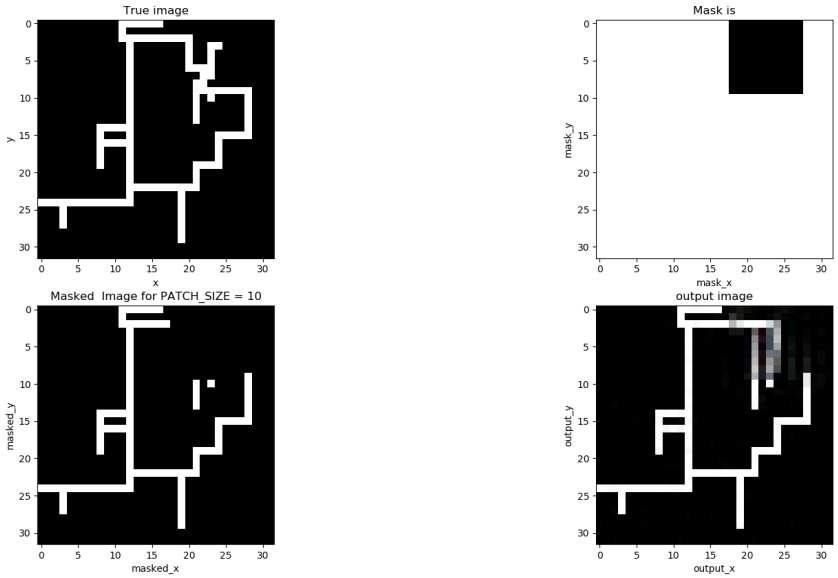

Figure 7. The Top left image shows the ground truth image, top right image shows the mask, bottom left shows the masked image, bottom right shows the prediction of the neural network. In this case, the network is able to make multiple axes.