CSCI 1430 Final Project Report: CNN Depth-aware

Team name: Baokun Huang, Sijie Ding.
Brown University

Abstract

In the computer vision area, identifying the location of the object in the spatial space is a crucial task. One method is using a fundamental matrix from two starting points. It is powerful but also needs some restricted conditions. We want to start an approach detecting the spatial location of all the objects in a single view. It is similar to how our human eyes work, from one view to know which one is closer or which one is farther. In the practice, we will use the CNN model to detect the depth from a given image and output the depth of each pixel. For the experiment, we want to try different structures to see which one is better. To enable the output to be an image size-like matrix, we will add a few more layers at the back of CNN. The result shows that our approach works well on the given data set and this enables us to think of more achievable ideas. In practice, if the algorithm works well, it can replace the RGB-D camera with the RGB camera and save a bunch of money.

1. Introduction

As computer vision becomes popular, many new algorithms have been developed to extract different information from the image. Once we heard of computer vision, we think of the human eyes. Since as humans, we can see the object, identify the class, or even identify the depth roughly. At least, humans can tell about the object's relative space, which ones are closer than the other ones. But the photos or pictures are 2D images, they cannot directly provide depth information by themselves.

We want to introduce a method to tell about the depth of each object in the image from one single picture. The result we want to show is that our network can give the relative depth of each object in the image. In practice, to output the relative depth of each object, we need to obtain the depth of each pixel in the image, which is different from detecting the class. In this case, we need to find a way to predict the result in an image size matrix. We did some research and with the help of TA, we take the approach to add a few more layers at the end of CNN to extend the last layer into the original image size, so that each cell in the output will represent the predicted depth.

In our original approach, we want to use a simple dense layer to transform the latest output into the image size matrix, but we found that the result is too poor, which even cannot provide rough outlines. The reason may be that too much information is contained in the matrix and the size of the matrix is too thicck, a simple dense layer cannot extract all this information to match the depth label. And the layers we added, in the end, enable the training variables to extend the matrix smoothly. Since we use several convolution and max-pooling layers to shrink down the matrix, we should also use several convolution layers and upsampling layers to extend up.

2. Related Work

Based on our topics, we searched on some data set which will provide a labeled depth. We found a couple of data sets but only choose the NYU Depth V2 since it gives the most detailed data set and is easy to load.

The data set we used is the NYU Depth V2 Nathan et al. [1]. In the data set, they also experimented using segmentation and 3D to detect the object's relative space. But their method is too complex and their goal is to identify the depth in a complex environment. In our case, we only want to detect the depth in a faster and simpler way. The difference is that they will result in a more clear edge for each object, but we save more computations.

The method we use to modify our network is from Iro et al. [3]. In the paper, they introduce a method to do up-convolution and extend the refined matrix to an image size output. After reviewing the method, we found that the front part of the construction can be optimized, since they only tried with one type of CNN, we will experiment with the various structure to find the best one for the result.

3. Method

Our workflow is loading the data, then prepossess the data, and then feed the data into our different models. After training is finished, we will visualize the result to see which

Layers	Output Size
Input	(224,224,3)
Convolution	(224,224,64)
Max pooling	(112,112,64)
Convolution	(112,112,128)
Max pooling	(56,56,128)
Convolution	(56,56,256)
Max pooling	(28,28,256)
Convolution	(28,28,512)
Max pooling	(14,14,512)
Dropout	(14,14,512)
Activation	(14,14,512)
Batch Normalization	(14,14,512)
Convolution	(14,14,256)
Up sampling	(28,28,256)
Convolution	(28,28,128)
Up sampling	(56,56,128)
Convolution	(56,56,64)
Up sampling	(112,112,64)
Convolution	(112,112,64)
Up sampling	(224,224,64)
Convolution	(224,224,1)

Table 1. Naive CNN structure

one is better on the outlines and the depth.

To build up the network, we starting from the homework sample network. But the output in this network is the highest possibility of the label, which doesn't match with our target. Instead of using the softmax, we try to replace it with a single dense layer. The result is not good. After doing some researches, we found the ideas from the paper from Iro et al. [1]. They implemented the structure with additional up-convolution layers to extend the refined matrix to be the input matrix size. We add a corresponding number of such layers in our naive network to test if it is doable. And then we tried the new head with VGG and our self-designed network as well.

Naive CNN In the first network (Table 1), our task is to test if the new layer will work. Before the dropout, we use the normal convolution and max-pooling to shrink the size of each slice. And also the number of the upsampling layer should be the same as the max-pooling layer to extend the matrix to be the same as input size. In the last layer, we apply a single filter to combine all the 64 layers into one as our result for each cell.

VGG In our second approach (Table 2), we append our new layer after the VGG to see how the effect is the VGG. And apply fine turning to save the training time. To keep the size, we also need to provide a corresponding number of up sampling layers.

Layers	Output Size
VGG	(7,7,1024)
Dropout	(7,7,1024)
Activation	(7,7,1024)
Batch Normalization	(7,7,1024)
Convolution	(7,7,512)
Up sampling	(14,14,512)
Convolution	(14,14,256)
Up sampling	(28,28,256)
Convolution	(28,28,128)
Up sampling	(56,56,128)
Convolution	(56,56,64)
Up sampling	(112,112,64)
Convolution	(112,112,64)
Up sampling	(224,224,64)
Convolution	(224,224,1)

Table 2. VGG + new layers structure

Our Model In our designed model (Table 3), we modify from the first naive CNN by adding a block and a batch normalization layer at the end of each block. And change the number of up convolution layers to match the max pooling.

4. Results

4.1. Technical Discussion

The original size of the data sets is 480*640*3 (length, width, channels). We resized all input images to 224*224*3 (length, width, channels) and output at the same size. This is aimed to save the space of memory and improve the training speed.

We used Mean squared error loss for all 3 models:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

This is a classical loss function, which is simple but very efficient.

We trained the mode by using 10% of the data set as training data set and ran for 50 epochs. The losses of each model are showed in table 4.

Form Figure 1 we can see the difference between each networks. The first image is the original RGB image. The second one is the result of Naive CNN. It successfully detected depth information, but the contour is blurred. The third one is the result of our model. It is the best one on accuracy. The fourth one is the result of VGG. It has a very clear contour, but the shape is deformed compared with the ground truth. The last one is the ground truth depth image.

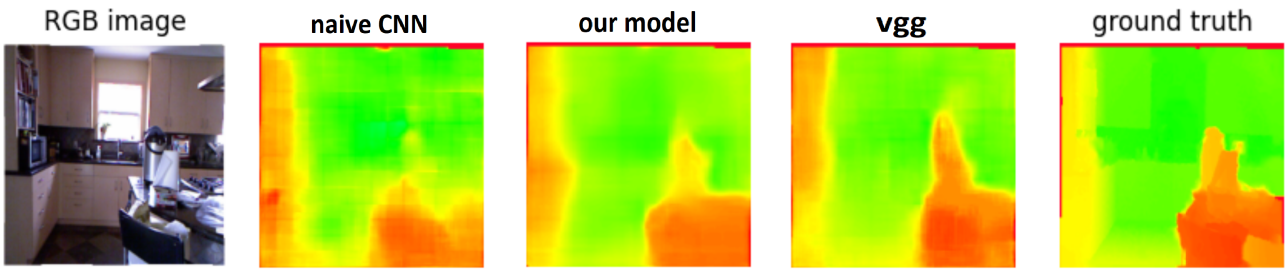

Figure 1. Results from different networks. *1:* Raw RGB image. *2:* result of Naive CNN. *3:* result of Our designed model. *4:* result of VGG. *1:* Ground Truth.

Layers	Output Size
Input	(224,224,3)
Convolution	(224,224,64)
Max pooling	(112,112,64)
Batch Normalization	(112,112,64)
Convolution	(112,112,128)
Max pooling	(56,56,128)
Batch Normalization	(56,56,128)
Convolution	(56,56,256)
Max pooling	(28,28,256)
Batch Normalization	(28,28,256)
Convolution	(28,28,512)
Max pooling	(14,14,512)
Batch Normalization	(14,14,512)
Convolution	(14,14,1024)
Max pooling	(7,7,1024)
Dropout	(7,7,1024)
Activation	(7,7,1024)
Batch Normalization	(7,7,1024)
Convolution	(7,7,512)
Up sampling	(14,14,512)
Convolution	(14,14,256)
Up sampling	(28,28,256)
Convolution	(28,28,128)
Up sampling	(56,56,128)
Convolution	(56,56,64)
Up sampling	(112,112,64)
Convolution	(112,112,64)
Up sampling	(224,224,64)
Convolution	(224,224,1)

Table 3. Our designed model structure

4.2. Societal Discussion

1. RGB-D cameras normally have a higher price than RGB cameras. We hope this project can find out a way to get depth information from RGB cameras so that in some situations RGB-D cameras can be replaced by RGB cameras.

Method	MSE Loss
Naive CNN	0.6916
VGG	0.8290
Ours	0.6876

Table 4. Results.

The 'NYU Depth v2' data set was collected by Kinect, which is an RGB-D camera developed by Microsoft. Kinect was used to work with the video game consoles to detect the actions of players. This is a consumer product, but not a professional product, which means the data set and the model we trained may not be accurate enough for any professional uses [2].

The RGB-D cameras are widely used on auto-drive vehicles, and for many countries in the world, they have got different standards to test the accuracy of the results. The model we trained for using RBG cameras to replace the RGB-D cameras may not be able to used on these auto-drive vehicle, because they are not accurate enough to pass the tests.

2. Manufacturers of robotics, developers of applications, and customers are the stakeholders for this project. Manufacturers of robotics with RGB-D cameras will find a way to replace them with RGB cameras at lower costs. Developers of applications will find a way to get depth information from RGB images. Customers will obtain a similar experience by using RGB cameras instead of RGB-D cameras.

   Once the cost of cameras can be reduced, more manufacturers will be able to take part in the selling market of robotics. The selling prices will drop as well, therefore more customers can have chances to enjoy the products. Furthermore, some applications that need special equipment now can be used on personal smartphones, since nowadays most smartphones have an RGB camera.

   One bad actor might misuse this technology for peeping into the privacy of others. For example, someone may

take a photo of their new houses by using a normal smartphone camera. With this technology, a bad actor may use it to get the information of the sizes of their rooms, the position of their houses, and then sell this private information to black markets. In this example, the owner of the house would be the victim.

3. The goal of this project is to replace RGB-D cameras by RGB cameras. However, when testing the model, we need to make sure the algorithm would not collect personal privacy. For example, when we test the algorithm in someone's house, the algorithm will be able to collect the depth information through the RGB cameras. This depth information may leak the geographical information of the house. Therefore, when we use this algorithm, we should remove all information that may cause a leak of privacy.

4. With this project, a normal camera will be able to do the same thing as an RGB-D camera. People will be able to get the depth information by just using a smartphone camera. This could affect community's civil rights of protecting their privacy. The project may provide convenience to criminals who are trying to steal community's privacy. This criminal use should be prohibited.

5. The data set was collected by Kinect, which was a product of Microsoft. It is design for consumer use, but not the professional uses. Some bias caused by the accuracy may happen and is not able to be mitigated by preprocess. This data set focuses on room images, therefore, the model may have a bias on room scenes, and may not behave as well as on other scenes.

5. Conclusion

Our experiment shows that appending the up-convolution layers to the existing network will let the algorithm be able to detect the depth of the RGB room image. And among our tests, we find that our designed model has the least loss grade in the testing, which means the batch normalization is such powerful in the CNN.

In partice, our algorithm makes an RGB camera be able to achieve depth information and replace RGB-D cameras in some situations. This would provide convenience to users who are unable to offer an RGB-D camera but instead use an RGB camera to replace it.

References

[1] I. Laina. Deeper depth prediction with fully convolutional residual networks. *IEEE International Conference on 3D Vision*, 2016. 1, 2

[2] K. Litomisky. Consumer rgb-d cameras and their applications. 2012. 3

[3] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. *ECCV*, 2012. 1

Appendix

Team contributions

Baokun Huang Algorithm code of models and training. Introduction, related work and methods part of the report.

Sijie Ding Algorithm code of processing and testing. Results and conclusion parts of the report.