Depth Estimation with Fully Convolutional Neural Network

Zhaoyanh HUANG, 21721097, Computer Science
drinkingcoder@zju.edu.cn, 18868107260
Hailin YU, 21721039, Computer Science
hailinyu@zju.edu.cn, 13819495414

Abstract

Use deep convolutional neural network to complete image classification task is not just classify the object, but also extract some significant semantic features. Estimate dense depth information from a monocular image is a crucial task for 3D geometry recovery with camera. In fact, we can maintain depth information from the extracted semantic information so we addressed a method that utilize the neural network which has been trained on image classification task to maintain the high level semantic features and then estimate the depth according to fine-tune a transpose convolutional neural network after it.

Key Words

Depth Estimation, Fully Convolutional Neural Network, Deep Learning

1 Introduction

Deep learning acts as a significant role in image processing recent years. Estimating depth from a monocular image is a significant task for 3D computer vision for scene geometry recovery. Images are generated by recording light from surrounding objects, which is called as affine projection. Lots of information is lost after this process, so depth can offer computer more coherent information in images. This technology can benefit some researchers that major in other areas of computer vision a lot, such as RGB-D 3D reconstruction with only RGB images, structure from motion, etc.

In this paper, we address a method that try to recover depth from high level semantic features with an end-to-end modified VGG19 model. Our method relies on a RGB and depth dataset, after training step we can generate a pixel-wise depth label for a specified RGB image. Our extended approach uses a pre-trained VGG19 neural network to extract hierarchical features from low-level to high-level, and then use a neural network which contains three transpose convolution layers and three normal convolution layers replacing the fully connected layers to generate depth map. A critical alternation should be noticed is that we only utilize the semantic information that extracted be the last convolution layer in VGG19 net rather than multi-scale approach proposed by Eigen[1]. This choice based on the consideration that we want to concentrate the depth recovery for objects in image instead of the whole image, which is more important in object tracking task, salience detection task, etc, but multi-scale architecture will accumulate cost in low-level texture areas which may reduce the performance in object depth estimation.

Estimating depth from images is a classical research area in computer vision, and there is a lot of works based on epipolar geometry. Estimating depth from more that one image based on stereo image and motion[4,5] can maintains high precision for images that contain abundant texture information, but they suffer from the scale problem because of the unknown baseline between cameras, and the unknown intrinsic camera parameters. Use traditional method to estimate depth from a monocular image can still be a hard problem, but it can be a very important area in computer vision. We try to use neural network to learn color – depth relation information on a constant scale dataset, and then we can estimate depth from a specified image with accurate scale.

Structure from motion with monocular camera suffers a lot from the scale variant problem and ill-posed problem which can be reduced slightly by sensor fusion with IMU information[2], but stereo image(which we can generate by estimating depth from monocular image) can still help to solve these problems a lot.

We tried to consist trimmed VGG-19 convolution layers and up-sampling layers to be a new neural network architecture and train it on NYU Depth v2 labeled dataset. Our main contributions are addressed below:

- Design a new neural network architecture that consists a pre-trained, fully connected layer trimmed VGG-19 net and a six layer up-sampling neural network.
- Design a new kind of up-sampling neural network architecture which use transpose convolution layer and convolution layer alternatively
- Replace relu activate function in VGG-19 with leaky-relu, which can increase robustness in fine-tuning process.

We will introduce more details of our works in following sections. In section two, we will simply introduce the related works about the topic that estimating depth from a single image. In section 3, the architecture of out network will be addressed, and training process， parameter configurations and other techniques that are used in our works will be detailed, too. The experiments and results are presented in section 4. Section 5 concludes the paper and describes the future work to improve the method proposed by this paper.

2 Related Work

Estimating depth from RGB image is an active research topic in computer vision because it's a critical technology for SLAM(simultaneously localization and mapping) and 3D reconstruction. Traditional methods relies on feature extraction and feature matching, so we can estimate depth of a detected point from epipolar geometry, which we call it triangulation. Some people use markcov random field to infer depth from local and global features that extracted from image by Saxena et.al[6].

With development of computer hardware and remarkable success of deep learning technology in image processing, people try to solve depth estimation problem by an end-to-end convolutional neural network, and lots works have been proposed, which achieves wonderful performance in deed recent years. Eigen et,al[1] firstly applied deep learning technology to depth estimation with an amazing success. They implement a two-stage neural network to complete a dense depth map regression task, where the first stage is feature extraction based on a AlexNet which has been pre-trained in ImageNet and the second stage is a up-sampling neural network which composed multi-layer output from the previous AlexNet to produce depth map. Their approach is improved a lot to predict normal map and semantic label with similar two-stage architecture, but they use deeper neural network VGG-16 to in the feature extraction stage.[1] Laina et,al[3] replaced VGG-16 with residual neural network to achieve better feature extraction and design a up-sampling layer to replace fully connected layers, which makes the neural network don't be limited by a specified image size anymore, and due to the fully connected layers are trimmed, size of the parameters in the neural network decrease a lot, too.

Our neural network architecture is composed by two stage, too. Borrowing the idea that trim the fully connected layers from Laina et,al., we use a fully convolutional layers trimmed VGG-19 neural network to extract hierarchical features on the first stage. On the second stage, we design a transpose convolution layer and convolution layer in turn to up-sample extracted features in order to regress a pixel-wise depth map. As VGG-19 doesn't use batch normalization technology, we replaced relu with leaky relu as activate function to maintain a good data distribution. Of course we used batch normalization in the neural network on the second stage. Data augmentation is performed on the dataset to improve the ability of generalization of out model.

Our networks architecture is also a fully convolutional networks without fully connected layers, which is consist of VGG-19 convolutional layers and up-sampling layer. In addition, we add batch normalization into up-sampling layers to accelerate the networks convergence and boost the ability of generalization., which will be detailed in the next section.

3 Approach

We will address details of our works in this section, which contains the global model architecture, feature extraction stage, up-sampling stage, the composition of loss function and data augmentation methods.

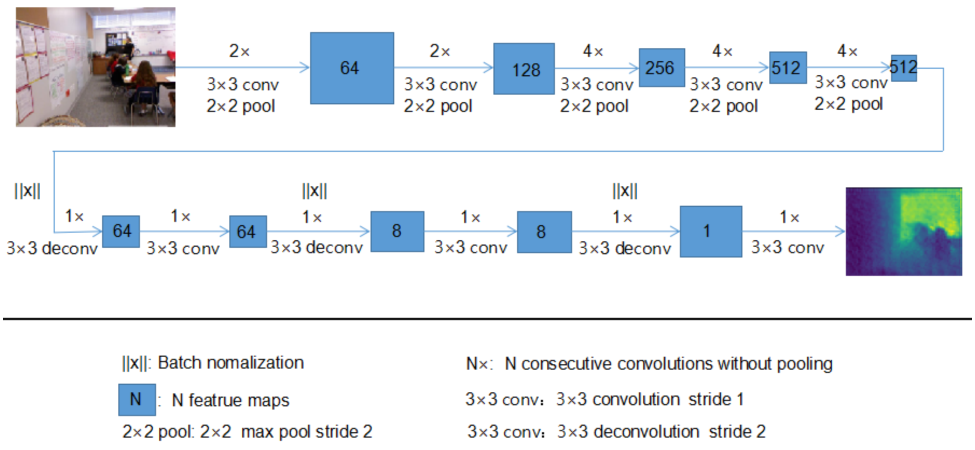

Figure 1. Network architecture.

3.1 Model Architecture

By referring Laina et,al, our network architecture consists two stage, which is shown in Fig.1. Stage 1 is drawn in the first row, which is a trimmed VGG-19 model. It extract features from original RGB image using convolution layers and pooling layers. The second stage of our architecture guides the network into learning its up-sampling ability through a sequence of deconvolutional and convolutional layers.

According to the statement presented in [3], we can feed image of arbitrary size into our neural network without limitation on architecture but with limitation on hardware. This paper presents an implementation with input size of 160x120 and output size of 80x60.

3.1.1 Feature extraction Stage

Deep convolutional neural networks extract features hierarchically by overlay convolution layer and pooling layer turn by turn. To capture objects' information in image, we use a VGG-19 network whose fully connection layers are trimmed and utilize the activation map from the last layer. Input of VGG-19 is an image whose size is 160x120.

As VGG-19 neural network doesn't take batch normalization layer to maintain a good distribution shape of data in each layer, we'll get so many zeroes in the feature map that output from the last layer. To avoid this, we replaced relu activate function that original used in VGG-19 with leaky relu in afterwards fine-tuning step.

3.1.1 Up-sampling Stage

Original VGG-19 neural network uses fully connected layers to classify images after the convolution layers. As we know fully connected layers consumes lots of parameters and this method only owns the ability to classify objects. To complete regression task, we trimmed the fully connected layers and then append an up-sampling neural network with output size of 80x60 to it. The experiment shows that the regressed depth map gets many style of blocks if we only use transpose convolution layers with specified kernel size and stride, so we add a convolution layer after each transpose convolution layer to solve this problem just like the architecture presented in Figure 1. In the forth section, we will show the performance improvement by leading convolution layer in.

3.2 Loss Function

We choose L2 loss function as training loss as bellow:

$$\mathrm{L}(y^*, y) = \frac{1}{n}\sum_i d_i$$

where $d_i = y_i^* - y_i$.There are many other loss function than can be used, but in this paper we just test this loss function.

3.3 Data Augmentation

We augment the training data with random online transformations which is introduced by Eigen et,al[1]. As translation will affect inherent geometry property that the image owns, we use a constant scale factor to remain the scale consistency.

There are five different data augment methods:

- Rotation: Input and target images are random rotated by [-5,5] degrees.
- Flips: Input and target are horizontally flipped with 0,5 probability.
- Color: Input values are multiplied globally by a random RGB value belong to [0.8, 1.2].

According to limitation of hardware and available time, we don’t use random scale to augment data.

Of course, we pre-scaled input images into size of 228x228.

4 Experiments and Results

We trained our model on NYU Depth v2[15] labeled dataset and use the benchmark proposed by Eigen et,al[1] to show performance. In this section, we will briefly introduce the dataset we used and show the results that generated by our model. The quantitative results will be presented.

4.1 NYU Depth

The NYU Depth dataset[15] is composed of 464 indoor scenes, taken as video sequences using a Microsoft Kinect camera. We use the official train/test split, using 249 scenes for training and 215 for testing, and construct out training set using the labeled data for these scenes. RGB inputs are down-sampled by quarter, from 640*480 to 160*120.The labeled data includes 1449 images, 654 of which are left as test data and others are used for training. Every RGB image in the labeled data correspond to a depth map, which have been filled in depth values.

As configuration above, the training set has 800 unique images. We train our network using SGD with batches of size of 16. We initialize the down-sampling part by the pre-trained weights of VGG-19 and the up-sampling part by normal random value with 0 mean and 1 variance. All layers of the network have the same learning rate:0.001.Training took 8 hours total 1000 epochs using a NVidia GTX970.

4.2 Results

We trained our network on NYU depth v2 labeled data with commonly used 795 images. For accurately evaluating our methods, we use official test data 654 images to compute the errors with several error metrics, such as RMSE(liner), RMSE(log) and squared relative difference and so on, which has clearly definitions in [1].

	A1(no conv)	A2(no aug)	A3(Final)	Eigen	Ladicky & al
Threshold $\delta < 1.25$	0.569	0.609	0.629	0.611	0.542
Threshold $\delta < 1.25^2$	0.849	0.862	0.880	0.887	0.829
Threshold $\delta < 1.25^2$	0.944	0.951	0.961	0.971	0.940
Abs relative difference	0.245	0.248	0.226	0.215	-
Sqr relative difference	0.269	0.260	0.219	0.212	-
RMSE(liner)	0.978	0.918	0.850	0.907	-
RMSE(log)	1.036	0.315	0.299	0.285	-
RMSE(log,scale invarient)	0.997	0.240	0.249	0.219	-

Table 1. Comparison on the NYU Depth dataset. A1 represents the network that use 3 transpose convolution layer to regress the depth map with data augmentation. A2 represents the network that also use 3 transpose convolution layer, but we insert a convolution layer after each transpose convolution layers. We don't use data augmentation method in this network. A3, which is the final model we maintained, represents the model that added data augmentation processing upon A2.

As is shown in Table 1, the result produced by the network trained by the data augmentation way is better than no data augmentation. So data augmentation indeed can improve the ability of generalization of networks. Comparing A1 and A3, we can find that the network, which up-sampling layers is equipped with convolution, is better than that no convolution.

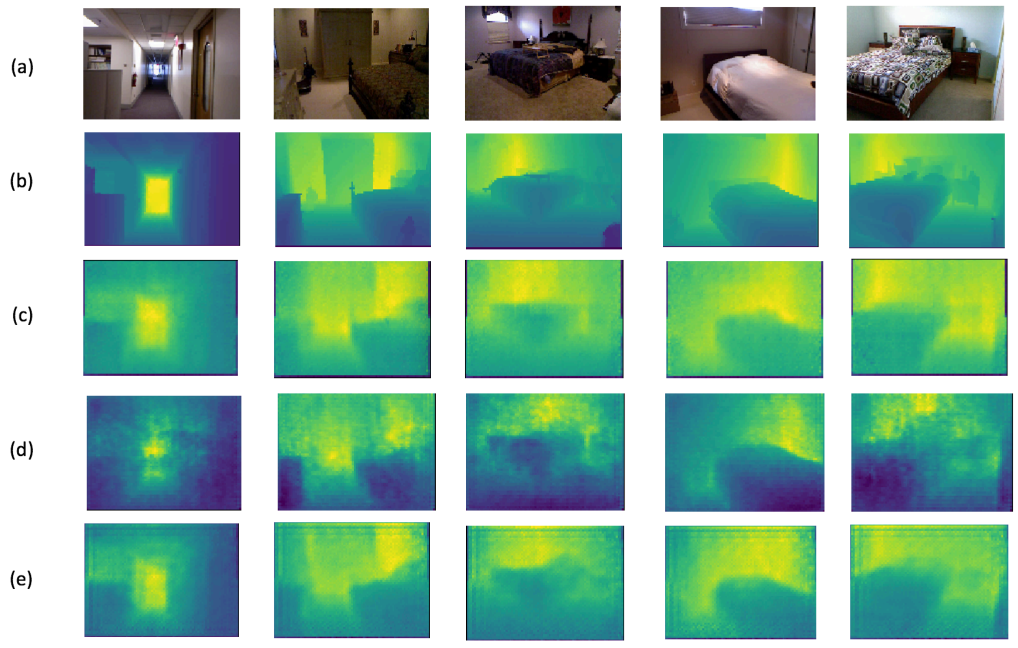

Figure 2. (a) input image; (b) ground truth (c)A1 network(no_conv); (d)A2 network(no_aug); (e)A3 network;

As you can see, although data augmentation can improve network's generalization, but it will result black margin surrounding the predict depth map. This effect results from the random rotation in data augmentation when we train the network, and it can be avoided.

5 Conclusion

Deep convolutional neural network is a suitable solution for monocular image depth estimation problem, because it can capture much prior information in previous trained dataset. A classical computer vision task – image recognition doesn't just only solve this specific problem, but also selected representative features hierarchically. Some famous deep learning architecture like AlexNet[10], VGG[8] and ResNet[13] can be treated like a feature extractor, so we can solve so many other problem like depth estimation from fine-tuning a pre-trained network. Our network uses VGG-19 convolutional layers as feature extractor who has been train on image net dataset, and fine-tune it with another up-sampling architecture by training it on NYU Depth v2 labeled data. Finally, we test it through a benchmark provided by Eigen[1]. As shown above, the estimated depth maps just have a blurry outline of objects and is lack of details, and we want to refine it through adjusting network architecture in the future.

In the future, we plan to go step further to adjust our network architecture. The network is composed of convolution, pooling and deconvolution, and convolution and pooling is used as down sampling to reduce the image resolutions, and in this process images lose a lot of details so that in the final stage depth maps is just an outline. In order to retain object details, we plan to additionally link every down-sampling layer output to corresponding up-sampling layer, which has the same input feature map size as down-sampling output. We believe that it can get a better result.

References

[1] Eigen, David, et al. "Depth Map Prediction from a Single Image Using a Multi-Scale Deep Network." Neural Information Processing Systems, 2014, pp. 2366–2374.

[2] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. IMU preintegration on manifold for ecient visualinertial maximum-a-posteriori estimation. In Robotics: Science and Systems (RSS), 2015.

[3] Laina, Iro, et al. "Deeper Depth Prediction with Fully Convolutional Residual Networks." 2016 Fourth International Conference on 3D Vision (3DV), 2016, pp. 239–248.

[4] R. Memisevic and C. Conrad. Stereopsis via deep learning. In NIPS Workshop on Deep Learning, volume 1, 2011.

[5] F. H. Sinz, J. Q. Candela, G. H. Bakır, C. E. Rasmussen, and M. O. Franz. Learning depth from stereo. In Pattern Recognition, pages 245–252. Springer, 2004

[6] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In Advances in Neural Information Processing Systems, pages 1161–1168, 2005.

[7] Li, Jun, et al. "A Two-Streamed Network for Estimating Fine-Scaled Depth Maps from Single RGB Images." 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3392–3400.

[8] Simonyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." International Conference on Learning Representations, 2015.

[9] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[10] Krizhevsky, Alex, et al. "ImageNet Classification with Deep Convolutional Neural Networks." Advances in Neural Information Processing Systems 25, 2012, pp. 1097–1105.

[11] Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." International Conference on Machine Learning, 2015, pp. 448–456.

[12] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. IJCV, 47:7–42, 2002.

[13] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[14] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1253–1260. IEEE, 2010.

[15] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 33(5):978–994, 2011.

Appendix(Task Arrangement)

Neural Network Architecture Design: Hailin YU
Data Processing and Augementation: Zhaoyang HUANG
Experiments and Conclusion: ZhaoyangHuang, Hailin YU