Depth Estimation from Monocular Images

Gitika Meher[1]
UC San Diego
9500 Gilman Dr, La Jolla, CA 92093
gkarumur@eng.ucsd.edu

Manjot Singh Bilkhu[1]
UC San Diego
9500 Gilman Dr, La Jolla, CA 92093
mbilkhu@eng.ucsd.edu

Anurag Paul[1]
UC San Diego
9500 Gilman Dr, La Jolla, CA 92093
a8paul@eng.ucsd.edu

Nikhil Mohan[1]
UC San Diego
9500 Gilman Dr, La Jolla, CA 92093
nmohan@eng.ucsd.edu

John Messerly[1]
UC San Diego
9500 Gilman Dr, La Jolla, CA 92093
jmesserl@eng.ucsd.edu

Abstract

The problem of depth perception from real world images is widely studied for its implementations in Augmented Reality and Robotics. While estimating depth from more than one image is a trivial computer vision task, estimating depth from a single image is a much more intricate issue. In this problem we propose two novel architectures to tackle this problem. The first architecture involves the use of Generative Adversarial Networks [4] to validate the depth maps patch-wise, generated by a conventional U-net [12] architecture. This approach allows the network to learn much faster and in turn achieve convergence in a few epochs. The second architecture builds on the Multi-Scale approach proposed by Eigen et. al. [2] by introducing a high scale VGG-16 [14] network to act as a semantic segmentor which feeds into subsequent scales. In both cases our methods achieve near state-of-the-art accuracy with lesser training time and GPU requirements.

1. Introduction

Estimating depth is an extremely crucial task in order to understand underlying geometric relations in a scene. There has been significant contributions in this field due to the advancements in ConvNets and Deep Learning in general, however the issue of estimating depth from just a single real world image is a fairly complicated task.

In the absence of any lighting or geometric cues, depth estimation from just a single image is generally a very ill posed problem. There can be multiple 3-D scenes which generate the same 2-D picture and an accurate depth map prediction is near impossible. The human brain however can still judge depth from a single picture and with a certain confidence level can predict the relative depths of the objects in the scene with reasonably good accuracy.

One of the earliest approaches for depth prediction involved over-segmenting the existing image into superpixel regions [10] and predicting the depths of those superpixel regions using Markov Random Fields [13]. These approaches provide a sparse depth estimate and are heavily dependent on the method and number of superpixels the image is segmented into.

Recent years have witnessed the explosion of deep Convolutional Neural Networks, since the breakthrough of Krizhevsky et al. [7], in a variety of Computer Vision Tasks. CNN features have been continually setting the benchmarks in vision tasks such as semantic segmentation, style generation, object detection, size prediction and depth estimation. The works of Liu et al [10], have also successfully merged Deep Convolutional Neural Networks with continuous Conditional Random Fields (CRFs) which when applied in the right setting directly ends up solving the Maximum Likelihood problem as the partition function can be analytically calculated.

Most of the architectures proposed in the past involve very convoluted models and large GPU requirements in order to provide a dense depth estimate. Often times such an accurate dense depth estimate is not required as relative

depths are sufficient for many tasks. Speed is often an issue in these cases as real time applications of depth estimation algorithms requires a quick assessment of depth in order to make split second decisions. One can observe that there is an apparent trade-off between the accuracy of the estimated depth maps and the running time of the model.
We attempt to tackle this problem by suggesting two novel architectures for the purpose of depth estimation. Firstly we we imporove upon the Multi Scale architecture proposed by Eigen et al. by introducing a VGG-16 [14] network to act a semantic feature extractor. This VGG 16 network being pretrained on the ImageNet dataset allows our network to achieve faster convergence time. Such a network is able to learn depth initially based on object segmentation and then refines the depth maps, to remove this strong dependence.
We also propose a second architecture which is inspired from the Fully Convolutional Residual Networks proposed by Laina et al [9], which aims to learn the mapping between a single RGB image and its depth map by a deep CNN trained end to end. We hypothesize that adding a patch discriminator network to the fully convolutional network allows the network to refine its' depth patch wise and subsequently allows visually stimulating depth results in under 5 epochs.
In summary the layout of this paper is as follows

- We expand on some of the relevant and recent literature concerning the question of depth estimation, focusing strongly on the works of Eigen et al [1], and Laina et al [9]. We restrict ourselves to applications of deep convolutional networks as these type of networks have achieved an enormous success when it comes to vision tasks.
- We propose two novel architectures and theoretically justify why both those architectures are able to learn faster than the previous methods.
- We elaborate on the mathematics behind GAN's and also discuss the scale invariant loss function [2] and the crucial role it plays for depth estimation.
- We discuss the results of both our models, display corresponding outputs and also compare our method to existing state of the art approaches.
- A detailed list of references for the curious reader.

2. Literature Review

Machine learning techniques for monocular depth perception can be traced back to 2004 with the seminal work of Saxena et al [13], who predicted depth at absolute and relative scales using Markov Random Fields. They approached the regression problem by dividing the image into patches, and estimating absolute and relative depth of each patch using a Gaussian probabilistic model. While we chose different regression techniques, we adopted the same multi-resolution scale and image patching approach.

Our multi-scale CNN architecture experiments can be attributed directly to the work of Dr. David Eigen at NYU and Robert Fergus at Facebook AI [2] [1]. They were the first to introduce a multi-scale architecture for depth prediction, with a 2-scale architecture that captures global depth and local depth separately. Notably, this work introduced a scale invariant loss to to account for the fact that the depth estimated at different pixel resolutions should be the same. This architecture acted as a springboard for our initial experiments regressing depth directly (without a GAN). Their work using VGG-16 [14] as a feature extractor for semantic segmentation was also a large influence for our experiments with VGG-16 in depth prediction. Eigen and Fergus's work also served as a baseline comparison for the 2019 state of the art in monocular depth prediction.

While no prominent generative architectures for depth prediction exist in the literature today, the work of Isola et al. at the Berkeley AI Research Laboratory influenced our experimentation with GANs [6]. Isola and his peers introduced the general problem of image translation, such as predicting a daytime representation of a night-time photo, or a colored picture from a black and white one. Isola paired a U-net generator with a "Markovian Discriminator" for their architecture, which we adopted and modified for the task of depth prediction. Namely, the Markovian Discriminator restricts the loss function to local NxN areas of the image, in order to regress on small local areas. This models the image in a similar way to Saxena's Markov Random Fields, and as such we saw it as a promising avenue to explore for depth prediction, which benefits from analyzing images at multiple resolution scales.

3. Architecture

3.1. Multi Scale Architecture

Multi-scale approaches, in the context of computer vision, attempt to model the regression problem at multiple resolution scales. Generally this translates to predicting at a "global" scale for an overall estimate, and then predicting at smaller patches for a more "refined" estimate. In the context of deep learning, multiscale architectures concatenate the output of one neural network into the input features of another network.

The CNN architecture we experimented with has three scales. The first scale extracts semantic attributes (edges, textures) from the original RGB image using pre-trained convolutional layers from VGG-16, and then upsamples them to match the dimensions of the next scale. VGG-16 was chosen as a feature extractor over AlexNet [8] or

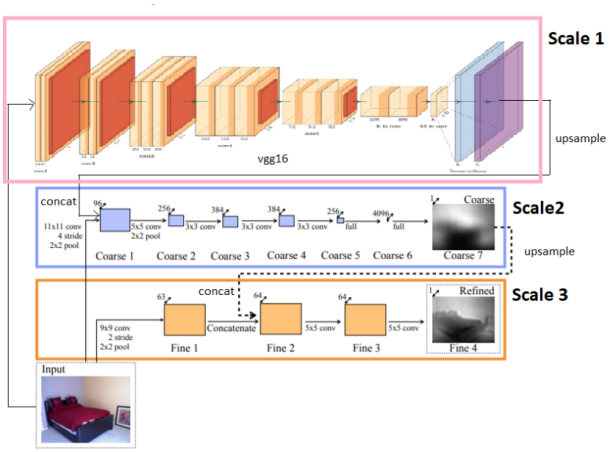

Figure 1. Three Scale Architecture

ResNet-50 [5] based on the 2016 experimentation of Laine (et al.) [9] and successful implementation of Eigen and Fergus [1]. It is noted that while Scale 1 uses pre-trained layers from VGG-16, it was unfrozen for training, in that we modified its parameters during backpropagation with a very small learning rate.

The second scale is a convolutional network with 3 convolutional and 2 fully connected ReLU layers that takes in as input the original RGB image and Scale 1's extracted semantic attributes, and outputs a depth estimate of the entire image. This output is called the "coarse" estimate, and represents a mean prediction of the depth map.

The third scale takes as input the original RGB image, as well as the coarse estimate, and generates a superior estimate. In Eigen and Fergus' 2014 work [2], this network is trained separately from the coarse network, and only looks at patches of the input in 45x45 blocks enforced by zero padded convolutional layers. As part of our initial experimenting, we unfroze the coarse network and trained them together on the same loss function. Our results were roughly the same as training the networks separately.

Performance of the refined network is largely determined by the performance of the coarse network. In order to remove dependencies between the coarse and fine scale loss function, a scale invariant loss function was used, which is mathematically described in the next section.

3.2. Patch GAN

GANs [4] are generative models that learn a mapping from random noise vector. The generator is trained to produce outputs that cannot be distinguished from real images by an adversarially trained discriminator, which is trained to do as well as possible at detecting the generators fakes. There is a great deal of low-level information like colorization, location of prominent edges etc. and it would be desirable to shuttle this information directly across the net-

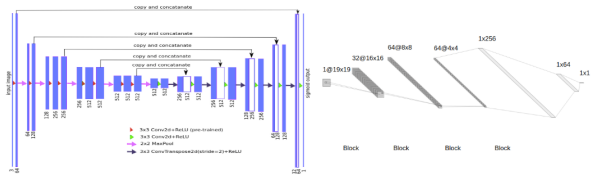

Figure 2. Patch GAN Architecture

work. To give the generator a means to circumvent the bottleneck for information like this, we add skip connections, following the general shape of a U-Net. The U-Net [12] is an encoder-decoder network with skip connections between mirrored layers in the encoder and decoder stacks.
For the discriminator architecture, we employ a patch-wise discriminator which only penalizes structure at the scale of patches. This discriminator tries to classify if each N x N patch in an image is real or fake. The discriminator runs convolutionally across the image, averaging all responses to provide the ultimate output of the discriminator.

4. Model Formulation

4.1. Generative Adversarial Networks

GAN learns a generative model G via an adversarial learning process which involves the joint training of both a generative network G and a discriminative network D simultaneously. The generator G is trained in such a way that it is able to fool the discriminator network D, and the discriminator D is trained to distinguish the actual image from a fake. In our setting we apply this process on to patches generated by the U-Net Convolutional network, which allows to refine the depth maps patch wise. Through empirical analysis we determine that a patch size of 19x19 with full stride performs optimally without significant increase in training time. The discriminator is trained to minimize the following binary cross entropy loss

$$\mathcal{J}^{(D)}(\theta^{(D)}, \theta^{(G)}) = \frac{-1}{2}\,\mathbb{E}_{x\sim p_{data}(x)}[\log D_\theta(x)] +$$

$$\frac{-1}{2}\,\mathbb{E}_{z\sim p_z(z)}[\log(1 - D_\theta(G_\omega(z))]$$

.

Here z denotes the random noise of the image, or in practical terms the sample depth estimate generated and x denotes the real data and ω denotes the parameters G and D.

The Generator G is trained using a linear combination of the scale invariant loss function and the L1 loss

$$\mathcal{L}(y_i, y_i^{'}) = \frac{2}{3}\left[\frac{1}{n}\sum_{i=1}^{n} d_i^2 - \frac{\lambda}{n^2}\left(\sum_{i=1}^{n} d_i^2\right)^2\right] +$$

$$\frac{1}{3}\left[\frac{1}{n}\sum_{i=1}^{n} |y_i - y_i^{'}|\right]$$

where $d_i = \log(y_i) - \log(y_i^{'})$. and

$$y_i = \begin{cases} y_i & y_i > 0 \\ 0.00001 & y_i \leq 0 \end{cases}$$

4.2. Scale Invariant Loss

Statistically as proven by the Multi-Scale Deep Network paper [2] by D.Eigen, MSE loss or L1 loss is simply a measure of how well the mean depth is predicted. Using an oracle to substitute the mean log depth of each prediction with the mean from the corresponding ground truth improves the performance considerably. Motivated by this, we use a scale-invariant error to measure the relationships between points in the scene, irrespective of the absolute global scale. For a predicted depth map y and ground truth y^*,each with n pixels indexed by i, we define the scale-invariant mean squared error (in log space) as:

$$D(y, y^*) = \frac{\sum_{i=1}^{n}(\log y_i - \log y_i^* + \alpha(y, y^*))^2}{2n} \text{ where,}$$
$$\alpha(y, y^*) = \frac{\sum_{i=1}^{n}(\log y_i^* - \log y_i)}{n}$$

is the value of α that minimizes the error for a given (y, y^*). For any prediction y, $\exp(\alpha)$ is the scale that best aligns it to the ground truth. All scalar multiples of y have the same error, hence the scale invariance. Two additional ways to view this metric are provided by the following equivalent forms. Setting $d_i = \log y_i - \log y_i^*$ to be the difference between the prediction and ground truth at pixel i, we have

$$D(y, y^*) = \frac{\sum_i d_i^2}{n} - \frac{(\sum_i d_i)^2}{n^2}$$

This representation of the above equation relates the metric to the original L2 error, but with an additional term that credits mistakes if they are in the same direction and penalizes them if they oppose. Thus, an imperfect prediction will have lower error when its mistakes are consistent with one another.

4.3. Training Loss

The Multiscale architecture uses the scale invariant loss for training and the Patch GAN Architecture uses an empirical combination of L1 loss and the Scale invariant loss for training.

Table 1. Table showing MSE, L1, BerHu and Scale Invariant loss for different learning rates and batch sizes for the Multi Scale architecture (T: Training and V: Validation)

Configuration	0	1	2	3	4	5
LR	5.00E-06	5.00E-06	5.00E-06	2.00E-05	2.00E-05	2.00E-05
BS	8	16	24	8	16	24
epochs	30	13	10	20	20	20
T:SIL	0.332	0.487	0.568	0.346	0.348	0.432
V:SIL	0.673	0.76	0.84	0.604	0.617	0.861
T:L1	0.104	0.118	0.137	0.105	0.097	0.117
V:L1	0.145	0.156	0.165	0.142	0.138	0.164
T:MSE	0.019	0.024	0.031	0.02	0.017	0.023
V:MSE	0.035	0.041	0.044	0.035	0.033	0.045
T:BerHu	0.089	0.103	0.118	0.09	0.085	0.101
V:BerHu	0.123	0.135	0.141	0.12	0.121	0.145

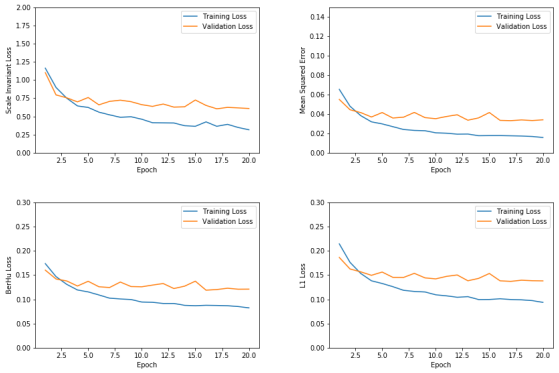

Figure 3. Plots showing different training loss functions with batch size 16 and learning rate as 2x10^{-5} for the Multi Scale architecture

Table 2. Table showing MSE, L1, BerHu and Scale Invariant loss for different learning rates and batch sizes for the PatchGAN architecture (T: Training and V: Validation)

Configuration	0	1	2	3	4	5
LR	5.00E-05	5.00E-05	5.00E-05	1.00E-05	1.00E-05	1.00E-05
BS	8	16	32	8	16	32
epochs	19	19	19	19	20	19
T:SIL	0.044	0.066	0.054	0.135	0.076	0.080
V:SIL	0.200	0.175	0.160	0.183	0.182	0.184
T:L1	0.073	0.090	0.082	0.135	0.098	0.099
V:L1	0.158	0.155	0.162	0.152	0.183	0.154
T:MSE	0.009	0.014	0.012	0.029	0.016	0.017
V:MSE	0.040	0.036	0.039	0.034	0.049	0.037
T:BerHu	0.063	0.075	0.067	0.109	0.080	0.080
V:BerHu	0.132	0.118	0.125	0.114	0.141	0.120

5. Results

Table 1 shows the training and validation losses for the mult-scale model for different learning rates and batch sizes. We observe the best results with batch size 16 and learning rate= 2x10^{-5} for training the Multi Scale architecture. Figure 3 shows the training and validation loss curves for the above mentioned hyper parameters. Table 2 shows the training and validation losses of PatchGAN model for different learning rates and batch sizes. We observe the best results with batch size 8 and learning rate= 1x10^{-5} for training the Patch GAN architecture. Figure 4 shows the training and validation loss curves for the above mentioned hyper parameters.

Figure 5 shows the predicted images by both the architectures against the original image and the ground truth depth maps from the test set of the NYU dataset.

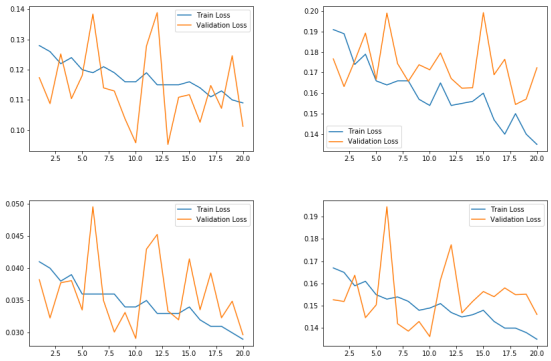

Figure 4. Plots for berHu (top-left), Scale-invariant (top-right), MSE (bottom-left) and L1 (bottom right) loss functions with batch size 8 and learning rate = $1x10^{-5}$ for the PatchGAN architecture

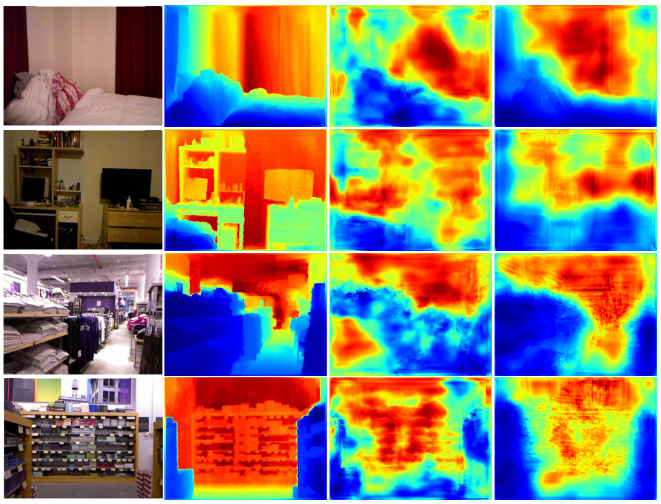

Figure 5. Original image, True Depth map, Depth Prediction by the PatchGAN, Depth Prediction by Multiscale respectively

6. Experiments

All of our models were trained on Nvidia GeForce GTX 1080 GPUs, and aside from VGG-16 had weights initialized randomly.

Our architectures were trained and tested on a 3 GB labelled subset of the NYU dataset [11] (428 GB). This comes out to 1448 images, of which the last 200 were saved for our validation dataset. The images come from an assortment of indoor scenes in bedrooms, bathrooms, kitchens and offices. Each sample of the training data contains a 320x240x3 pixel RGB image, and a corresponding 320x240 depth map, captured with Microsoft Kinect. The images were used in raw format, i.e. no down-sampling or dimension adjustments were performed. Because we only used a subset of the NYU dataset, the dataset was augmented with horizontal image flips and slight random rotations, approximately tripling the number of original images.

For comparison, a collection of losses (such as L1 loss, MSE loss, and scale invariant loss) were collected at a variety of batch sizes, learning rates and number of epochs for each architecture. These can be found in Tables 1 and Tables 2, and plotted at each epoch in figures 3 and 4. We found that our loss converged on the validation set at around 20 epochs for both networks, and that the scale invariant loss performed the best. More specific experiments for each architecture are detailed below in subsections.

6.1. Multiscale Experiments

The three scale architecture can be found in preceding sections. While Eigen and Fergus trained their networks independently, we opted to unfreeze the coarse and refined networks and train them together to evaluate if the results improved. We found that they were not superior. Additionally, we experimented with a variety of learning rates, batch sizes and loss functions. These can be found in Table 1. The initial experiments with MSE loss failed to converge and produced poor results, while the scale invariant loss performed better. We experimented with freezing and unfreezing the VGG-16 network (Scale 1), and found that results were stronger when VGG-16 was trained during backpropagation as well, but only at a small learning rate. At low epochs, or when VGG-16 was not trained on depth estimation, the network tended to confuse semantic features (such as squares, circles, and flat textures) as having high depth.

6.2. Patch GAN Experiments

The Generative Adversarial Network is an inherently complicated model, requiring careful choice of the Generator, Discriminator and an adequate loss function. For purposes of depth map estimation, it is essential that the Generator network functions as an excellent feature extractor. We experimented with two models for this very purpose, a ResNet [5] model and a U-net [12] style architecture. The results of the U-net architecture were visually more stimulating and we proceeded with that model. This can be attributed to the fact that the U-Net model having symmetric contracting and expansive halves, with skip connections, the network ensures that both local and global information can be preserved.

For the discriminator portion we experimented with patch sizes of varying dimensions starting from 30x30 and moving downwards till 11x11 in decrements of 2. There is an apparent trade-off between patch-size and the visual aesthetics of the results. A patch size of 19x19 was observed empirically to produce the most visually pleasing results.

The third and final experiment involved the use of different loss functions to train the PatchGAN architecture. As elaborated upon in section 4.2, the scale invariant loss function performs very well in tasks like depth estimation. In order to decrease the training time of our network, we used a

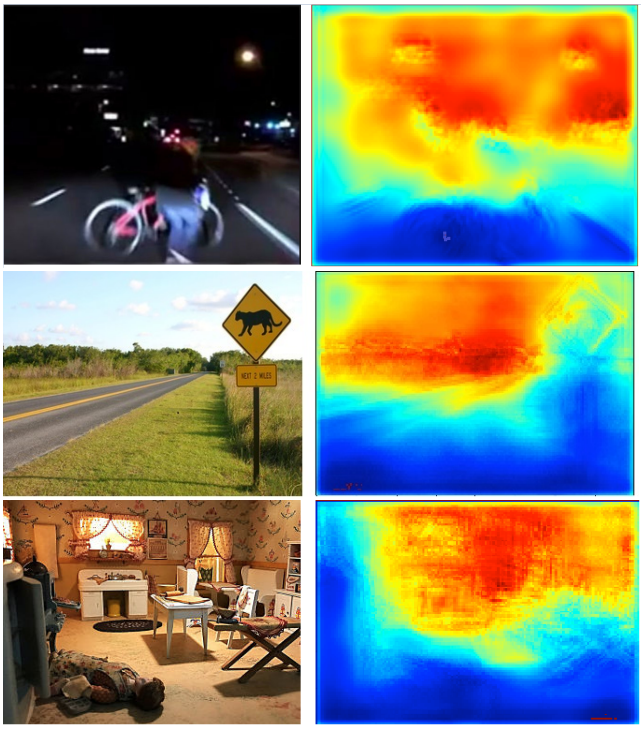

Figure 6. Original image and predicted depth maps using the Multi-Scale Architecture

weighted combination of the L1 loss and the scale invariant loss. The L1 loss function is excellent at allowing the network to generalize well as it penalizes outliers lesser. We experimented with 3 sets of weights namely [0.33,0.67], [0.5,0.5], [0.67,0.33]. Assigning a higher weight to the scale invariant loss function produced lower error rates and thus we assigned a weight of 0.67 to the scale invariant loss function and a weight of 0.33 to the L1 loss function.

7. Discussion

Empirically the PatchGAN model performed worse than the multi-scale network. The results on our validation sets, seen in Figure 5, emphasize that the GANs output creates splotchier, less cohesive depth regions. The cohesiveness of the multi-scale's depth could be attributed to its use of a coarse estimate, which finds rough mean estimates of large regions, or the VGG-16 feature extraction, which finds textures, shapes and edges. Cost wise, the multi-scale architecture was also less expensive to train: on the same dataset, the multi-scale network took around 3 minutes per epoch, while the GAN took around 20 minutes per epoch.

We tested the MultiScale architecture with some real world images outside the NYU [11] dataset. Figure 6 captures these experiments showing the original image and the predicted depth maps. We tested the network with a dark image and a bright image. In both these cases, the network predicted the depth quite well. While the depth predictor was unable to find the woman on the bike, the general outline of her body is the darkest blue (closest to the camera), and the region off the road is known to be far away (light red). The second image with the sign is notable because the network managed to outline the shape of the square cheetah sign, and segment it from the rest of the far-away sky, which emphasizes the benefit of VGG-16 as a feature extractor.

The third image was taken in a dollhouse. Dollhouse depth estimation is a canonical problem in monocular depth estimation, and elegantly demonstrates why the question is ill posed. The dollhouse example reinforces that our network learns relative depth (in absence of absolute depth cues) by showing that the "television" and doll are closer to the camera than the walls and fireplace.

8. Conclusion

In this paper, we have presented and compared two architectures for estimating depth from monocular images. We have extended the work of Eigen et. al. [2] by using pretrained VGG-16 as feature extractor, and have achieved quite good results. We have also demonstrated that our model is able to generalize to unseen data (Figure 6). We were optimistic about using GANs for this task as we believed that this problem could be seen as a domain adaptation one but were not able to achieve as good results as the multiscale model. This could be attributed to the small size of dataset that we used was insufficient to train GAN end-to-end. In terms of future work, we would like to train our models on larger datasets by using the full NYU datset or use Kitti [3] dataset which is also widely used for this task. In terms of architecture, we would like to combine the two architectures proposed in this paper by using a multi-scale GAN architecture in which we can have a generator-discriminator model at each scale in order to generate better depth maps.

9. Code

All the code for our work can be found at Github with Patch GAN model hosted at `https://github.com/manjotms10/learning-depth` and the multi-scale model hosted at `https://github.com/Gitikameher/Depth-Estimation-using-single-monocular-images`. All of our results can be reproduced on a single GPU by running the scripts corresponding to training the model in the respective repositories.

The scripts to download the NYU [11] dataset were borrowed from `https://github.com/MasazI/cnn_depth_tensorflow`.

References

[1] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[2] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network, 2014.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 2672–2680, Cambridge, MA, USA, 2014. MIT Press.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2017.

[7] 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.

[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] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. *2016 Fourth International Conference on 3D Vision (3DV)*, 2016.

[10] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. *CoRR*, abs/1411.6387, 2014.

[11] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention MICCAI 2015*, page 234241, 2015.

[13] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images, 2006.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.