Floors are Flat: Leveraging Semantics for Real-Time Surface Normal Prediction

Steven Hickson* shickson@gatech.edu
Karthik Raveendran† krav@google.com
Alireza Fathi† alirezafathi@google.com
Kevin Murphy† kpmurphy@google.com
Irfan Essa* irfan@cc.gatech.edu

Abstract

We propose 4 insights that help to significantly improve the performance of deep learning models that predict surface normals and semantic labels from a single RGB image. These insights are: (1) denoise the "ground truth" surface normals in the training set to ensure consistency with the semantic labels; (2) concurrently train on a mix of real and synthetic data, instead of pretraining on synthetic and fine-tuning on real; (3) jointly predict normals and semantics using a shared model, but only backpropagate errors on pixels that have valid training labels; (4) slim down the model and use grayscale instead of color inputs. Despite the simplicity of these steps, we demonstrate consistently improved state of the art results on several datasets, using a model that runs at 12 fps on a standard mobile phone.

1. Introduction

In this paper, we address the problem of learning a model that can predict surface normals and semantic labels for each pixel, given a single monocular RGB image. This has many practical applications, such as in augmented reality and robotics.

Most high-performing methods train deep neural networks to perform the task of estimating surface normals using large training sets. However, an often overlooked aspect of such approaches is the quality of the data that is used for training (and testing). We have found that the standard technique for estimating surface normals from noisy depth data, such as the widely used method of Ladicky et al. [16], can result in inconsistent estimates for the normals of neighboring points (see Figure 4 for an example). This results in methods with inferior performance.

We propose a simple technique to fix this, by regularizing the prediction of normals that correspond to the same surface. This encodes our intuition that floors should be flat and pointing up, etc. To estimate which pixels belong to the same surface, we leverage the fact that many depth datasets also have per-pixel semantic labels. This in itself does not tell us which facet of the object a pixel belongs to, but we use simple heuristics to solve this problem, as described in Section 3. See Figure 1 for an illustration of the benefits of this approach. Code is released to create the data here[1].

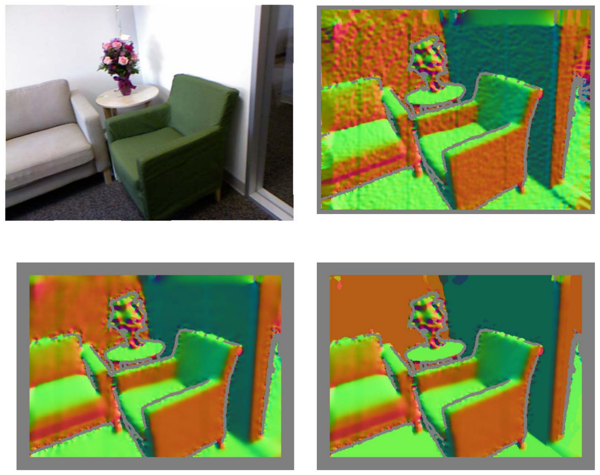
Figure 1: Visualization of different ways of computing "ground truth" normals. Top left: a sample image from the NYUDv2 dataset. Top-right: computed using method similar to [6] with a small window. Bottom-left: results of our method using larger depth-adaptive smoothing. Bottom-right: results of our method after semantic smoothing (if labels are available). Note that the back and right wall are cleaned up to a large degree due to this correction.

Unfortunately, even after such "data cleaning", most real world datasets are still too small to train deep models, so it has become popular to leverage synthetically generated images. These are noise-free, but it is not obvious how best to combine real and synthetic data. The standard practice (e.g., [18]) is to pretrain on the synthetic Scenenet dataset

*Google/Georgia Tech
†Google

and then fine tune on the real dataset. We propose a simple improvement to this idea, which is to train the model on a carefully chosen mix of real and synthetic images in each minibatch. This simple insight improves results considerably, as we show in Section 4.

In addition to improving the way data is used, we propose some improvements to standard modeling methods. In particular, we train a model to jointly predict surface normals and semantic labels, using an encoder-decoder network with two output heads. We take care to only back-propagate errors on outputs for which we have labels. We show this approach improves performance on both tasks, as we discuss in Section 5.

Finally, since most of the applications of depth estimation require a real-time method, we describe some tricks to make our model much smaller and faster, with negligible loss in accuracy. The result is a method that can run at 12fps on a standard mobile phone, while achieving state-of-the-art accuracy on standard benchmarks. Video demo here[2].

In summary, our main contributions are as follows:

- A method for computing reliable ground truth normals using depth adaptive computation with ”semantic smoothing”.
- A method for training by mixing synthetic and real data which gives state of the art results.
- A method that jointly learns semantics and surface normals in an end-to-end manner, increasing the accuracy of both.
- A way to make the model run in real-time (12 fps) on a mobile phone while still giving good accuracy.

2. Related Work

Traditional methods for estimating surface normals were largely limited by sources for ground truth data, and instead incorporated explicit priors such as shading, vanishing points [13], or world constraints [9]. With the advent of widely available and inexpensive depth sensors, data-driven approaches to this problem became more popular. Ladicky et al [16] introduced a discriminatively trained learning based algorithm by combining pixel and segment-level cues. Fouhey [7] and colleagues explored the use of learned sparse 3D geometric primitives and higher level constraints to predict surface normals.

In recent years, convolutional neural networks (CNNs) have proven to be a very effective means of tackling a wide range of image-level tasks. Wang et al. [29] introduced the first CNN-based method to solve the problem of dense surface normal estimation by fusing both scene level and patch level predictions. Eigen and colleagues [6] were the first to predict depth, surface normals, and semantic segmentation using the same multi-scale network architecture for each task (though not jointly). Bansal et al. [3] improved upon this architecture using skip connections and used it as input to jointly predict pose and style of 3D CAD models from an RGB image. In SURGE, Wang et al. [28] use a dense CRF to refine the output of their CNN model and demonstrate higher quality results on planar regions.

Researchers have also begun to explore the connections between various pixel level labelling tasks. Dharmasiri et al. [5] demonstrate that by having a single network jointly predict surface normals, depth, and curvature, they are able to almost match or improve upon networks tuned for these tasks independently. Similarly, Nekrasov [20] and colleagues explored the connections between depth estimation and semantic segmentation with a focus on the effects of asymmetric dataset sizes. Xu et al. [30] developed a prediction and distillation network that uses multiple intermediate representations such as contours and surface normals to achieve the final task of depth estimation and scene parsing. Similarly, Kokkinos [15] showed that a single unified architecture is capable of learning a wide range of image labeling tasks. A couple of works [21, 31] enforce consistency between joint predictions of depth and normals. In this paper, we show that semantic segmentation can improve normal prediction when predicting both jointly which results in even higher performance on planar regions. Some work, such as [17] have had success purely predicting planar regions instead of surface normals, though this limits the scope of the problem.

Another promising avenue for gathering data for surface normal estimation is synthetic rendering. Zhang et al. [32] train their normal prediction network on a large dataset of rendered images and fine-tune it on real data. Ren et al. [22] use an unsupervised domain adaptation method based on adversarial learning to transfer learned features from synthetic to real images. In this paper, we use synthetic data in our training but batch-wise mix it with real data in an end to end training setup.

3. Computing better ground truth normals

In this section, we discuss two simple methods for computing better ground truth normals from (real) depth datasets.

3.1. Datasets

We use two commonly used real-world depth datasets.

NYUDv2 [19] is a dataset of indoor environments taken with a Kinect device which results in approximately 450,000 640x480 RGB-depth pairs. 1449 of these images, split into a predefined train/test, have the depth pre-processed and have semantic labels for each pixel. This is the dataset used by most methods for evaluation. See Figure 2 for an example.

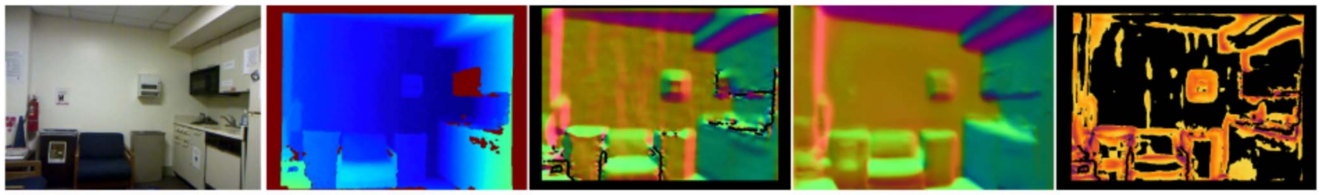

Figure 2: Illustration of the NYU data, together with the predictions of our model on them. Columns, from left to right: RGB image, depth, ground truth surface normals, our predictions, error image (where black is under 11.25 degrees errors, and then error increases from yellow to purple).

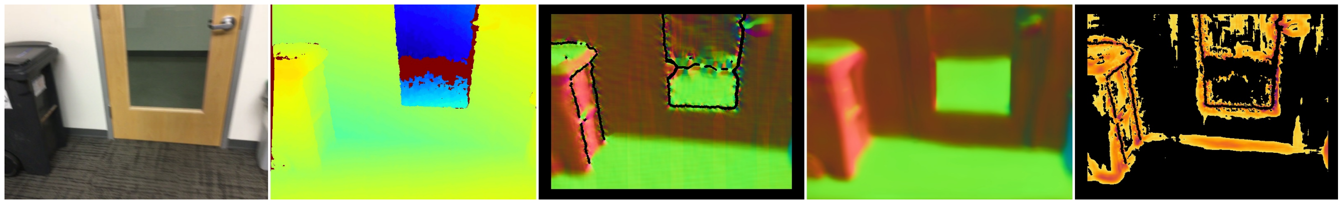

Figure 3: Qualitative evaluation on the Scannet data. Columns are the same as in Figure 2.

Scannet [4] is a dataset of approximately 2 million 640x480 RGBD sensor images that also include pixel-level semantic segmentation. Instead of 2D annotation like in NYUDv2, these are done in the full 3D environment and projected back into 2D. This allows it to have many more annotations compared to NYUDv2; however, due to this method, the annotations and edges don't match up as perfectly. See Figure 3 for an example.

3.2. Problems with current techniques

Surface normals for real world datasets are typically derived from the depth data captured by commodity depth sensors or stereo matching algorithms. For instance, the NYUDv2 dataset was captured using a Kinect v1, while ScanNet uses a similar Structure sensor. These sensors are well known to suffer from axial noise which is related to the distance of the surface from the sensor. As a consequence, surface normals that are computed from this data tend to exhibit artifacts that are noticeable, especially on distant planar regions.

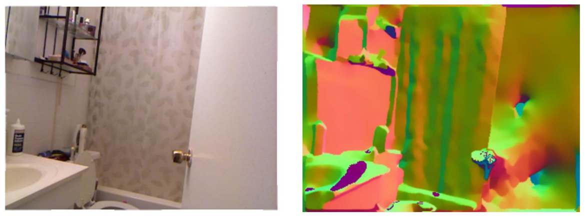

Figure 4: A common example of the errors seen in the normals from [16] that many use for training and evaluation. In these visualizations, (r, g, b) map to (x, y, z) of the normal at that location. Note the oversmoothing, which reduces and removes the normals of small objects. This image also demonstrates why it is important to only backpropagate on pixels that have valid depth, as the right side of the image has incorrect normal data due to noisy and missing depth values.

Broadly speaking, prior work has used one of two normal estimation methods to generate ground truth for training: least-squares estimation on a per-point basis using RANSAC after de-noising the point cloud [16], or local plane computation using the covariance matrix over a window [19]. In our experiments, we have found both of these approaches result in ground truth normals that have considerably more errors. For instance, in Figure 4, the former method produces oversmoothed and incorrectly oriented planar patches on regions like the sink, while Figure 5(d) produces highly noisy planar surfaces. We hypothesize that this could cause inferior results when used to train. In Table 1, we show that training on noisy ground truth normals results in a noticeable drop in accuracy. Training on [19] results in a mean angle error of 27.5 degrees, compared to 22 degrees when trained on our proposed normals. When visual inspecting the results shown in Figure 5 and Figure 4, it is obvious that past papers have been training and evaluating on erroneous data. Figure 5e) Statistics show that 20% of the planar angles are off by 90 degrees and 38% are off by 18 degrees. We correct this to give 0 planar angle error.

3.3. Improving Normals Using Point Clouds

We propose to use the method introduced in [14] to compute surface normals from a point cloud. We begin by smoothing the depth and filling holes as in [6] and then con-

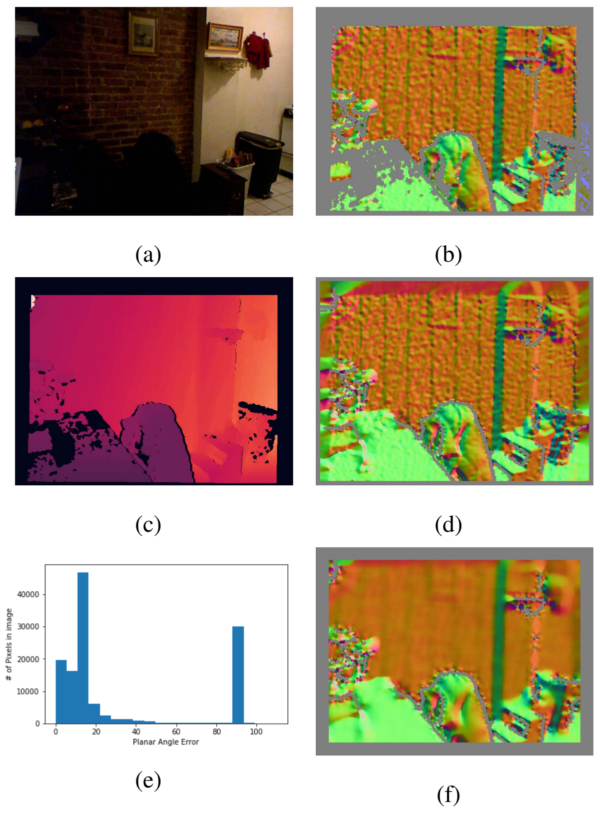

Figure 5: Effect of window smoothing size on surface normals. a) RGB image. b) Normals computed with a normal smoothing size of 10. c) In-painted depth. d) Normals computed similar to [19] from (c) using window size 10. e) The planar angle error of b). f) The normals with window size 30 that we train on (backpropagating only when c is valid).

struct a 3D point cloud with PCL [24]. A key insight here is to use a large depth-adaptive smoothing window that adequately compensates for the noise introduced by the sensor, while not smoothing over visually salient depth discontinuities. While this is a straight-forward approach, it is important to note that it has not been done by any of the previous papers and our ablation studies show it greatly improves results. For this, we use the integral image approach implemented in PCL [14]. Compared to [19], this samples a larger window more densely based off of the depth of the current point. We select a smoothing parameter of 30 for both real datasets (NYUDv2 and ScanNet) and 10 for the synthetic SceneNet dataset since it has minimal noise in its rendered depth estimates. We evaluate this method in Section 3.5.

3.4. Improving Normals with Semantics

To further reduce the amount of noise and get closer to absolute truth, we can leverage the semantics of the datasets. For certain semantic classes, e.g. walls and floors, we know that the results are usually planar (or at least piecewise planar). We use that information to smooth out the normals for those instances. Given that the datasets we are using all have some level of semantic information labeled, this is free contextual information.

While semantic segmentation gives us labels for objects, it does not distinguish between facets of the same class (for instance, walls facing in different directions). We perform an efficient post-processing step to identify regions with pixels that have consistent normals and semantic labels. We adapt the standard connected components algorithm to start at a pixel and grow the region outwards by adding pixels with normals that are within 30 degrees of the current averaged normal of the region, and of the same class as the starting pixel. We restrict this process to semantic labels that we assume to be planar. However, even if this assumption is violated, the normal variance constraint prevents arbitrary growth of these supposedly planar regions. Once we have computed the regions, we assign the averaged normal to all pixels of this region if the region is of a minimum size. An example of this is shown in Figure 1. We evaluate this method in Section 3.5.

3.5. Evaluation

	Accuracy			Error
Method	$\leq 11.25°$	$\leq 22.5°$	$\leq 30°$	Mean Angle
Baseline	46.2%	57.7%	63.8%	27.5°
Denoising	49.5%	64.6%	71.1%	22°
Semantics	60.6%	77.9%	83.4%	14.7°

Table 1: Accuracy and error rates of our normals model (without semantic output head) when evaluated on the NYUDv2 normals from [16]. First row shows results using our simple network training on standard normals as shown in the top right of Figure 1. Second row shows results using our denoising method (bottom left of Figure 1). Third row shows results using our semantic smoothing method (bottom right of Figure 1). Evaluation is performed on the normals from [16], which demonstrates the necessity of a dataset cleanup.

Table 1 shows an ablation study done on NYUDv2 with different types of training data all evaluated on [16] with the simple mobile encoder-decoder network described in Section 6 without the performance increases we discuss later. The first row shows the results when trained on the normals used by many papers as shown in the top right of Figure 1, the second row shows training on our normals computed from Section 3.3, the final row shows training on our proposed ground truth surface normals that are semantically corrected. The normals from Section 3.3 result in much better accuracy compared to [6] or [16], especially in the larger angle errors and the mean angle error. Leveraging semantics improves results even more. This simple idea improves the mean angle error by almost 13 degrees and reduces the smallest angle errors by 14%. This is a substantial increase better than most new architectures would yield.

4. Combining synthetic and real data

We train and evaluate our network on several publicly available datasets, both real and synthetic, to reduce our dataset bias and produce more robust normals, as we explain below.

4.1. Synthetic datasets

Scenenet [18] is a semi-photorealistic synthetic dataset of indoor environments comprised of ~ 4 million 320x240 images with corresponding depth and semantics. See Figure 6 for an example. We compute the normals for Scenenet using the method proposed in Section 3; however, we use a normal smoothing size of 10 given the input depth data is less noisy than data from conventional depth sensors.

4.2. Mixing real and synthetic

Since most prior work utilizes NYUDv2, our method was initially trained and evaluated only on it. However, we found this doesn't necessarily generalize well to other data, as shown in Table 3. (All results are obtained using the normals branch of the model architecture that is shown in Figure 8; see Section 6 for details.)

We also discovered that the standard practice of pretraining on Scenenet and finetuning on NYU results in a model that generalizes poorly. However, by simply mixing 10 synthetic scenenet images with 1 real image in every minibatch, we were able to improve performance on both datasets. Best results were obtained by mixing all 3 datasets, using 10 parts of Scenenet, 5 parts of Scannet, and 1 part of NYUDv2. Qualitative results on Scenenet are shown in Figure 6, on NYU are shown in Figure 2, and on Scannet in Figure 3. See Supplementary for more examples.

4.3. Comparison with the state-of-the-art

In Table 2, we show our results compared to previous state of the art surface normal estimation methods. Previous methods compute normals using the method of [19] or [16] applied to various datasets. Here we compare all methods by evaluating on NYUDv2; for baseline methods, we compute normals using the method of [16], whereas for our method, we use our approach for computing normals during training. For testing, all methods use the method of [16] to compute normals. We outperform the previous state-of-the-art (SOTA), [21], despite using a more than 100x smaller model, due to the higher quality of our data. For completeness, we also report the performance of our method when evaluated on our proposed method of computing normals from NYU; this test set is more accurate, and is also more similar to training, so we see performance is even greater.

	Accuracy			Error	
Method	11.25	22.5	30	Mean Angle	rmse
[8]	39.2	52.9	57.8	35.2	-
[16]	27.7	49.0	58.7	33.5	-
[5]	44.9	67.7	76.3	20.6	-
[28]	47.3	68.9	76.6	20.6	-
[21]	48.4	71.5	79.5	19	26.9
Ours	48.9	**72.3**	**81.2**	**17**	30.2
Ours on NYU'	**59.5**	**72.2**	77.3	19.7	**19.3**

Table 2: Comparison against state of the art on surface normal estimation task. All methods (except in the last row) are evaluated on the normals from the NYUDv2 dataset computed using the method of [16]; in the last row, we show our method evaluated on the normals from NYU computed using our method, which we denote by NYU'.

	Training set	
Accuracy	Scenenet+NYU FT	Datasets Mixed
NYU % < 11.25	57	**59.5**
% < 22.5	69.6	**72.2**
% < 30	74.6	**77.3**
Mean Angle Error	21.3	**19.7**
Scannet % < 11.25	36.7	**50.1**
% < 22.5	54.6	**63.2**
% < 30	60.9	**68.2**
Mean Angle Error	34.1	**28.8**
Scenenet % < 11.25	21	**64.5**
% < 22.5	48.2	**70.7**
% < 30	59.9	**68.2**
Mean Angle Error	37.6	**26.1**

Table 3: Normal Accuracy comparisons with different testing and training datasets. The columns are the training set used and the rows are the evaluation accuracy for each individual dataset. Scenenet+NYU FT means the standard practice of pretrained on synthetic and finetuned on NYUDv2. Datasets Mixed means the dataset is trained all from scratch with a batch-wise mix. The best result for each row is bold.

5. Jointly predicting semantics and normals

In order to evaluate the effect of combining normals and semantic labeling, we did an ablation study using just the Scannet dataset. We chose it because it is a large, complex, real dataset with both normals and semantic labels. We used the 13 semantic labels from NYUDv2 for our experiments. These consist of bed, books, ceiling, chair, floor, furniture, objects, picture, sofa, table, tv, window, and wall. To fairly

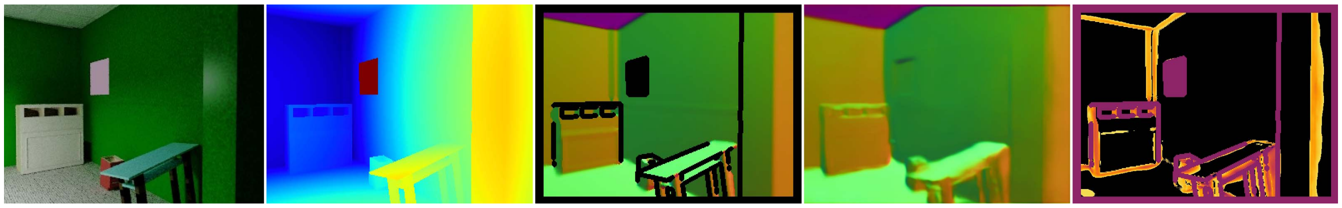

Figure 6: Qualitative evaluation on Scenenet data. Columns, from left to right: RGB image, depth, ground truth surface normals, our predictions, error image (where black is under 11.25 degrees errors, and then error increases from yellow to purple).

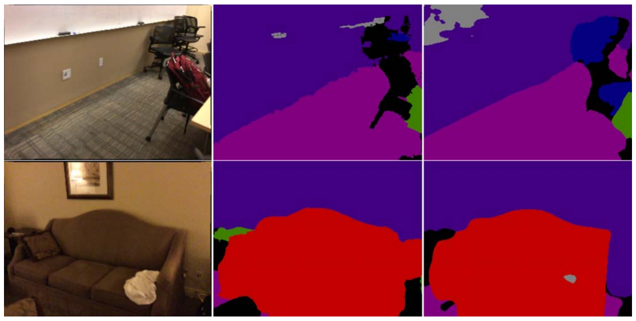

Figure 7: Examples of our semantic labeling predictions for the Scannet dataset. From left to right: RGB image, ground truth, our predictions.

compare, we train on only Scannet in this ablation study, and ignore other datasets.

5.1. Semantics

In order to train semantic labeling, we use our same architecture and training with slight changes. We change the regression output of 3 channels with a cosine loss to a classification output of 14 labels using a softmax cross-entropy loss. Experimentally, semantic labeling seems to be a harder task than surface normal estimation, so in order to train, we finetune our whole architecture initialized from our normals prediction with a lower learning rate (0.001). Results showing the need for this are included in the Supplementary material. Semantic only prediction is shown in the Semantics column of Table 4. See Figure 7 for some qualitative results. Note that there is occasional error in the ground truth of the semantics as well. Even though semantics are just an intermediate task for our method, our results are still very promising. Our prediction in Figure 7 correctly predicts both chairs as chairs (blue), even though the ground truth doesn't have this labeled correctly.

5.2. Joint Prediction

To train our method jointly, we duplicate the decoder using the architecture shown in Figure 8. We then finetune both the encoder and the dual decoders using the weights from our normal prediction network. The cosine and softmax cross-entropy losses are summed with a 20x weight modifier given to the cosine loss to balance them. An ablation study demonstrating why this is used is in the Supplementary material.

Contrary to prior work[15] that shows joint prediction reduces accuracy, our network improves when combining both semantics and normals. This is shown in Table 4. Surface normal estimation improves slightly (though it is important to note that small changes in the surface normal accuracy can still make large differences in practice due to the difference in angle error being so noticable when wrong). Interestingly, semantic labeling gets a large 6% increase in pixel accuracy. We hypothesize this is due to the importance of shape as a cue for semantic labels. Note that this actually outperforms the previous results evaluated on Scannet in Table 3 as well. It's possible that without new normals proposed in Section 3, this performance increase would not happen.

	Method		
Accuracy	Normals	Semantics	Joint
$\% < 11.25$	49.3	N/A	**50.9**
$\% < 22.5$	63.2	N/A	**65.2**
$\% < 30$	68.2	N/A	**70**
Mean Angle Error	29	N/A	**28**
Semantic Accuracy %	N/A	59	**65.6**

Table 4: This shows the results of Joint semantics and normals prediction on Scannet. The Normals column is the accuracy of a network only trained on Scannet normals. The semantics is the accuracy when only trained on Scannet semantics. Joint is the accuracy when both are trained concurrently as per Section 5.

6. Training a realtime model

In this section, we describe how to use the above techniques, combined with a lightweight model, to build a realtime mobile system with state of the art accuracy. We discuss our model size, training pipeline, and important tricks

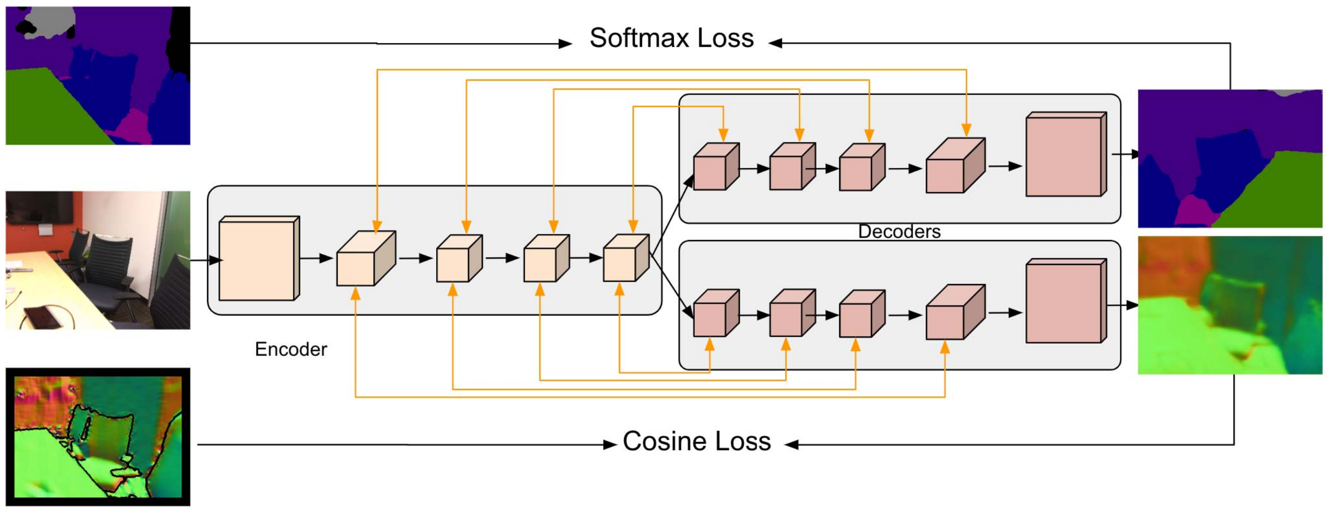

Figure 8: Our Architecture for joint prediction involves a shared encoder inspired by Mobilenet [25], followed by two U-net decoders. Each outputs its prediction and has a separate loss for either segmentation or normal prediction. The losses along with regularization are summed and optimized jointly. When doing just normal prediction, we simply drop the segmentation decoder and loss. See Section 6 for details.

(reducing the number of channels and utilizing grayscale) used to get the model on a mobile device.

6.1. Model

Prior approaches to the task of normal prediction have used feature extractors trained on VGG [26] or ResNet [12]. In contrast, we use a light-weight architecture that lends itself well to mobile applications. For our surface normal experiments and ablation studies, we use a modified version MobileNetV2 [25] encoder followed by the U-net decoder [23]. The key changes to MobileNetV2 are a normal residual instead of the inverted residual, PReLU [11] instead of ReLU, removing global average pooling, and increasing the convolutional filter size to 5.

For the decoder, we use U-net with 4 bilinear resizes, convolutions, and concatenations (see Supplementary material). These correspond to the blocks in MobileNetV2. The final output of the decoder is resized to the width and height of the input image (320 x 240 in our experiments), with the number of channels defined by the output task (i.e. 3 for normals and 14 for semantic labeling of NYU13).

After training our network, we can remove unnecessary ops and only use the normals encoder-decoder path by converting that model to a flatbuffer using Tensorflow Lite[10]. Our final network is under 2MB in size. We run inference using this model on the phone via ops implemented as OpenGL shaders.

6.2. Finetuning vs. Training from scratch

Conventionally, encoder-decoder networks use a larger encoder like ResNet101 (which is pretrained on Imagenet) and then fine-tune them for the specific task. However, for the task of surface normal estimation, we found that training from scratch in an end-to-end manner gave us better results. This could be due to the Imagenet dataset bias, our small network encoder, or the uniqueness of the task.

For training, when learning only surface normals in a single architecture as in our ablation studies, we use RMSProp [27] with a weight decay of 0.98, a learning rate of 0.045. When we train on surface normals and semantics, we fine tune off the surface normals model with a lower learning rate of 0.001.

6.3. RGB vs. Grayscale

Interestingly enough, for the task of surface normal estimation, color doesn't give much more of an advantage over grayscale data. This is shown in Table 5. This suggests that the neural network learns edges and color invariant features. This can potentially reduce the size and number of operations in a network. An ablation study on this is shown in Table 5.

6.4. Network Size

In Mobilenet [25], an encoder-decoder architecture is proposed with network size and speed described in the number of multiply-adds (MpAdds). We also test our normal prediction network as a function of network size in the same manner. The results are shown in Table 6.

	% of images grayscale		
Accuracy	0%	50%	100%
NYU $\% < 11.25$	59.1	59.2	59.1
$\% < 22.5$	72.2	72.2	72.2
$\% < 30$	77.4	77.3	77.3
Mean Angle Error	19.5	19.4	19.5
Scannet $\% < 11.25$	49.6	49.5	49.6
$\% < 22.5$	63.6	63.6	63.4
$\% < 30$	68.6	68.6	68.5
Mean Angle Error	28.8	28.8	28.9
Scenenet $\% < 11.25$	60.7	63.2	63.3
$\% < 22.5$	70.3	70.6	70.5
$\% < 30$	72.9	72.9	72.8
Mean Angle Error	27.1	26.5	26.5
Average $\% < 11.25$	57.075	57.625	57.7
$\% < 22.5$	69.025	69.05	68.975
$\% < 30$	73.325	73.3	73.225
Mean Angle Error	24.225	24.075	24.125

Table 5: This ablation study shows the effect of color on the network by changing the percent of input training images that are converted to grayscale. Interestingly enough, color does not seem that important for surface normal estimation.

	Channel Multiplier			
Accuracy	12	16	22	32
NYU $\% < 11.25$	56.1	57.1	58	59.3
$\% < 22.5$	67.7	68.6	69.3	69.6
$\% < 30$	72.3	73.1	73.7	73.9
Mean Angle Error	22.8	22.3	22	21.8
ScanNet $\% < 11.25$	44.5	46	47.6	50.1
$\% < 22.5$	60.3	61.4	62.3	63.2
$\% < 30$	65.9	66.9	67.6	68.2
Mean Angle Error	30.6	30	29.5	28.8
SceneNet $\% < 11.25$	59.9	61.5	62.3	64.5
$\% < 22.5$	68.8	69.6	70	70.7
$\% < 30$	71.4	72.1	72.4	78.2
Mean Angle Error	27.8	27.2	26.9	26.1
Average Eval $\% < 11.25$	54.08	55.13	56.48	58.15
$\% < 22.5$	66.55	67.35	68.20	68.63
$\% < 30$	71.10	71.85	72.53	74.10
Mean Angle Error	25.55	25.05	24.63	24.20
# million MpAdds	467	673	987	1624

Table 6: Here we use an ablation studty to test performance vs network size. The channel multiplier is a multiplier that determines the number of output channels calculated at each block. For instance, the final output of the encoder at channel multiplier 32 has 1280 channels, whereas, at channel multiplier 16, it has 640 channels.

A semantic segmentation model was also proposed by [25] with DeepLab as a decoder, where the last encoder layer is removed to minimize model size. The deeplab model is 2.75B MpAdds with stride 16 and 152.6B MpAdds with stride 8. We found that it is actually better to keep the last layer and just reduce the channel size of each layer; this results in a faster and smaller model. Our proposed network has 1.624B MpAdds at its largest, and only 467M for the mobile version, 300x smaller than the fast deeplab version. The other SOTA methods we compare against earlier in the paper, that utilize Resnet-101 or VGG-19, have between 91B and 5000B MpAdds, which is several orders of magnitude larger than our proposed network even though our method has much results, which is a large contribution.

6.5. Applications

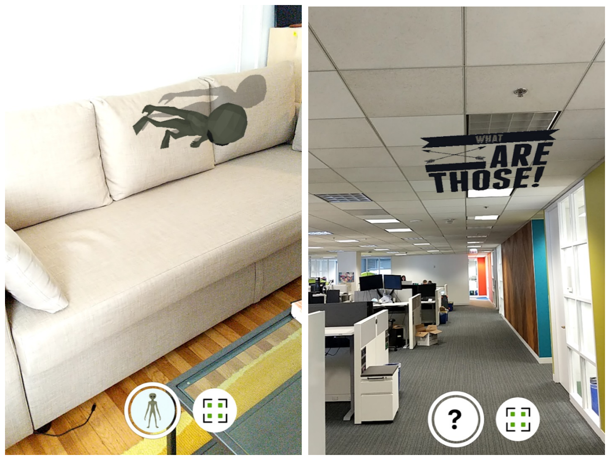

Figure 9: A sample AR application that uses the surface orientation to place a virtual character and text[2].

Using the channel scaling and other model minimization techniques discussed above, we created a lightweight architecture that runs at 12fps on a mobile phone. To demonstrate our SOTA results on normal estimation in real-time, we use this estimation to place stickers on surfaces in their natural orientation. A video of this demo is included in the supplementary material. Screenshots showing this demo running on the mobile device are shown in Figure 9. By simply clicking a region, the sticker or object can be placed realistically in AR using the predicted normals.

7. Conclusion

We have shown several simple methods for significantly improving the accuracy of any CNN method for predicted surface normals, namely: calculate the ground truth normals in a better way; combine real and synthetic data in a better way; and jointly train for normal prediction and semantic segmentation. We have also shown how to use these ideas to train a lightweight model that gives state of the art results, has low memory footprint, and runs at interactive rates on a mobile phone.

References

[1] Paper code. `https://github.com/StevenHickson/CreateNormals`.

[2] Paper video. `https://www.youtube.com/watch?v=QrXqmUBlmbc`.

[3] A. Bansal, B. C. Russell, and A. Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. *CoRR*, abs/1604.01347, 2016.

[4] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, volume 2, page 10, 2017.

[5] T. Dharmasiri, A. Spek, and T. Drummond. Joint prediction of depths, normals and surface curvature from rgb images using cnns. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 1505–1512. IEEE, 2017.

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

[7] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3d primitives for single image understanding. In *2013 IEEE International Conference on Computer Vision*, pages 3392–3399, Dec 2013.

[8] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3d primitives for single image understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3392–3399, 2013.

[9] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. pages 353–360, 12 2013.

[10] Google. Tensorflow lite. `https://www.tensorflow.org/lite/`, 2018. Accessed: 2018-11-14.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, Oct 2007.

[14] S. Holzer, R. B. Rusu, M. Dixon, S. Gedikli, and N. Navab. Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2684–2689. IEEE, 2012.

[15] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, volume 2, page 8, 2017.

[16] L. Ladicky, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014.

[17] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz. Planercnn: 3d plane detection and reconstruction from a single image. *arXiv preprint arXiv:1812.04072*, 2018.

[18] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison. Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 4, 2017.

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

[20] V. Nekrasov, T. Dharmasiri, A. Spek, T. Drummond, C. Shen, and I. Reid. Real-time joint semantic segmentation and depth estimation using asymmetric annotations. *arXiv preprint arXiv:1809.04766*, 2018.

[21] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 283–291, 2018.

[22] Z. Ren and Y. J. Lee. Cross-domain self-supervised multitask feature learning using synthetic imagery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[23] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[24] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (pcl). In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[28] P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, and A. L. Yuille. Surge: Surface regularized geometry estimation from a single image. In *Advances in Neural Information Processing Systems*, pages 172–180, 2016.

[29] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–547, 2015.

[30] D. Xu, W. Ouyang, X. Wang, and N. Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. *CoRR*, abs/1805.04409, 2018.

[31] Z. Yang, P. Wang, W. Xu, L. Zhao, and R. Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.

[32] Y. Zhang, S. Song, E. Yumer, M. Savva, J.-Y. Lee, H. Jin, and T. Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5057–5065. IEEE, 2017.