Contrastive vs Classification Pre-Training for Monocular Depth Estimation

William Ray
University of Massachusetts Amherst
wray@umass.edu

Abstract

In this work I compare contrastive and classification pre-training for monocular depth estimation on the NYU Depth Dataset V2 [4] with the ResNet50 architecture [3]. I explore three different training strategies to further train the two pre-trained models. I show that while the two pre-training strategies yield similar performance on this task, the classification pre-training objective outperforms the contrastive across all explored downstream training strategies.

1. Introduction

Depth estimation is a key component of many vision and robotics applications, but depth sensors are not as accessible as cameras and can not be applied to all applications because of their size and weight. Camera sensors are ubiquitous and highly accessible but lack the accuracy and robustness compared to depth sensors when applied to depth estimation tasks. Monocular depth estimation algorithms can enable cameras to estimate the depth of objects in a scene and developing better depth estimation algorithms can improve the reliability of camera-based depth sensors.

A contrastive training objective encourages a model to learn a representation of the data that is similar for related data points and dissimilar for unrelated data points. These kinds of objectives can train models in an unsupervised way and can learn powerful representations of the data [6]. Because contrastive objectives can be learned unsupervised, they can provide a useful starting point for problems when much of the available data is unlabeled. Exploring what types of information contrastive models encode in their intermediate representations can help us identify what problems may benefit from contrastive pre-training.

Contrastive methods like CLIP have been shown to have highly nuanced latent representations that capture many details of an image [6]. Perhaps other contrastive methods encode information related to depth. Investigating how a contrastive encoding can be used to recover depth information can indicate what information is contained in that encoding.

ResNet50 is a powerful computer vision architecture for processing images that improved the state of the art performance on ImageNet image classification [3], and is commonly used as a pre-trained model to apply to other downstream tasks. Pre-trained ResNet50 models exist for both classification and contrastive tasks. Classification models predict a probability distribution over many different classes and are trained to minimize the cross entropy between the predicted distribution and the true class.

2. Related Work

Ma et al. [8] applied the ResNet50 architecture to the task of monocular depth estimation and showed that a pre-trained ResNet50 depth estimation model performs much better than a randomly initialized model with the same architecture trained in the same way. The authors trained an architecture similar to the one explored in this paper, where a ResNet50 encoder was followed by a decoder that upsamples the last intermediate feature map to a depth map. The biggest difference with their work and mine, is I train a contrastively pre-trained ResNet model. Additionally I used a larger subset (500 vs 1,152 images) of the NYU Depth Dataset V2 and trained my models for 70 epochs instead of 20 epochs [4]. This work showcases the importance of pre-training when applying ResNet50 to the task of depth estimation.

Chen et al. [7] applied a more advanced architecture to the problem of monocular depth estimation. They presented a novel architecture that used low-level, mid-level, and high-level feature maps, where deeper feature maps are considered to be higher level. The mid-level feature maps were processed with a stack that used residual blocks, and all three stacks were concatenated into a single feature map that was then upsampled with upconvolutions into a depth map. This is comparable to my work, but the models in this paper do not have access to low or mid-level features maps. Their model proved to be state of the art at the time of their publication.

3. Methods and Architecture

I use one model architecture to compare two pre-training tasks across three training strategies for the downstream task of monocular depth estimation.

The architecture is a ResNet50 [3] encoder paired with a depth prediction decoder comprised of convolutional transpose layers.

The two pre-training tasks are 1. classification - a classification task on ImageNet images and 2. contrastive - a contrastive learning objective on ImageNet images.

The three training strategies are 1. fine-tuning - where the encoder and decoder are trained end-to-end 2. probing - where the decoder is trained and the weights in the encoder are frozen and 3. switch - switches from probing to fine-tuning halfway through training, where for the first half of training the encoder is frozen, but during the second half of training the encoder unfreezes and the model is trained end-to-end.

Dataset

I trained and evaluated the models on the NYU Depth Dataset V2. I only used the cleaned portion of the dataset which consisted of 1,449 RGB images and their corresponding depth maps. I made this decision because the cleaned depth maps were missing no data, unlike the depth maps in the raw section of the dataset. The raw dataset was also over 400GB, compared to the cleaned dataset, which was just under 3GB [4]. The models likely would have benefitted from a larger dataset as they appeared to overfit somewhat to the training dataset. To address this I applied some light data augmentation to the training dataset.

Data Processing

I scaled the values in the RGB images from 0-255 to 0-1 and scaled the depth maps to roughly have mean 0 and standard deviation 1. The resolution of the images was 640x480, but I processed the images to a 224x224 resolution which is the resolution the ResNet architecture was originally trained on [3]. To accomplish this scaling I resized the images to 320x240 using bilinear interpolation and then took a random 224x224 crop. By cropping the images randomly, each epoch had slightly different data which increased the variety of the training data. To further increase data variety I flipped the images horizontally with probability 0.5. The random crop and horizontal flip were applied uniquely across each batch during training. The original ResNet50 paper applies a similar data augmentation to their dataset [3].

Classification ResNet50

I used the pre-trained ResNet50 model contained in *torchvision.models.resnet50* [5] from *Deep Residual Learning for Image Recognition* as the encoder of the model pre-trained on classification. This ResNet50 was trained on 1.28 million ImageNet images to classify into 1,000 different classes [3].

Contrastive ResNet50

For the encoder of the contrastive model, I used a pre-trained ResNet50 acquired from the SimCLR PyTorch repository [1]. This repository contains an unofficial implementation of Ting et al. [2]. The model I used was trained for 600 epochs with a batch size of 2,048 on the ImageNet dataset. The pre-training task minimized a contrastive loss across augmented images. Such augmentations included, a random crop with flip and resize, color distortion, and Gaussian blur. The contrastive loss function used was the Normalized Temperature-scaled Cross Entropy function [2].

ResNet50 as an Encoder

ResNet50 has a 2,048x7x7 feature map followed by a pooling layer and dense layers [3]. I use this feature map as an input to my decoder and discarded the original ResNet50 layers following the 2,048x7x7 feature map. Ma et al. [8] showed that this feature map could be successfully applied to the task of monocular depth estimation. I also prepended a batch normalization layer to the input of the encoder to give the model further control over the scale and offset of the input RGB image.

Depth Prediction Decoder

The depth prediction decoder contains five upsampling blocks. Each upsampling block consists of a convolutional transpose layer followed by a convolutional layer. The convolutional transpose layer uses a strided 3x3 kernel to upsample the input by a factor of two and reduces the depth of the feature map by a factor of four. The convolutional layer has a 5x5 kernel and maintains the depth of the feature map. The fifth upsampling block reduces the depth from 8 to 1 with its convolutional transpose layer. After the fifth upsampling block a batch normalization layer is applied. Each intermediate feature map has the same number of elements as the original 2,048x7x7 feature map.

Training the models

In total I trained 30 models with six different configurations for 70 epochs each. Three configurations used the classification ResNet50, and three configurations used the contrastive ResNet50. For each ResNet50 pre-training task, I experimented with three different training strategies; fine-tuning, probing, and switching from probing to fine-tuning halfway through training. During probing and the first half of the switch configurations the encoder was frozen and its weights were not updated. During the fine-tuning configurations the model was trained end-to-end. Several characteristics were held constant across the different training configurations. All configurations used a mean squared error loss function between the model's outputs and the ground truth depth maps, and all configurations used the Adam optimizer with a learning rate of 0.0001 and a batch size of 64. These hyperparameters are not the result of an extensive hyperparameter search and its possible that they are not the best hyperparameters for the model.

In each configuration the weights in the decoder were randomly initialized and the order and augmentation in the training data varied between configurations. To account for this variation and to achieve more consistent results I trained each configuration five times and averaged the results.

The samples contained in the train, validation, test split of the data were held consistent across all experiments. The training set had 1,152 samples, while the validation and test set had 144 and 145 samples respectively.

4. Experimental Results

I evaluated the six training configuration on five different metrics. Mean squared error (loss), mean relative error (rel) and percent accuracy with three different threshold values ($1.25, 1.25^2$ and 1.25^3.)

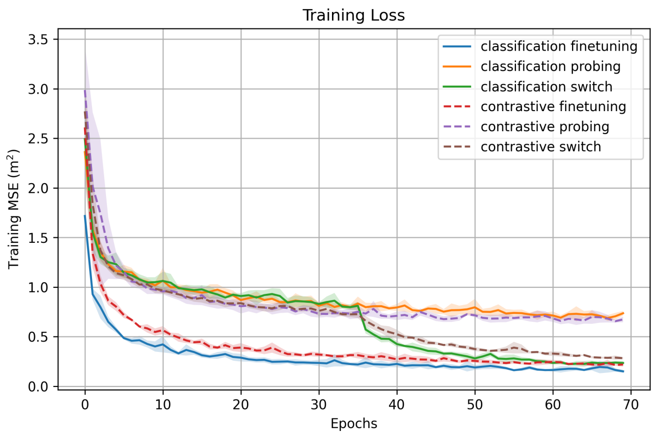
$$\text{loss} = \frac{1}{N}\sum_{i=1}^{N}(p_i - p_i^*)^2 \tag{1}$$

$$\text{rel} = \frac{1}{N}\sum_{i=1}^{N}\frac{|p_i^* - p_i|}{p_i^*} \tag{2}$$

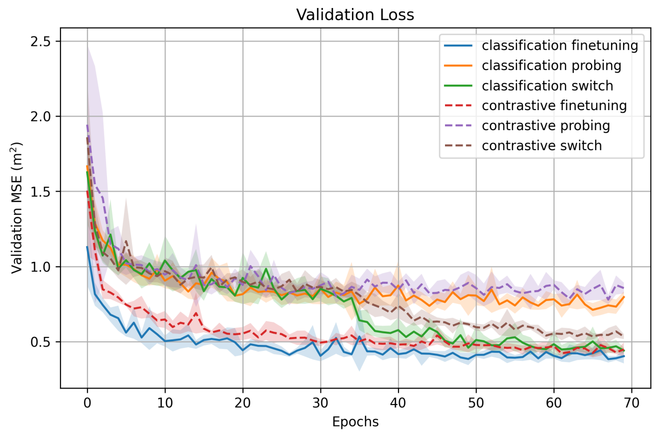
$$\begin{gathered}\delta_i = \max\left(\frac{p_i}{p_i^*}, \frac{p_i^*}{p_i}\right) \\ I_i = \begin{cases} 1 & \text{if } \delta_i < threshold \\ 0 & \text{otherwise} \end{cases} \\ \%\ \text{acc} = \frac{1}{N}\sum_{i=1}^{N} I_i \end{gathered} \tag{3}$$

Where p_i is the predicted depth and p_i^* is the ground truth depth of pixel i. Mean relative error and percent accuracy were used as metrics in Chen et al. [7]. Metrics were calculated across all pixels in the testing data, and then further averaged over different training configurations.

Across all three training strategies the classification pretraining appeared to be superior to the contrastive pretraining at the task of monocular depth estimation. This contradicts my initial hypothesis that the contrastively pretrained model would be able to store more nuanced information about the images and would therefore be able to better predict the depth of the images. Evidently the classification objective caused the model to encode more depth information than the contrastive objective.



(a) The average training loss calculated over each epoch of all six training configurations.



(b) Validation loss calculated over the validation data of all six training configurations.

Figure 1: Each line represents the average of five different training runs for a particular configuration. The shaded region around the line represents the standard deviation of the training runs. The models pre-trained with a classification object are shown as a solid line, where as the models pre-trained with a contrastive object are shown as a dashed line. The units of the Y-axis are scaled to be in meters^2.

Switching from probing to fine-tuning halfway through training did not improve performance over fine-tuning the model from the beginning. I had feared the encoder may have been getting degraded at the beginning of training because its gradient updates were coming from a randomly initialized decoder. I thought freezing the encoder until the decoder had been trained may make the gradient updates more stable. If this hypothesis was correct I would expect the switch training configuration to achieve better performance than the fine-tuning configuration, but that is not the case. Perhaps backpropogating through a randomly initialized network does not degrade pre-trained layers.

	loss	rel	$\delta \leq 1.25$	$\delta \leq 1.25^2$	$\delta \leq 1.25^3$
contrastive probing	0.814	0.266	0.566	0.856	0.954
classification probing	0.846	0.245	0.554	0.855	0.959
contrastive switch	0.570	0.210	0.663	0.914	0.979
contrastive finetuning	0.479	0.190	0.704	0.935	0.987
classification switch	0.460	0.183	0.713	0.941	0.987
classification finetuning	0.412	0.178	0.744	0.948	0.989

Figure 2: Average results of the six training configurations evaluated on the testing data. Lower loss and rel are better, for all other metrics higher is better. Training configurations are sorted by rel from worst to best.

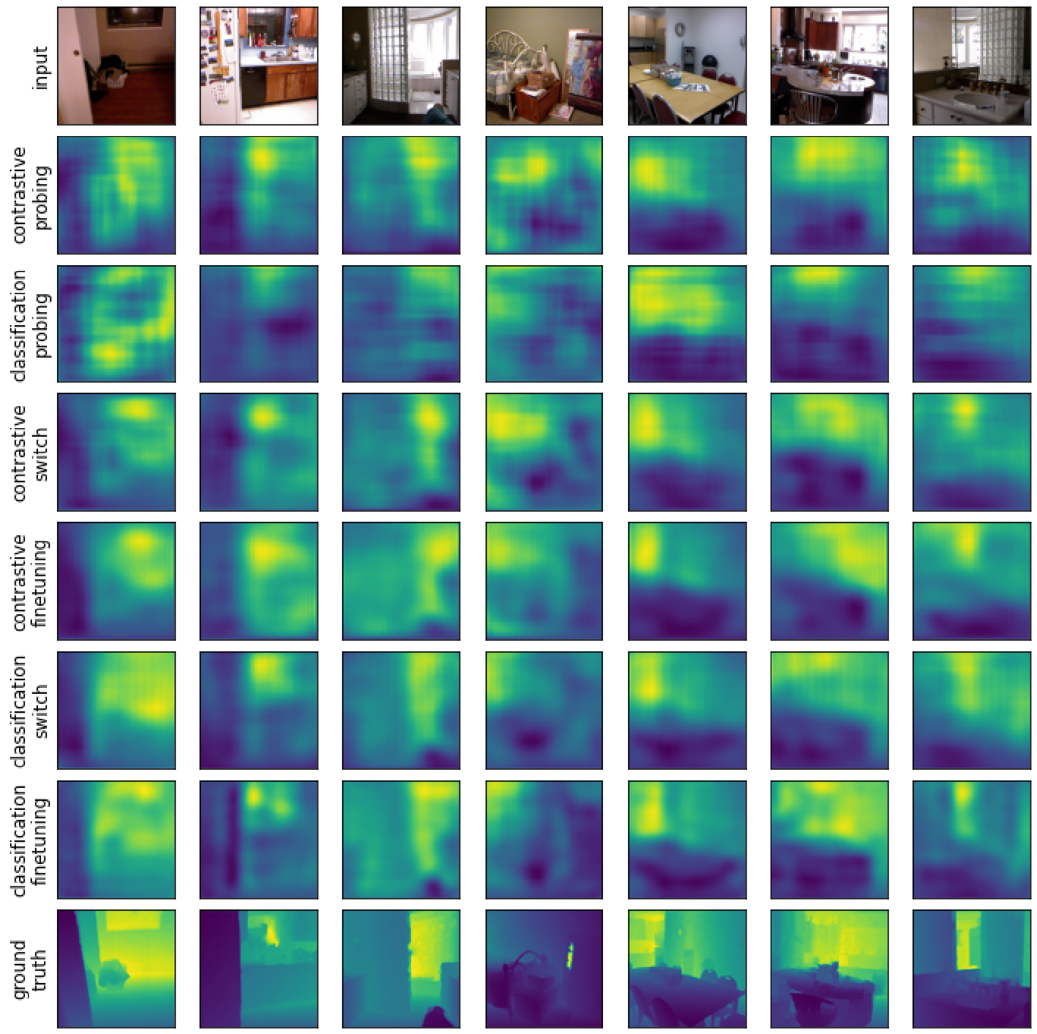


Figure 3: Depth estimation on seven random images from data withheld for testing. Each row is the output of one training configuration. The rows are in the same order as in figure 2. The two rows associated with probing contain grid-like artifacts. This may suggest that my decoder is under-powered and cannot learn to predict a smooth depth map without the help of the encoder. The higher performing models have noticeably more detail in their depth estimations. This leads me to conclude that if the decoder had access to lower-level feature maps and it could apply those lower-level features to generate more detailed depth maps, such a model would perform well on this task when measure with these metrics.

Limitations

I did not train my architecture from a random initialization. Doing so would have made for a good baseline to see how much the classification and contrastive pre-training improves performance.

The models output very blurry depth maps. They appear to understand the general regions of depth, its possible if the decoder had access to lower-level feature maps in the resent model it would be able to incorporate more spatial detail into the final depth estimations.

Acknowledgements

I want to say thank you to my professor Dr. Evangelos Kalogerakis for giving me a new perspective on deep learning in the 3D setting and helping me develop the skills to tackle a project like this. Thank you for a great course! I also want to thank the TAs Dmitrii Petrov, Zhan Xu, and Difan Liu for their help and guidance. Thank you to Kamran Zolfonoon for providing the partial inspiration for this project.

References

[1] AndrewAtanov and senya ashukha. Simclr pytorch. `https://github.com/AndrewAtanov/simclr-pytorch`, 2021. 2

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. 2

[3] He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 2

[4] Silberman Nathan, Hoiem Derek, Kohli Pushmeet, and Fergus Rob. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1, 2

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 2

[6] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 1

[7] Chen Songnan, Tang Mengxia, and Kan Jiangming. Predicting depth from single rgb images with pyramidal three-streamed networks. *Sensors*, 2019. 1, 3

[8] Ma Xiaobai, Geng Zhenglin, and Bie Zhi. Depth estimation from single image using cnn-residual network. In *Stanford CS231n Reports*, 2017. 1, 2