Reconstruction and Classification Based on Reduced Image Information

Qianyun Lu
Electrical Engineering
Stanford University
qylu@stanford.edu

Abstract

This project studies the 2D/3D reconstruction and classification based on highly compressed data: quantized linear gradients. The original image can be perfectly reconstructed from the full precision linear gradients. 2.25-bit linear gradients result in less precise reconstruction, but still contain enough information for simple visual tasks like classification. Based on the 2D reconstructions, 3D carving is performed, which can be more accurate than using the original images. Additionally, tiny convolutional neural networks are trained on RGBD images using the same compressing technique. Experiments show that the models achieve high accuracy in spite of the aggressive compression, where the depth information contributes a lot. A model trained on 1.5-bit linear gradients achieves $> 93\%$ *accuracy with only 120 parameters, where the data compression rate is about* 5.33*.*

1. Introduction

Artificial intelligence has created enormous value, and therefore it is called the new electricity. Apparently, machine learning (ML) is one of the most powerful tools and has transformed many industries, including computer vision. Mathematically, neural networks use gradient descent to minimize the fitting loss to the data, while convolutional neural networks (CNNs) are the most effective and widely used architectures in vision tasks. Generally speaking, machine learning is energy-hungry. Take image classification as an example, theoretically, CNNs can correctly classify all examples as long as they have "seen" enough images. In ML classes, we are often told to augment the datasets when our models perform poorly. This leads to a blind faith in "big" data. Too much data can also result in other problems even though there is a possible performance enhancement.

Many CNNs are trained on huge datasets. For example, ImageNet takes up $\sim$ 150 GB and COCO takes up $>$ 35 GB. In order to fit the nonlinearities of these data, large networks such as VGG-16 & ResNet-101 [5] are often needed. Large data sizes and network sizes consume tremendous computational resources accompanied by huge carbon emission. This has also limited the low-end applications due to the high costs. To solve these issues, many researchers are developing compression techniques [4] or new network architectures. Compact networks have emerged, *e.g*. MobileNets [11], SqueezeNets [6], Once-for-All [2].

Figure 1. Humans can easily detect the girl and the horse although the sketching contains much less information than photos.

Among all the solutions, most of them are about shrinking the model size by either compression or architecture search. It seems, however, people seldom question about the data. Is it always better to have more data? In fact, more data might hurt [1]. This inspires the idea to reduce the data without hurting the model performance.

Intuitively, we can compare the machine vision to human perception. A person can classify objects based on very limited information. For example, we can tell if contours belong to people or horses without knowing their skin color or facial expression. Figure 1 shows another example: it is easy for us to detect the girl and the horse even though it contains much less information than a real photo. By this analogy, we can see that there probably exists much redundancy in the data for simple vision tasks like classification. What interests me is the possibility to refine the input data to CNNs.

Figure 2. Edges of a car and the original photo.

2. Related Work & Approach

Prior works provided some evidences. Histograms of oriented gradients (HOG) showed excellent precision in human detection [3]. There were also experiments on aggressively quantized logarithmic-gradients [14]. However, the authors did not prove the sufficiency of using gradients only.

On the other hand, almost all state-of-the-art CNNs are trained on single view images, which is not sufficient under certain circumstances. In view of this, adding depth information can help. For instance, LIDAR data, *i.e*. depth, can be combined with RGB for pedestrian detection [10, 12].

In this project, I will justify the practicability to establish computer vision systems on reduced image data. To find out how much data can be recovered from a highly compressed image, I perform 2D reconstruction using CNNs. The 2D reconstructions are also utilized for 3D reconstruction. Additionally, same compression technique is applied to RGBD data and then used for person-presence detection. More concretely, this project explores the following points:

(i) *2D image construction based on linear gradients.* I recover the lost information based on just edges as illustrate by Figure 2. These edges are called 'linear' gradients ($L\nabla$) in this project, which are extracted by applying a naive filter. Techniques to reconstruct multi-channel images are also studied.

(ii) *3D image reconstruction based on linear gradients.* Based on the successful reconstruction of 2D images, 3D reconstruction is performed, which involves a small ablation of the precision of $L\nabla$.

(iii) *Classification based on reduced RGBD data.* Experiments show that $L\nabla$ contain sufficient information for simple visual tasks like classification. Accordingly, I train CNNs on these highly compressed datasets. Additionally, the depth data is used to compensate for the information loss due to the aggressive compression.

According to my experiments, the compression technique reserves sufficient information for 2D/3D reconstruction as well as highly accurate classification. This will yield more efficient computer vision systems which are friendly to edge devices.

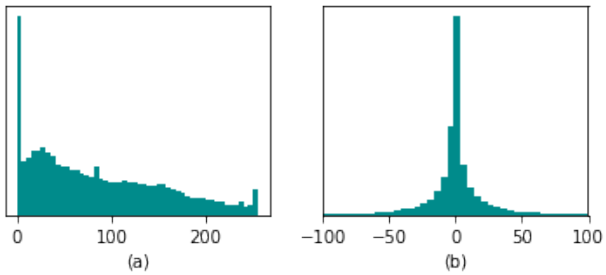

Figure 3. Pixel distribution of (a) grayscale images (b) $L\nabla$.

3. 2D & 3D reconstruction

3.1. Linear gradients ($L\nabla$)

Denote an 8-bit image by $P \in \mathbb{R}^{H\times W}$, where H, W are the height and width in pixels, $P_{ij} \in [0, 255]$. In this project, the linear gradients ($L\nabla$) of an image, denoted by G, are computed as follows:

$$G = P * f \qquad \text{where} \quad f = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{1}$$

Accordingly, $G_{ij} \in [-255, 255]$. For RGB data, the gradients are computed channel-wise. Grayscale images are employed in this project to use as little information as possible. To my understanding, taking $L\nabla$ is a kind of normalization; it redistributes the pixels and always gives a near-normal distribution, as shown in Figure 3. G is further quantized to lower precision to reduce the data volume. Denote the image after quantization by Q.

3.2. Dataset & CNN architecture

Dataset: For the 2D image reconstruction, I use a subset of PASCAL VOC 2007. Specifically, more than 7 thousand objects of 10 classes from the provided trainval set are cropped, resized to $(96, 96)$, and converted to grayscale. They are split into train:val:test = 5171:1108:1109. I also use other RGB photos for validation, which will be explained later.

Network architecture: Here I use a naive CNN that only contains two convolutional layers, as shown in Figure 4. Tunable parameters include the kernel sizes f_1, f_2 and the channel number $\#_c$. Note that the second Conv2D only has 1 channel because grayscale images are used. Additionally, mean squared errors are computed as the loss.

3.3. Evaluation metric & 2D reconstruction

Since we want to minimize the difference between the output and input images of the CNN, the evaluation metric

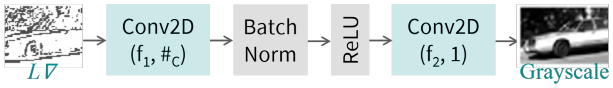

Figure 4. Naive CNN for 2D reconstruction.

Figure 5. Left: grayscale image. Middle: 2.25-bit $L\nabla$. Right: reconstruction.

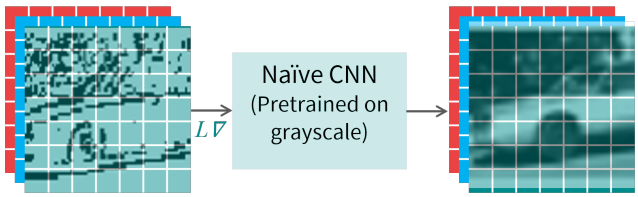

Figure 6. Channel-wise & block-wise reconstruction.

Figure 7. Left: RGB image. Right: 2.25-bit $L\nabla$ reconstruction.

is selected as mean square errors (MSEs). The linear gradients are quantized to 2.25 bits. In other words, for each pixel Q_{ij} in the quantized image, $Q_{ij} \in \{-2, -1, 0, 1, 2\}$. An example reconstruction based on 2.25-bit $L\nabla$ is shown in Figure 5. Despite the noises, most information in the original grayscale image is preserved. The image resolution is $(96, 96)$.

Further, we consider multi-channel images of higher resolution, *e.g.* RGB pictures. Large images are reconstructed channel-wise and block-wise, and at last stitched together, as indicated by Figure 6. Figure 7 shows the reconstruction of an image from PASCAL VOC 2007. Figure 8 displays examples using different block sizes (the image is from problem set 3).

3.4. 3D reconstruction

According to 2D reconstruction, a large amount of information is preserved in the quantized $L\nabla$. Then it should also give acceptable 3D reconstruction. Here I use the code

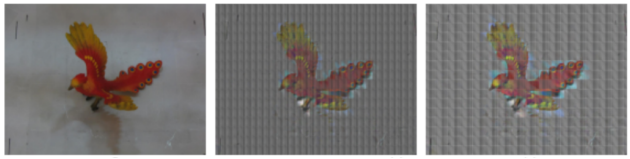

Figure 8. Left: RGB image. Middle: 2.25-bit $L\nabla$ reconstruction, small blocks. Right: 2.25-bit $L\nabla$ reconstruction, large blocks.

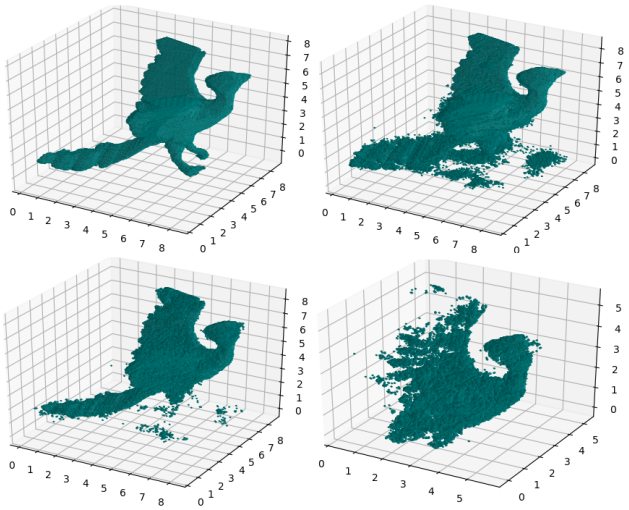

Figure 9. Upper left: use true silhouette. Upper right: use estimated silhouette based on the original RGB image. Lower left: use estimated silhouette based on full-precision $L\nabla$. Lower right: use estimated silhouette based on 1.5-bit $L\nabla$.

for space carving using estimated silhouettes from problem set 3; I modified the function for better estimation of silhouettes. Figure 9 shows the reconstruction based on the original RGB image, full-precision $L\nabla$, and 1.5-bit $L\nabla$. The reconstructions based on full-precision $L\nabla$ and the RGB image are comparable: some parts are noisier, some parts are less noisy. For 1.5-bit $L\nabla$, the reconstruction is not successful because of the aggressive quantization.

3.5. Discussion

To sum up, most information is reserved in aggressively quantized images, which is sufficient for simple visual tasks like classification. There are certainly information loss as implied by the stripes in the reconstructions. Theoretically, the loss results from the removal of absolute values of pixels. For example, J_1 and J_2 have the same linear gradient. Quantization also leads to loss. For future work, it would be interesting to remove the stripes/edges of the tiles. An intuitive solution is to also save one horizontal and one vertical

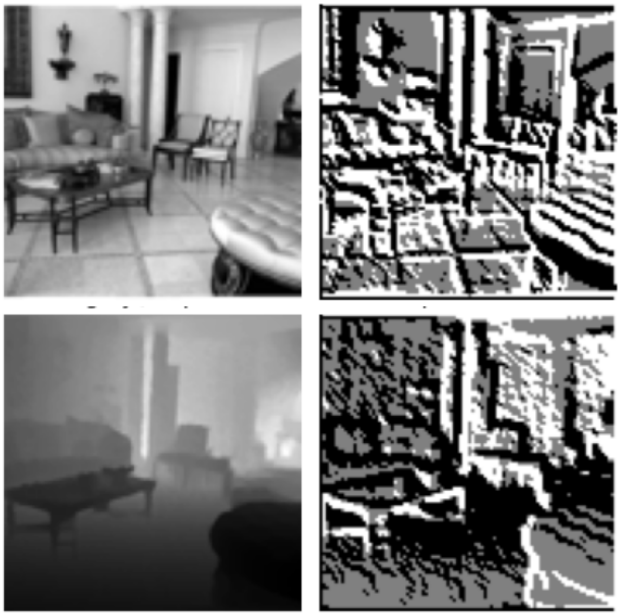

Figure 10. Upper left: grayscale image. Upper right: 1.5-bit $L\nabla$ of the grayscale image. Lower left: depth image. Lower right: 1.5-bit $L\nabla$ of the depth image.

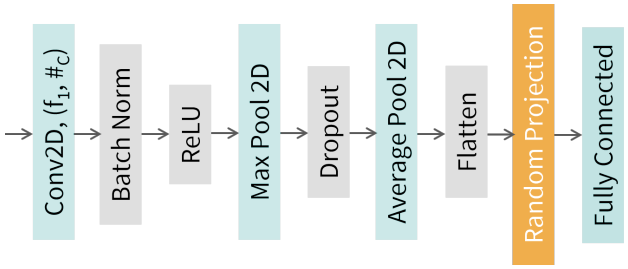

Figure 11. A simple CNN with random projection.

Label:Class	0: 'background'	1: 'person'	Total
# of examples	1449	723	2172

Table 1. Person presence dataset.

edges of each image.

$$J_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 2 \\ 0 & 2 & 0 \end{bmatrix} \qquad J_2 = \begin{bmatrix} 0 & 254 & 0 \\ 254 & 0 & 255 \\ 0 & 255 & 0 \end{bmatrix}$$

In terms of 3D reconstruction, the full-precision $L\nabla$ are quite noise-resistant. With improved 2D reconstructions, it is possible to get better 3D reconstructions as shown in Figure 9.

4. $L\nabla$-based RGBD image classification

As introduced in course lectures, single view can cause problems, especially for highly compressed data, *e.g*. 1.5-bit linear gradients in my case. However, if depth information is available, the problems may be solvable. Next, I will work on RGBD datasets and provide a solution to low-power devices based on $L\nabla$.

4.1. Dataset & CNN architecture

Dataset: I build my own dataset based on NYU depth v2 and RGBD people's dataset [9, 13, 8]. As shown in Table 1, 2172 images of two classes, 'background' and 'person', are converted to grayscale and concatenated with the depth image for classification. In other words, each image has 2 channels of data, one from the grayscale and the other from the depth image. These images are further split into train:val:test = 1520:326:326. Similarly, compute and quantize the linear gradients of the grayscale and depth images; examples are shown in Figure 10.

Network architecture: It turns out that a simple CNN suffices for this task. Figure 11 shows the CNN architecture, which incorporates random projection to further reduce the model size [7].

4.2. Evaluation metric & experiments

This is a classification problem; therefore, the cross entropy loss is used and accuracy is computed for evaluation. To prove that the depth information is helpful and it can also be compressed, I tested 6 cases as grouped below. Here, 8-bit corresponds to full-precision.

4.2.1 8-bit grayscale + 8-bit depth *vs*. 8-bit grayscale

In this case, both the grayscale images and the depth images are full-precision, *i.e*. 8-bit. As shown in Figure 12(a), the depth information can largely improve the accuracy when the model is small. As the number of parameters increases, the gap narrows down.

4.2.2 8-bit $L\nabla$ of grayscale + 8-bit $L\nabla$ of depth *vs*. 8-bit $L\nabla$ of grayscale

Now compare the results of full-precision linear gradients. Same as before, the accuracy is higher with an extra channel of depth, and the gap in between is smaller with more parameters, as displayed in Figure 12(b).

4.2.3 1.5-bit $L\nabla$ of grayscale + 1.5-bit $L\nabla$ of depth *vs*. 1.5-bit $L\nabla$ of grayscale

Figure 12(c) compares the models trained on 1.5-bit $L\nabla$. Again, the depth images can significantly improve the accu-

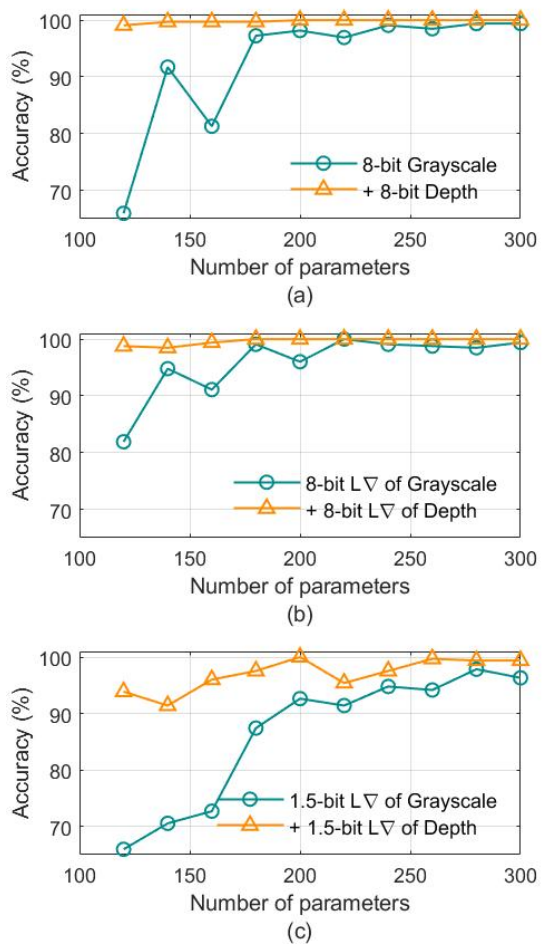

Figure 12. The accuracy with and without the depth information (a) Section 4.2.1 (b) Section 4.2.2 (c) Section 4.2.3.

racy when the models are small. Note that the gap between two curves is narrower than the previous two cases.

4.3. Discussion

All 3 test groups show that the depth information can effectively improve the accuracy, where the improvements are more significant for smaller models, *e.g.* $> 15\%$. As displayed in Figure 13, 8-bit $L\nabla$ yield more accurate models compare with 8-bit raw data, which can be explained by the normalization mentioned in Section 3.1. In view of this as well as the 2D/3D reconstruction, it is highly possible that linear gradients will show great performance in other applications.

With the depth information, the models trained 1.5-bit on $L\nabla$ data have quite high accuracy even with a small number of parameters, *e.g.* 120. In terms of data size, the compression rate for a single pixel is 5.33. For all 2172 images of resolution (96, 96), we need 40 MB for the unprocessed raw data, but only 7.5 MB for 1.5-bit $L\nabla$, which is a significant reduction for edge devices. Besides, the reduction will result in faster inference. (1.5-bit $L\nabla$ are sparse matrices that can be saved more efficiently in memory.) Also note that the dataset is small in this project; for larger datasets, the saving will be considerable.

5. Conclusion & Future Work

This project proposes an effective compression technique for images: quantized linear gradients, which shows excellent performance for both 2D/3D reconstruction and classification. For the 2D reconstruction, the original image can be perfectly reconstructed from the full precision linear gradients; 2.25-bit linear gradients result in less precise reconstruction, but still contain enough information for simple visual tasks like classification. Based on the 2D reconstructions, 3D carving is performed, which can be more accurate than using the original images. Additionally, tiny convolutional neural networks are trained on RGBD images using the same compressing technique, where the data compression rate is about 5.33. For example, 40 MB data can be effectively reduced to 7.5 MB. Experiments show that the models achieve high accuracy in spite of the aggressive compression, where the depth information contributes a lot. A model trained on 1.5-bit linear gradients achieves $> 93\%$ accuracy with only 120 parameters.

In the future, it would be interesting to explore the following points. (i) Improve the 2D reconstruction by removing the stripes/edges of tiles. (ii) Try more 3D construction cases. (iii) Train the CNNs on more challenging datasets, *e.g.* more examples, more classes.

References

[1] Deep double descent. https://openai.com/blog/deep-double-descent/.

[2] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han. Once-for-all: Train one network and specialize it for efficient deployment, 2020.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005.

[4] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[6] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016.

[7] Q. Lu. Random projection in convolutional neural networks: Face recognition and video classification. *Stanford EE270 Project*, 2021.

[8] M. Luber, L. Spinello, and K. O. Arras. People tracking in rgb-d data with on-line boosted target models. In *2011*

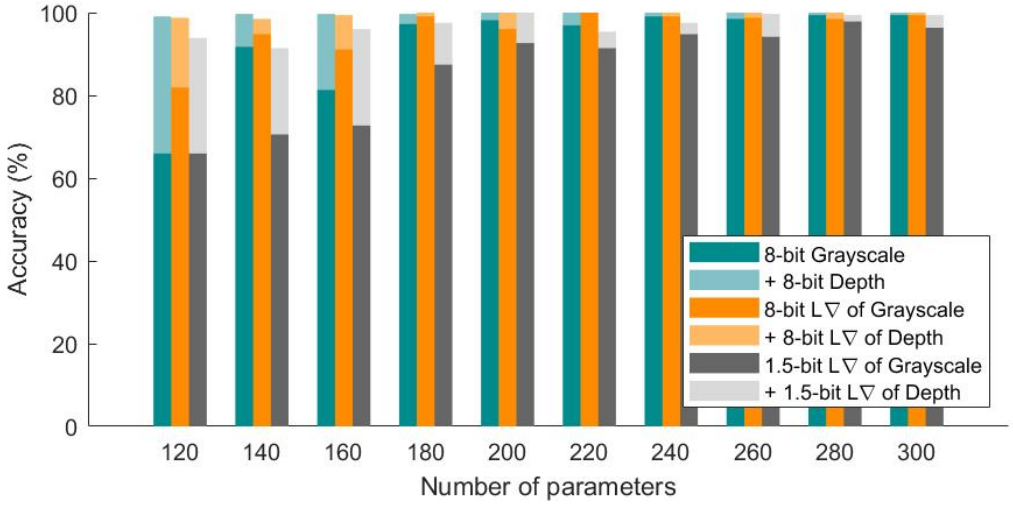

Figure 13. The accuracy with and without the depth information for all 3 test groups.

IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 3844–3849, 2011.

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

[10] C. Premebida, J. Carreira, J. Batista, and U. Nunes. Pedestrian detection combining rgb and dense lidar data. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4112–4117, 2014.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[12] L. Spinello and K. O. Arras. People detection in rgb-d data. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3838–3843, 2011.

[13] L. Spinello and K. O. Arras. People detection in rgb-d data. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3838–3843, 2011.

[14] C. Young, A. Omid-Zohoor, P. Lajevardi, and B. Murmann. A data-compressive 1.5/2.75-bit log-gradient qvga image sensor with multi-scale readout for always-on object detection. *IEEE Journal of Solid-State Circuits*, 54(11):2932–2946, 2019.

This is the final report of CS231A project, March 2021.