CS230

Evaluation of Sparse Image Representations for Visual Recognition

Anthony Li (antli)
Department of Computer Science
Stanford University
antli@stanford.edu

Abstract

Visual recognition tasks such as image classification and localization are some of the most common applications of deep neural networks. However, large quantities of image and video data can impose high costs for storage and transmission, which can limit the application of neural networks in resource-constrained scenarios. While there has been substantial research focus on techniques to train sparse neural network models to reduce computational costs, there has been comparatively less focus on introducing sparsity to the input data representation. I evaluated the effects of sparse image representations on the performance of MobileNetV2 and ResNet18 models trained on an image classification task. The best-performing sparse representation, color-preserved edges, attained 41% classification accuracy on the Tiny ImageNet dataset using MobileNetV2, compared to 48.6% attained with the original images.

1 Introduction

Sparsity is an important and widely used tool for reducing the size and computational cost of deep neural networks. Sparse neural networks, which are those in which a large subset of model parameters are zero, are needed in many applications due to space or inference time restrictions [4]. There exists a substantial body of work studying techniques and challenges relating to training sparse neural networks, often in the context of computer vision tasks [5].

A natural question that follows is whether the application of sparsity can be extended all the way to the input representation. More concretely, for a computer vision task in which a neural network accepts image inputs, is it possible to derive an input image representation in which most input values are zero while preserving acceptable task performance?

This project investigates whether highly sparse image representations, such as edge positions identified by traditional edge detection algorithms, can be sufficient for common computer vision tasks using standard CNN architectures. Emphasis is placed on sparse representations that are relatively inexpensive to compute using hand-coded algorithms and do not require the use of secondary neural networks. A sparse representation that allows for strong performance in visual recognition tasks while being inexpensive enough to compute on low-power edge devices can be valuable in applications that require offloading inference to remote machines. Since sparse image representations also consume less bandwidth to transmit, this can be especially valuable in bandwidth-limited scenarios.

CS230: Deep Learning, Spring 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related Work

Empirically, humans are quite capable of visually comprehending even rough hand-drawn sketches, which are one form of sparse image representation. Additionally, recent work has found biological evidence that complex natural images can be reliably represented by sparsely active neurons in the primary visual cortex [13], which may mean that sparse representations of images are used in human and animal visual systems.

Yu et al. developed Sketch-a-Net [14], a convolutional neural network (CNN) specifically designed to recognize sketches which surpassed human performance on sketch recognition. This suggests that CNN architectures are capable of achieving strong performance working with sparse and diverse input images. Neural networks have also been trained on sketch generation and sketch completion tasks with strong results [7].

George et al. [6] studied localized compression as a technique for altering the input image representation to reduce input size without needing to crop or downscale. This produced a 1-2% higher classification accuracy compared to downscaling, indicating that creative input representations can improve CNN performance.

3 Dataset

Experiments were performed mainly using the Tiny ImageNet dataset [1], which contains 100,000 images of 200 classes downsized to 64x64 colored images. Each class has 500 training images, 50 validation images, and 50 test images, although the test images were not used because their labels are not included in the dataset. The relatively smaller size of Tiny ImageNet compared to the full ImageNet dataset allowed for quicker iteration while training models from scratch, with most models generally converging within about 12 hours of training on a single GPU. All images were upscaled to 224x224 to meet the minimum input size requirement of the models trained.

4 Approach

4.1 Selection of sparse representations

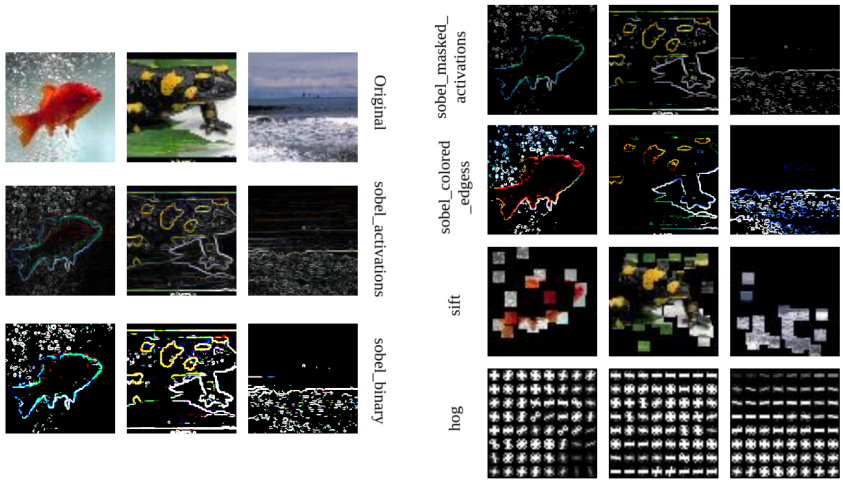

Figure 1: Samples of each sparse representation evaluated

Promising sparse representations were identified primarily by taking inspiration from classical computer vision techniques, including Sobel edge detection [8], SIFT keypoint identification [10], and histogram of oriented gradients (HOG) [3]. Edge detection was an especially promising focus area because it is generally known that deep convolutional neural networks tend to learn edge detectors in their first layers when trained on visual recognition tasks [9]. Edge detection had also been theorized

to play a central role in human visual perception, and therefore has been well studied in classical computer vision [2].

Specifically, the following representations were evaluated:

1. **Sobel edge activations** *(sobel_activations)*

   This representation is produced by passing each image (post-upscaling) through the Sobel filter implementation in Kornia [11], an image processing library that integrates with PyTorch. Default Sobel filter parameters were used. The filter produces an edge intensity map for each of the three color channels. No clipping or other processing was performed on the edge intensity map.

2. **Sobel binary edges** *(sobel_binary)*

   This representation is created starting from *sobel_activations*, but with each edge intensity value >0.25 represented as 1 and each edge intensity value ≤ 0.25 represented as 0. This produces a binary map of edge locations for each channel. While this representation contains significantly less information than *sobel_activations* and was therefore expected to exhibit poorer performance, it enables significantly higher data sparsity and compression since each input value is binary.

3. **Sobel masked edge activations** *(sobel_masked_activations)*

   This representation is a hybrid of *sobel_binary* and *sobel_activations*, created by applying *sobel_binary* as a mask on top of *sobel_activations*. This creates a representation that is similar to *sobel_activations* but is more sparse, since low activations are zeroed.

4. **Sobel color-preserved edges** *(sobel_colored_edges)*

   This representation is produced by first averaging the edge intensities from *sobel_activations* across channels. Pixels in positions where average edge intensity is ≥ 0.2 are preserved, while all other pixels are zeroed. This effectively preserves pixels (with color information) in positions near edges.

5. **Pixels near SIFT keypoints** *(sift)*

   In this representation, the SIFT implementation in OpenCV is used to identify keypoints (pre-upscaling). A mask is then created with value 1 at each keypoint position and 0 elsewhere. The mask is dilated using three iterations of the *binary_dilation* implementation in the *skimage* package, and the mask is applied to the image. This causes pixels near SIFT keypoints to be preserved, and all others to be zeroed.

6. **Histogram of oriented gradients (HOG) visualization** *(hog)*

   This representation is produced by visualizing HOG features produced by *skimage* (pre-upscaling). HOG features were generated using 8x8 pixel cells.

Samples of each representation are shown in Figure 1.

4.2 Model training

A sparse representation can be evaluated by training the same neural network architecture twice on a visual recognition task - first using the original images, and second using the transformed sparse images. Both models should be trained from scratch using identical hyperparameter settings. Of particular interest is the performance delta between these two models - an ideal sparse representation should result in a model that is similar in performance to the model trained on the original images, while also demonstrating competitive absolute performance.

To help facilitate better generalization of results across different model architectures, each sparse representation was evaluated using two well-known architectures: MobileNetV2 and ResNet18. Both architectures have official PyTorch implementations, which were used with default hyperparameters and without taking advantage of any pre-trained weights.

For each model architecture, the final fully-connected layer was replaced to output 200 values representing the 200 Tiny ImageNet classes, instead of 1000 values for the standard ImageNet classes. No other architectural changes were made aside from replacing the final classification layer. Models were trained using the PyTorch *NLLLoss* loss function and evaluated based on Top-1 and Top-5

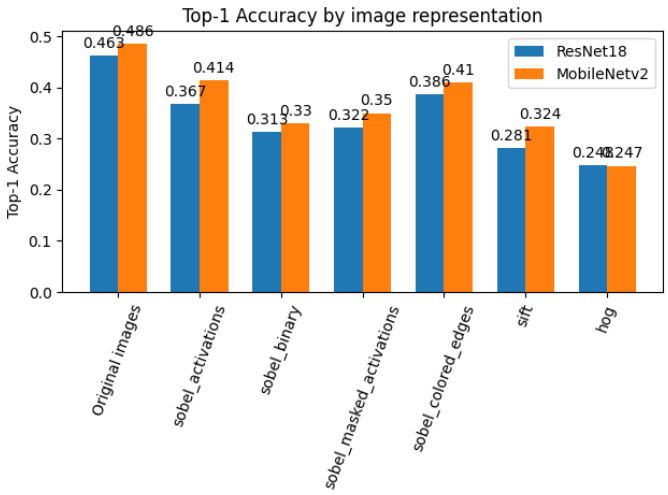

Figure 2: Maximum Top-1 Validation Accuracy attained by each model trained

classification accuracy. Classifier implementation was adapted from [12] and migrated to PyTorch Lightning.

Due to the small size of the Tiny ImageNet dataset, both MobileNetV2 and ResNet18 were quite prone to overfitting. Therefore, evaluation was focused on maximum validation accuracy attained during training (simulating early stopping), rather than the validation accuracy attained at the end of training.

5 Results and Discussion

Figures 2 and 3 show the maximum top-1 and top-5 validation accuracy that each model attained during training. None of the sparse representations were able to match the performance of the models trained on the original images, which was not unexpected. The edge activations (*sobel_activations*) and color-preserved edges (*sobel_colored_edges*) performed the best, both achieving top-1 accuracy of about 41% with MobileNetV2 compared to 48.6% on the original images. The *sobel_colored_edges* also performed the best based on top-5 accuracy, achieving 68.7% with MobileNetV2 compared to 74.9% on the original images.

Between ResNet18 and MobileNetV2, MobileNetV2 generally performed slightly better across the board. However, it is interesting to observe that relative performance between different sparse representations is nearly identical for both architectures, providing evidence that the results seen may be reasonably generalizable across architectures.

Additionally, the performance gap between models trained on the original images and models trained on sparse representations is considerably smaller when evaluated based on top-5 accuracy instead of top-1 accuracy. This indicates that for more challenging tasks, it may be more difficult to find a sparse representation that still allows high model performance.

It is also worth noting that the *sift* and *hog* representations performed more poorly than all of the representations based on edge detection. I found it somewhat surprising that *sift* performed poorly, given that on many images there are enough keypoints to reveal a significant swath of the image making it less sparse than most of the other representations. This may be because SIFT keypoints are better suited for feature matching than object identification, so many of the keypoints may not be on the object of interest and therefore important areas of the image could be missing.

It was also quite difficult to generate reasonable HOG visualizations for such small (64x64) images, given that each image needs to be divided into smaller patches (cells) to create a visualization for each cell. It would be interesting to see whether HOG visualizations can perform better with larger input images.

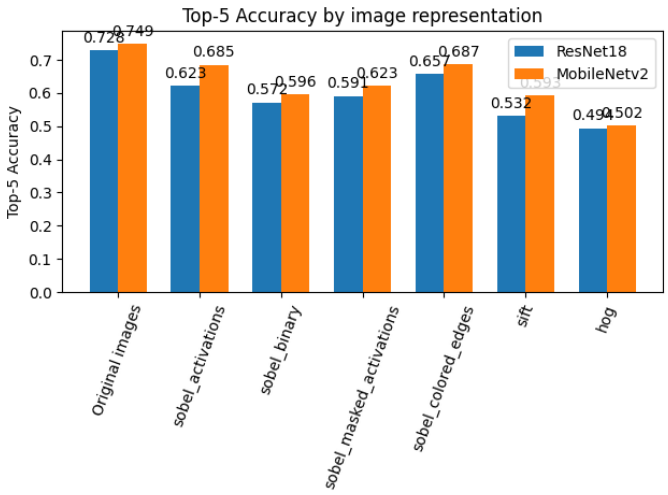

Figure 3: Maximum Top-5 Validation Accuracy attained by each model trained

6 Conclusion

Overall, it appears to be quite difficult to create a sparse representation that results in no reduction to image classification task performance. However, if some performance reduction is acceptable in exchange for potential storage, bandwidth, and compute savings from working with sparse images, the results indicate that sparse representations based on edges (and particularly edges with color information preserved) can work well and are worth exploring further. With additional tuning of thresholds and edge detector settings, it may be possible to further reduce the performance gap between edge-based sparse representations and original image representations for visual recognition tasks.

7 Limitations and Future Work

Due to the time and compute cost of training on the entire ImageNet datset, models were trained and evaluated only on Tiny ImageNet. An important next step would be to validate the performance of the best sparse representations on the full ImageNet dataset, to see whether they would still perform well on a more difficult task, with more data and images larger than 64x64. Additional model architectures can also be tested to see whether sparse representations that are effective for MobileNetV2 and ResNet18 would also be effective for other architectures. Finally, future work can also evaluate whether sparse representations that work well for image classification would also work well for related tasks like segmentation and localization.

8 Contributions

Anthony Li worked on this project independently.

References

[1] Tiny ImageNet.

[2] Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 200(1140):269–294, February 1978.

[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, San Diego, CA, USA, 2005. IEEE.

[4] Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners, 2020.

[5] Utku Evci, Fabian Pedregosa, Aidan Gomez, and Erich Elsen. The difficulty of training sparse neural networks, 2020.

[6] Christopher A. George and Bradley M. West. Localized compression: Applying convolutional neural networks to compressed images, 2019.

[7] David Ha and Douglas Eck. A neural representation of sketch drawings, 2017.

[8] N. Kanopoulos, N. Vasanthavada, and R. L. Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of Solid-State Circuits*, 23(2):358–367, 1988.

[9] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[10] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[11] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch, 2019.

[12] Alexander Wu. A Beginner's Tutorial on Building an AI Image Classifier using PyTorch, July 2019.

[13] Takashi Yoshida and Kenichi Ohki. Natural images are reliably represented by sparse and variable populations of neurons in visual cortex. *Nature Communications*, 11(1), February 2020.

[14] Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Sketch-a-net that beats humans, 2015.