Anycost GAN Optimization

Mahmoud Khalifa, Anthony Ou
mahmoudk@mit.edu, aou@mit.edu

Abstract

Efficent inference of machine learning models have become relevant in the past year because of massive improvements in the quality of large models. The Anycost GAN is a model that has been optimized for practical use cases. In particular, it features a capability to handle a different computational cost loads, and input image qualities. However, we believe that the Anycost GAN can be made even more efficient with the ideas that we have learned in class. In this work, we will apply ideas of magnitude based weight pruning and parameter quantization to this model in an effort to further improve performance. While not all of our ideas were successful, we have shown through our exploration that it is possible to significantly improve the model performance with minimal loss in generated image quality.

1 Introduction

With recent developments in generative models, there has been a renewed interest in applications of machine learning models. However, many generative models such as Large Language models or diffusion models are too large and run at speeds too slow to be of use on edge devices. As such, there has been much interest in both reducing number of parameters and decreasing MACs. One such model is the Anycost GAN, a GAN made efficient for the purpose of quick generation while still maintaining the quality of the generated images (1).

The Anycost GAN is capable of generating images at a range of computational costs while producing visually consistent outputs. These varied costs include different image resolutions and generation times. It achieves this by using a version of neural architecture search. The model is trained so that it is flexible to channel reduction, as such it is capable of producing low-cost sub-generators by taking a subset of the trained weights without additional fine-tuning. While the Anycost GAN is very powerful for its consistency across different computational costs, it is not yet optimal. We believe that we can achieve more efficient performance of this model through the application of techniques that we have learned in this class. Namely we applied techniques of fine grained pruning, channel pruning, K-means quantization, and linear quantization to further optimize the Anycost GAN.

2 Related Works

2.1 Pruning

Considerable work has been done in structured weight-pruning, or the systematic removal of a subset of a model's weights to improve performance (2). Recent examples include applications to Large-Language models (3) and deep convolutional neural networks (4). As described in lectures, fine-grained pruning or magnitude based pruning has shown to be capable of drastically reducing model sizes with minimal loss of accuracy (5). It has specifically been applied to both vision networks (6), and language networks (7) and is central to the lottery ticket hypothesis (8). There have been many other efforts to both introduce and exploit sparsity in model parameters (9). Channel pruning is capable of giving drastic model speed-ups in addition to drastically reducing model size (10) (11). Due to its ability to reduce inference costs, channel pruning has been widely used to prepare models

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

for operation on edge devices (12). While these methods are generally coupled with fine-tuning to recover lost performance, it is possible to have high performance without fine-tuning (13).

2.2 Quantization

K-means quantization was first explored as Huffman coding of model parameters and has been shown to drastically improve model speed and reduce model size (14). A variation of this method has been used in many use cases including image compression (15) and weather forecasting (16). These methods have since been built upon to develop more sophisticated methods of quantization (17). Linear quantization casts all floating point arithmetic to integer arithmetic to drastically speed-up inference (18). These methods have been expanded to also make training more efficient and includes other efforts to (19). Other variations of this method have shown improvements in accuracy on edge devices (20). These methods have recently been specifically applied to GANs for improved performance in edge devices (21) and for improved handling of compressed images (22).

3 Pruning

In optimizing the Anycost GANs model, two primary techniques were employed: Fine Grained Pruning and Channel Pruning. These techniques aimed to reduce the model size and increase inference speed, respectively. The outcomes of these optimizations are discussed in the following subsections.

3.1 Fine Grained Pruning

Fine Grained Pruning, particularly Magnitude Based Fine Pruning, was employed to reduce the size of the neural network model while minimizing the impact on performance. This pruning technique targets individual weights in the network, removing those with the smallest absolute values under the assumption that they contribute less to the network's output.

The pruning process can be mathematically described as follows. Let W represent the weight matrix in a neural network layer. We define a pruning threshold θ, typically based on the desired pruning percentage p. The pruned weight matrix W' is then derived by setting weights below this threshold to zero. The expression for this operation is:

$$W'_{ij} = \left\{ \begin{array}{ll} 0, & \textit{if } |W_{ij}| \leq \theta \\ W_{ij}, & \textit{otherwise} \end{array} \right\}$$

In this process, W'_{ij} denotes the weight in the pruned matrix W', and W_{ij} is the corresponding weight in the original matrix W. The threshold θ is selected such that a predetermined percentage p of the smallest absolute weights in W are eliminated.

Our approach first involved examining the distribution of parameters across layers, as illustrated in Figure 8. We observed a concentration of parameters in the Convolutional Layers, leading us to focus our fine-grained pruning efforts there. The distribution of weights prior to pruning resembled a normal distribution (Figure 6). After applying Magnitude Based Fine Pruning, the resulting weight distribution is depicted in Figure 7.

Implementing Magnitude Based Fine Pruning resulted in a 20% reduction in the overall size of the model. We didn't see any meaningful change in inference speed look Figure 9. This significant reduction was achieved without noticeable degradation in the quality of the generated images, as evidenced in Figure 1. Attempts to further prune the network at 30% and 40% pruning ratios are shown in Figures 2 and 3, respectively. Figure 2 indicates a slight loss in image quality, suggesting that even a minor reduction in model size can adversely affect our specific task of generating faces. A similar trend is observed in Figure 3, where excessive pruning leads to poor results.

This experiment portrays the efficacy of Magnitude Based Fine Pruning in reducing model size while preserving output quality. However, it also highlights the need for careful calibration of the pruning process, as excessive pruning can lead to a noticeable decline in performance.

3.2 Channel Pruning

Channel Pruning focuses on accelerating the inference process of the neural network. This method involves reducing the number of channels in the convolutional layers, thereby decreasing the computational load during inference and leading to faster processing times.

In the context of AnyCost GANs, which are designed to be adaptable across various computational constraints, applying Channel Pruning requires careful consideration. These models often have built-in mechanisms to adjust their complexity based on available resources, and Channel Pruning could potentially conflict with these dynamic adjustments.

Despite these considerations, we implemented Channel Pruning in our AnyCost GAN model. The process involved systematically reducing the number of channels in selected convolutional layers, with the goal of finding a balance between efficiency and performance.

The implementation of Channel Pruning resulted in a 18% speedup in inference time when we tested out different images with different settings and averaged the times in Figure 9. However, this increase in efficiency came at the cost of image quality. The observed severe decrease in image quality, as shown in Figure 4, highlights the trade-off inherent in Channel Pruning: while it enhances computational efficiency, it can also lead to a significant compromise in model performance if not carefully managed.

In our experiments, we observed that the reduction of channels led to a noticeable decline in the fidelity of the generated images. This degradation is particularly evident in complex image regions where higher representational capacity is required. It suggests that while Channel Pruning can be effective for reducing computational requirements, it must be applied judiciously, especially in generative models where image quality is paramount.

Moving forward, it is essential to consider adaptive pruning strategies that align with the dynamic nature of AnyCost GANs. This might include developing methods that can intelligently prune channels based on the specific requirements of the generated image or the available computational resources at inference time.

4 Quantization

4.1 K-Means Quantization

In an effort to optimize the AnyCost GAN, we employed K-means quantization, a technique aimed at reducing the model size and potentially speeding up inference by quantizing the model's weights. The quantization process was conducted with varying bitwidths of 2, 4, and 8, setting the number of clusters k to $2^{bitwidth}$.

Mathematically, K-means quantization can be described as follows. For a given set of weights W in the neural network, the K-means algorithm partitions W into k clusters. Each weight $w \in W$ is then replaced by the nearest centroid of the cluster it belongs to. The quantized weights W' are given by:

$$W' = \{c_i \mid c_i = \textit{argmin}_{c \in C} \|w - c\|^2,\ w \in W\}$$

where C is the set of centroids for the k clusters.

Our experiments revealed that for bitwidths of 2 and 4, the image quality deteriorated significantly, as evidenced by the results in Figure 5. This quality degradation is attributed to the coarse quantization, which leads to a loss of detail in the weight representation.

Furthermore, we observed a substantial increase in the initial setup time for the quantized models. The setup times were approximately one minute for a bitwidth of 2, five minutes for a bitwidth of 4, and an hour for a bitwidth of 8. This suggests that the complexity of the K-means clustering process increases with the number of clusters, impacting the initial preparation phase of the model.

Despite these drawbacks in initial setup time and image quality at lower bitwidths, the quantization process did not affect the operational efficiency during inference. Specifically, the time required for manipulating generated images (such as changing facial expressions) remained constant across all tested bitwidths.

These findings highlight the delicate balance in K-means quantization applied to AnyCost GANs, emphasizing the critical role of bitwidth selection. While lower bitwidths offer model size reduction, they come at the cost of significant image quality degradation and increased setup time, making them less viable for high-quality image generation and time-sensitive applications.

4.2 Linear Quantization

Based on what we learned in this class, we expected that implementing linear quantization into this model would result in drastic improvements in performance as linear quantization is independent from pruning and subnetwork search AnyCost performs. However, the implentation of AnyCost that we were using does not allow for easy integration of well used linear quantization packages and so we were forced to implement it manually. Given the prevalent usage of floating point noise, weight normalization, and adaptive channel inference it was infeasible to implement linear quantization end-to-end. Instead, we opted to quantize the convolution, as per figure 8 the convolutions are the computationally heaviest portion of this model. As such, for every convolution module, we quantized the input q, the weights w and the bias b then performed the convolution as integers to produce a quantized output y.

$$x = S_x(q_x - Z_x)$$
$$w = S_w(q_w - Z_w)$$
$$o = S_o(q_o - Z_o)$$
$$b = S_b(q_b - Z_b)$$

Where S and Z are the scale and zero-point respectively and q is the integer representation. Due to how the model applied scaling and normalization after selecting with its channel ratios, we decided not to implement per-channel quantization and instead used a single scale and zero-point for each convolution module. We used a global bitwidth of 8 bits to select our scale and zero-point, and used 8-bit integers to represent q. The model was run normally once and the inputs and outputs for each module were used to calibrate the selection of the scales and zero-points. As we covered in Lab 2, the result for the output is:

$$q_o = (\text{CONV}[q_x, q_w] + q_b - \text{CONV}[Z_x, q_w]) \cdot (S_x S_w / S_o) + Z_o$$

Since all operations are now integer operations, we expected to see considerable speed-ups from this implementation. However, to the contrary we experienced very large decreases in performance; we determined that the reason for these slow-downs is that the PyTorch implementation for convolutions is far slower on integers than on floats. This is an issue that does not seem to have much support for. Given that the entirety of the code base is in PyTorch, we decided that it is infeasible to continue down this line of investigation as to do so would require a drastic overhaul of the codebase and all alternatives we explored seemed to either contain errors or be not applicable to our use case.

5 Fine Tuning

In our study, we have employed techniques such as Fine Grained Pruning and K-means Quantization to optimize the AnyCost GAN. Future work in this research is to explore the effects of fine-tuning on these optimized models. Fine-tuning, a process where a pre-trained model is further trained on a specific dataset or task, can lead to notable improvements in performance, especially in models that have undergone significant structural changes.

We anticipate that fine-tuning the pruned and quantized models could result in considerable enhancements. The idea is that fine-tuning allows the model to adjust its parameters more effectively to the dataset's specific features. This adjustment could help in offsetting performance losses incurred due to pruning or quantization. It might even make it feasible to prune the models further or to use lower bitwidths in quantization, thereby increasing the model's speed without a substantial compromise in image quality.

However, our ability to pursue fine-tuning is currently constrained by two significant factors:

1. Large Dataset Size: Our models were originally trained on the Flickr-Faces-HQ Dataset (FFHQ), which contains 70,000 high-quality images. The sheer size of this dataset presents a challenge for fine-tuning, especially considering the computational resources required to process such a large amount of data effectively.

2. Lack of GPU Resources: Another critical limitation is the absence of adequate GPU resources. Fine-tuning, particularly on large datasets like FFHQ, demands substantial computational power, typically provided by GPUs. Without access to such resources, it's impractical to undertake the fine-tuning process, which is essential to realize the full potential of the optimized models.

6 Conclusion

This study explored enhancing the Anycost GAN's efficiency, focusing on fine-grained pruning, channel pruning, and K-means quantization, alongside an attempt at linear quantization. Our results illuminated the complex trade-offs in model optimization: while fine-grained pruning effectively reduced size with minimal quality loss, channel pruning highlighted the balance between speed and fidelity. K-means quantization's impact on image quality was dependent on bitwidth selection, and our linear quantization efforts revealed integration challenges in complex models. Although constrained by resource limitations and unable to explore fine-tuning, our findings offer insights into the balance between model efficiency, size, and output quality. This research contributes to the further optimization of Anycost GAN and hopefully will help run it on smaller devices.

Figures

Figure 1: 20% Pruned Model

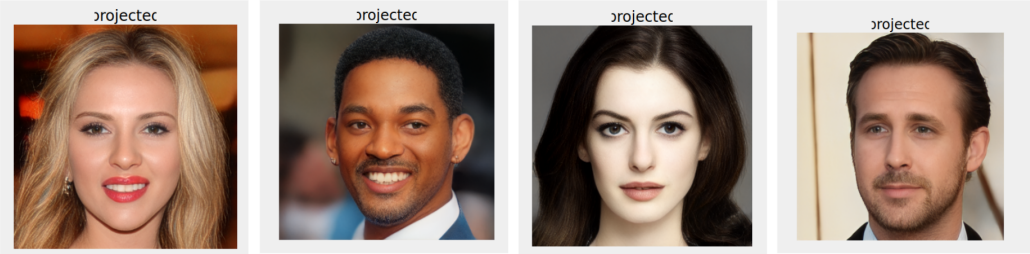

Figure 2: 30% Pruned Model

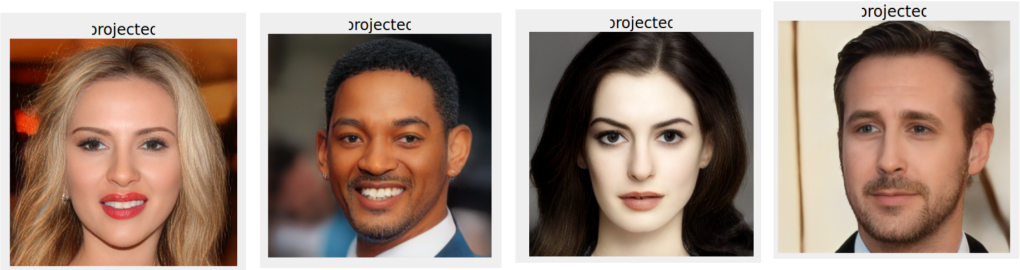

Figure 3: 40% Pruned Model

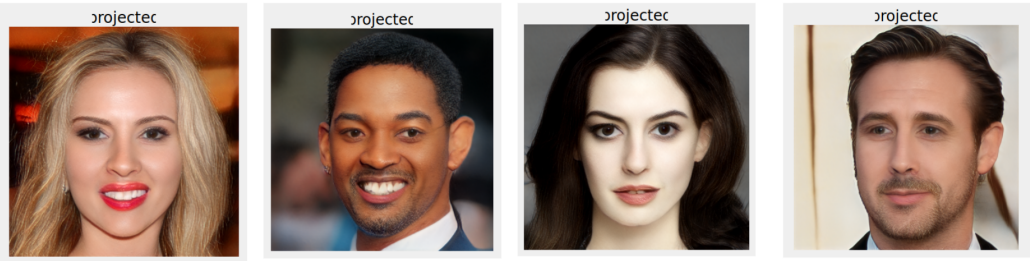

Figure 4: Channel Pruned Model

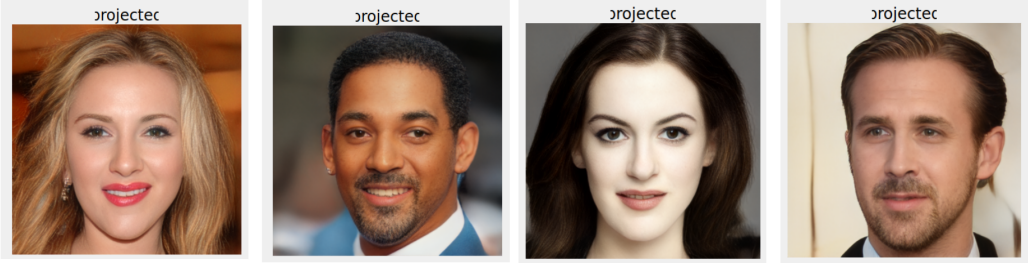

Figure 5: K-Means Quantized Model

Figure 6: Layer Weights

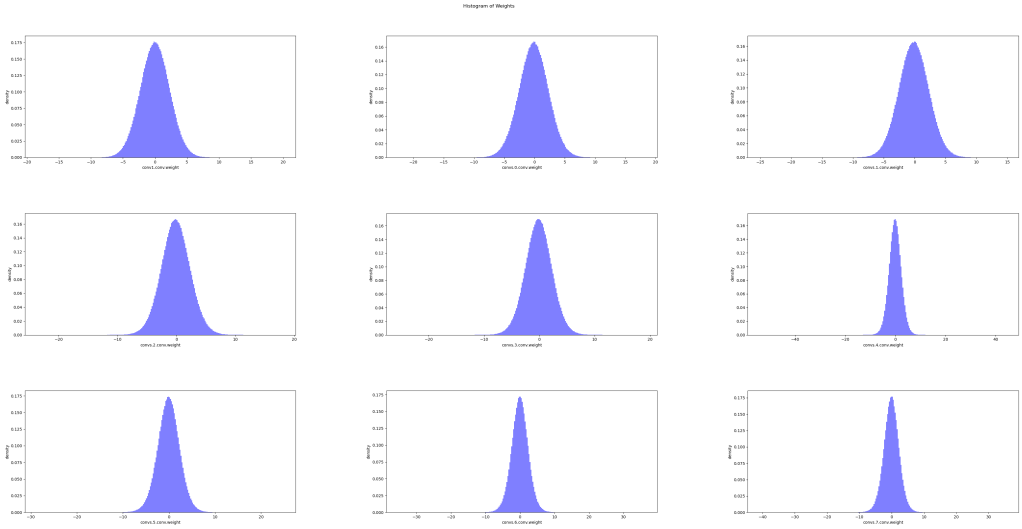

Figure 7: Layer Weights

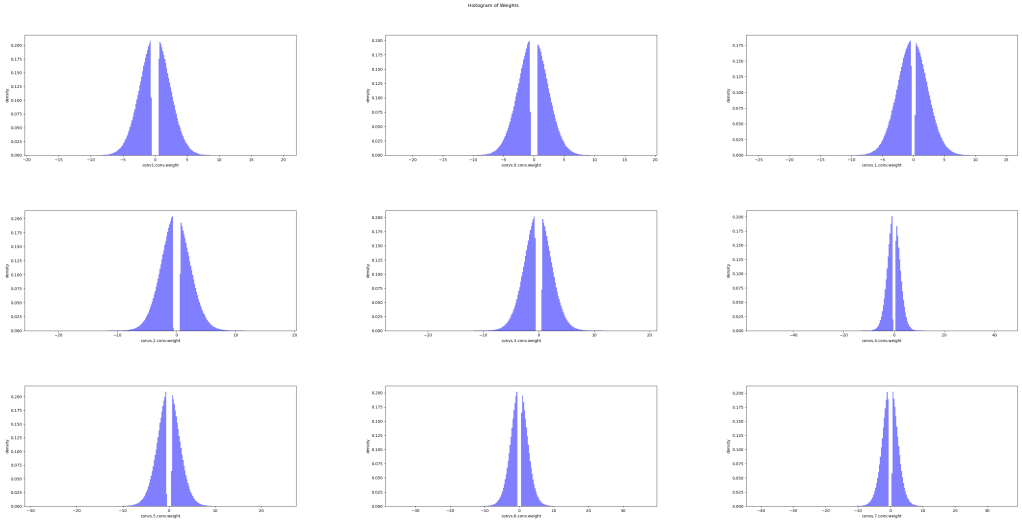

Figure 8: Layers Number of Parameters

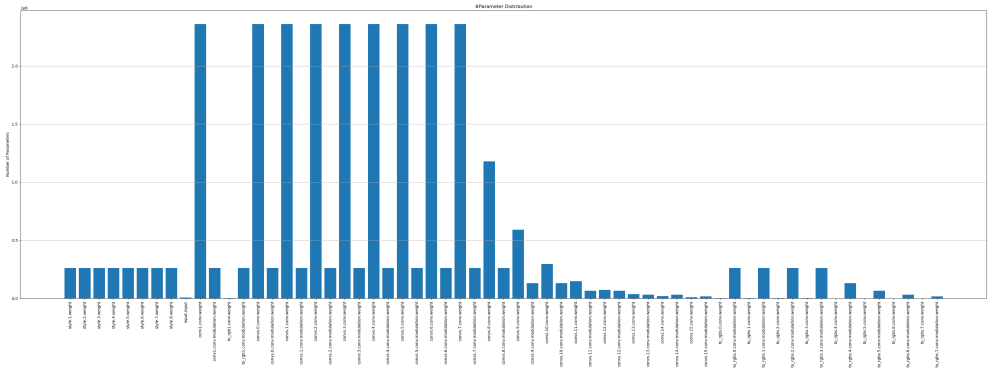

Figure 9: Average seconds to generate image

Image	Original	Fine Grained	Channel Pruning
Ryan	0.7s	0.72s	0.57s
Anne	0.7s	0.67s	0.58s
Will	0.68s	0.69s	0.56s
Scarlett	0.69s	0.67s	0.54s
Average	0.6925s	0.6875s	0.5625s

References

[1] J. Lin, R. Zhang, F. Ganz, S. Han, and J.-Y. Zhu, "Anycost gans for interactive image synthesis and editing," *arXiv preprint arXiv:2103.03243*, 2021.

[2] G. Fang, X. Ma, M. Song, M. B. Mi, and X. Wang, "Depgraph: Towards any structural pruning," 2023.

[3] M. Xia, T. Gao, Z. Zeng, and D. Chen, "Sheared llama: Accelerating language model pre-training via structured pruning," 2023.

[4] Y. He and L. Xiao, "Structured pruning for deep convolutional neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–20, 2023. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2023.3334614

[5] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," 2015.

[6] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," 2016.

[7] T. Gale, E. Elsen, and S. Hooker, "The state of sparsity in deep neural networks," 2019.

[8] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "Linear mode connectivity and the lottery ticket hypothesis," 2020.

[9] T. Hoefler, D. Alistarh, T. Ben-Nun, N. Dryden, and A. Peste, "Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks," 2021.

[10] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," 2017.

[11] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," 2019.

[12] L. Xue, Z. Wang, X. Wang, and G. Li, "The road to on-board change detection: A lightweight patch-level change detection network via exploring the potential of pruning and pooling," 2023.

[13] S. Bai, J. Chen, X. Shen, Y. Qian, and Y. Liu, "Unified data-free compression: Pruning and quantization without fine-tuning," 2023.

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

[15] C. Reich, B. Debnath, D. Patel, and S. Chakradhar, "Differentiable jpeg: The devil is in the details," 2023.

[16] C. Chen, T. Zhou, Y. Zhao, H. Liu, L. Sun, and R. Jin, "Svq: Sparse vector quantization for spatiotemporal forecasting," 2023.

[17] P. Stock, A. Joulin, R. Gribonval, B. Graham, and H. Jégou, "And the bit goes down: Revisiting the quantization of neural networks," 2020.

[18] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017.

[19] Y. Yang, X. Dai, J. Wang, P. Zhang, and H. Zhang, "Efficient quantization strategies for latent diffusion models," 2023.

[20] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2017.

[21] P. Wang, D. Wang, Y. Ji, X. Xie, H. Song, X. Liu, Y. Lyu, and Y. Xie, "Qgan: Quantized generative adversarial networks," 2019.

[22] Q. Mao, T. Yang, Y. Zhang, S. Pan, M. Wang, S. Wang, and S. Ma, "Extreme image compression using fine-tuned vqgan models," 2023.