OPT2025: 17th Annual Workshop on Optimization for Machine Learning 1–15

FineAMP: Optimization-Based Automatic Mixed Precision Quantization for Efficient Diffusion Model Inference

Burak Bartan [*] BBARTAN@ALUMNI.STANFORD.EDU
Ruizhong Qiu [†] RQ5@ILLINOIS.EDU
Rafael Esteves RESTEVES@QTI.QUALCOMM.COM
Yuwei Ren REN@QTI.QUALCOMM.COM
Weiliang Will Zeng WZENG@QTI.QUALCOMM.COM
An Chen ANC@QTI.QUALCOMM.COM
Qualcomm AI Research [‡]

Abstract

Quantization of weights and activations is crucial in ensuring efficient inference for large machine learning models. Different operations within a machine learning model vary in their sensitivity to quantization; some introduce significantly more error than others. Mixed precision quantization methods aim to leverage this observation to improve model efficiency without causing a significant drop in the model's performance. We propose an optimization-based method, called FineAMP, that automatically determines bit-width assignments for individual operations. We show that fine-grained, per-operation bit-width assignments result in lower on-device latency compared to coarser, uniform bit-width strategies.

1. Introduction

Diffusion models have achieved remarkable success in generating images with both high fidelity and diversity (Ho et al. (2020); Song et al. (2020); Rombach et al. (2022)). Running diffusion models on mobile devices is challenging due to intensive computational and memory demands. Quantization offers a promising solution by reducing model size, latency, and power consumption, making on-device deployment more feasible.

Mixed precision quantization provides flexibility to select different bit-widths for different operations or layers, depending on the granularity that the target hardware allows. We propose a fine-grained mixed precision algorithm that assigns bit-widths for each operation. We call this algorithm FineAMP, short for fine-grained automatic mixed precision. Bit-width selection for mixed precision is a combinatorial problem since each operation's bit-width is to be selected from a discrete set of options. Because modern machine learning models such as SDXL (Podell et al. (2024)) have a total number of operations that scale to thousands, finding the optimal set of bit-widths for all the operations (that satisfy a given accuracy requirement) is computationally intractable. Remarkably, we demonstrate that with only a small number of accuracy evaluations (on the order of tens), it is possible to estimate the quantization sensitivities of thousands of operations.

We list the main contributions of this paper as follows.

[*] Work done while employed at Qualcomm AI Research.
[†] Work done during internship at Qualcomm AI Research.
[‡] Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

- **Scalable mixed precision:** We have formulated a quantization sensitivity estimation algorithm to efficiently generate fine-grained mixed precision profiles.
- **Better compute and accuracy trade-off:** We have shown through numerical simulations that our method achieves a better trade-off between computational cost and accuracy compared to previous operation-type based coarse mixed precision methods.
- **Reduced latency on device:** We have verified that our algorithm can generate mixed precision profiles that achieve significant latency reductions on a mobile platform.
- **Conversion cost control:** We have proposed an integer linear programming (ILP) based conversion cost control algorithm that leads to additional reductions in latency by modifying mixed precision profiles to reduce the bit-width conversion cost (see appendix).

2. Scalable mixed precision

The high number of operations in a large model makes it computationally intractable to measure the quantization sensitivities of each operation. Conducting sensitivity analysis for each quantized operation in the Unet in the SDXL model (Podell et al. (2024)) could take several days to complete, even when using an A100 GPU with 80 GB of memory and just 10 prompts. Our method shortens the measurement time by estimating fine-grained sensitivities from aggregate sensitivities through the optimization problem that we will introduce in this section.

The proposed algorithm, FineAMP, consists of 3 stages. In the first stage, we collect aggregate SQNR values for groups of operations. More precisely, given a group of operations, we disable quantization for all operations except for the ones in the group, and evaluate the model output. This output is the noisy model output $\widetilde{\boldsymbol{y}}$. We also evaluate the full precision model output $\boldsymbol{y}$. Then we compute the SQNR using $\text{SQNR} := 10\log_{10}\frac{\|\boldsymbol{y}\|_2^2}{\|\widetilde{\boldsymbol{y}}-\boldsymbol{y}\|_2^2}$. This aggregate SQNR computation tells us how sensitive the operations in the given group are to quantization when they are simultaneously quantized. In the second stage of FineAMP, we solve an optimization problem to estimate the individual (per-operation) SQNR values from the aggregate SQNR values collected in the first stage. The final stage simply assigns bit-widths to individual operations based on the estimated SQNRs.

Aggregate sensitivity evaluations. The key idea behind collecting aggregate sensitivity evaluations is to group operations that may exhibit similar quantization sensitivities. For the Unet model in SDXL, we construct the groups (i) based on the operation types such as softmax, convolution, matmul, and (ii) based on the blocks: down_blocks_1, down_blocks_2, up_blocks_1. One could devise other ways of constructing groups such as random sampling. However, randomly constructed groups may combine operations with high and low quantization sensitivity. This could make it harder for the optimization problem to identify the individual error contributions from each operation.

Assuming we construct d groups, we need to run $d \times p$ forward passes (assuming batch size is 1) to collect the aggregate SQNR values, where p is the number of prompts. For the Unet model in SDXL, the number of operations (excluding the data movement operations

such as transpose, reshape, permute, etc) is approximately 2000 while we construct $d = 40$ groups based on the above recipe. The computational cost associated with the collection of SQNR values is reduced by 98%.

Optimization problem for estimating fine-grained sensitivities. The second stage decomposes the aggregate SQNRs into individual per-operation SQNRs using an optimization problem that we have formulated and will describe in this subsection. The input is the aggregate SQNRs for each group of operations and the output is an estimated SQNR for each operation.

We first state the optimization problem formulation and then describe the variables and constraints:

$$\min_{z \in \mathbb{R}^N} \|z\|_1 = \sum_{i \in [N]} z_i \tag{1}$$

$$\text{s.t. } 10^{-\frac{U}{\tau}} \leq z_i \leq 1, \forall i \in [N] \tag{2}$$

$$10^{-\epsilon} 10^{-\frac{R_{b/o,j}}{\tau}} \leq \sum_{i \in \mathcal{S}_{b/o,j}} z_i \leq 10^{\epsilon} 10^{-\frac{R_{b/o,j}}{\tau}}, \forall j \in [d] \tag{3}$$

$$10^{-L} z_{t_{j,k}} \leq z_{t_{j,k+1}} \leq 10^{L} z_{t_{j,k}}, \forall j \in [n_o], \forall k \in [|T_j| - 1] \tag{4}$$

$$z_i = z_{i'}, \forall i, i' \in \mathcal{S}_{b,j} \cap \mathcal{S}_{o,j'}, \forall j \in [n_b], \forall k \in [n_o] \tag{5}$$

where $z_i \in \mathbb{R}$ is the decision variable corresponding to the quantization noise for operation i (i.e., $z_i = 10^{-\frac{r_i}{\tau}}$, where r_i is the SQNR when quantizing only operation i, and τ is a hyperparameter). Above, we assume that the total number of operations is N, and $n_{b/o}$ is the number of blocks or operation types. $\mathcal{S}_{b/o,j}$ denotes the set of operations of block or operation type j while $R_{b/o,j}$ is the corresponding aggregate SQNR. ϵ is a tolerance parameter ensuring solution feasibility.

We have the ℓ_1-norm of the noise vector z in the objective in (1), which promotes sparsity among the noise terms. This is to enforce the assumption that many operations are not sensitive to quantization. Since quantization noise for those operations is small, we can leverage the sparsity and put emphasis on the potentially more sensitive operations. The constraints in (2) define the range for the variables z_i. The constraints in (3) enforce the relationship between the individual per-operation SQNRs and the aggregate SQNRs, which has first been derived by Lin et al. (2016):

$$\sum_{i \in \mathcal{S}_{b/o,j}} 10^{-\frac{r_i}{\tau}} \approx 10^{-\frac{R_{b/o,j}}{\tau}}. \tag{6}$$

The constraints in (4) enforce that adjacent occurrences of the same operation type exhibit similar sensitivity to quantization. The strength of this constraint is controlled by a hyperparameter L. We find that a good value for the parameter is $L = 0.4$ and we set $L = 0.4$ across all experiments. The constraints in (5) reduces the search space of the optimization problem by leveraging that the operations of the same type and block are indistinguishable from the quantization sensitivity perspective. We solve the linear program (1)–(5) using CVXPY (Diamond and Boyd (2016); Agrawal et al. (2018)) and the CBC solver (CBC (2025)). Once we solve the optimization problem for the noise vector z, we map it into SQNR using the element-wise formula $-\tau \log_{10}(z)$.

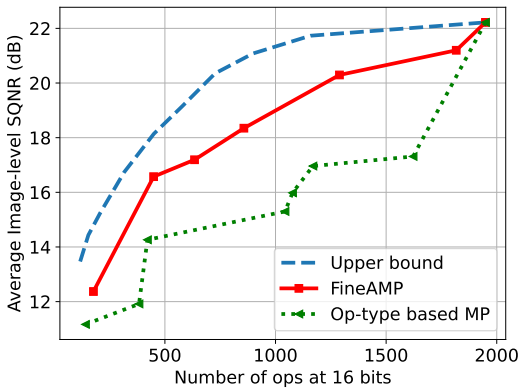


Figure 1: Average image-level SQNR (across 10 prompts) in dB as a function of number of operations at 16 bits for the Unet of the SDXL model. The right-most point shows when all operations are quantized to W8A16.

Bit-width assignments. Stage 2 returns a list of estimated SQNR values for each operation by solving the problem in (1)-(5). Since we have considered two bit-width options (8 bits and 16 bits) in our experiments, the assignment of bit-widths given the sensitivity list can be done in a straightforward way as follows. We set a threshold S_{thr} and assign 16 bits for operations with estimated SQNR below the threshold, and 8 bits for the remaining operations. We vary the threshold to obtain accuracy-complexity trade-off curves for FineAMP, which are shown in Section 3.

3. Experiments

Experimental setup. We use AIMET (Siddegowda et al. (2022); AIMET (2025)) to simulate the accuracy of mixed precision profiles. We leverage the model export capability of AIMET to generate files for on-device execution of models. We evaluate the quality of the generated images under different quantization schemes using the metrics SQNR, PSNR (peak signal-to-noise ratio), LPIPS (Zhang et al. (2018)), and HPSv2 (Wu et al. (2023)). We apply FineAMP to SDXL with 20 diffusion steps and LCM SSB-1B (Luo et al. (2023)) with 4 diffusion steps.

We keep the text encoder and VAE components of the models at floating point precision while we execute Unet using fixed-point mixed precision (INT8, INT16). All weights are quantized to INT8; mixed precision is applied only to activations. Unet is the most compute intensive component and needs to be run multiple times.

Compute-accuracy trade-off. Figure 1 shows the average image-level SQNR as the performance metric on the vertical axis and the number of operations at 16 bits on the horizontal axis. The number of operations assigned to 16 bits can be thought of as a proxy to the latency of the resulting mixed precision profile. The red curve with square markers shows the accuracy for different mixed precision profiles obtained by setting different thresholds in FineAMP. The blue dashed curve serves as an upper bound and is obtained by measuring

the individual operation sensitivities (the evaluation of individual operation sensitivities takes a few days to complete and is only done to compute the upper bound). The green dotted line is the baseline where the bit-widths are determined based on the aggregate SQNRs of operation types. This corresponds to what is commonly done in practice. By moving to a finer-grained bit-width assignment scheme, we obtain a significantly better curve than the green dotted curve.

Figure 1 demonstrates that for a given budget for the number of 16 bit operations, assigning bit-widths on the operation level (i.e. fine-grained) rather than operation-type level (i.e. coarser) leads to a significantly higher accuracy. Another way to interpret this result is that for a given accuracy requirement, FineAMP can find a mixed precision profile with more operations quantized to 8 bits.

Accuracy simulations. Table 1 shows the accuracy evaluations for different mixed precision quantization schemes as well as the baselines of FP32, W8A8 and W8A16. The third column shows the percentage of operations that run at 16 bits. Table 1 confirms that different metric evaluations are in consistency with the SQNR evaluations in Figure 1. FineAMP enables us to find fine-grained bit-width assignments with improving accuracy as the threshold is increased.

Table 1: Accuracy metrics across 71 prompts. FineAMP(28) denotes the FineAMP algorithm with a threshold of 28. Standard deviations are reported for each metric.

Model	Quantization profile	% of 16 bit ops	PSNR (dB) ↑	HPS ↑	LPIPS ↓
SDXL - Unet	FP32	N/A	N/A	0.317 ± 0.037	N/A
	W8A16	100%	29.1 ± 3.5	0.317 ± 0.031	0.055 ± 0.031
	FineAMP (28)	44.0%	25.9 ± 2.8	0.317 ± 0.036	0.091 ± 0.035
	FineAMP (26)	42.9%	24.9 ± 2.5	0.307 ± 0.037	0.112 ± 0.036
	W8A8	0%	10.3 ± 1.0	0.131 ± 0.021	0.51 ± 0.06
LCM-SSD-1B - Unet	FP32	N/A	N/A	0.294 ± 0.040	N/A
	W8A16	100%	27.8 ± 2.4	0.294 ± 0.040	0.065 ± 0.025
	FineAMP (26)	78.1%	25.7 ± 2.7	0.293 ± 0.040	0.087 ± 0.032
	FineAMP (24)	55.4%	24.0 ± 2.4	0.292 ± 0.040	0.113 ± 0.040
	W8A8	0%	11.3 ± 1.6	0.154 ± 0.037	0.480 ± 0.041

On-device evaluation results. Table 2 shows latency results for a single forward pass of the Unet model of SDXL when deployed to a device with Snapdragon® 8 platform[1]. The first column is the threshold that we use in determining bit-widths; if the estimated SQNR for an operation is lower than the threshold, we select 16 bits for that operation, otherwise it is quantized to 8 bits. The second column shows the latency on the hardware. The last column is the simulated accuracy for the corresponding mixed precision profile. Table 2 shows that FineAMP can generate mixed precision profiles that run faster on device than W8A16 while maintaining high image quality. Higher thresholds worsen the latency due to more 16-bit operations.

Typically image-level SQNRs above 15-16 dB indicate that the visual distortions in the generated images due to quantization are minimal and are hard to spot. As can be seen in

1. Snapdragon is a product of Qualcomm Technologies, Inc. and/or its subsidiaries.

Table 2: On-device latency for a single SDXL Unet step under mixed precision profiles from FineAMP, with simulated image-level SQNR (dB) averaged over 10 prompts.

Threshold (dB)	Latency (sec)	Image SQNR (dB)
∞ (W8A16)	0.607	22.2
30	0.608	20.3
28	0.583	18.3
26	0.504	17.2
24	0.489	16.6
20	0.447	12.4

Table 2, when the threshold is set to 24, FineAMP generates a mixed precision profile that achieves an image-level SQNR of 16.6 dB with a latency of 0.489 seconds, 19.4% lower than running all operations in W8A16. We show in the appendix that the latency can be further reduced by utilizing our conversion cost control algorithm, leading to a very efficient and accuracy-preserving quantization scheme.

4. Conclusion

Quantization is key for deploying AI on limited-resource devices. While manual bit-width tuning is common, automatic mixed precision simplifies the process. Our method efficiently estimates operation sensitivities with minimal accuracy evaluations, enabling faster search and hardware-friendly profiles with improved latency.

References

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

AIMET. Ai model efficiency toolkit (aimet), 2025. URL https://quic.github.io/aimet-pages/.

CBC. Coin-or branch-and-cut solver, 2025. URL https://coin-or.github.io/Cbc/intro.html.

Weihan Chen, Peisong Wang, and Jian Cheng. Towards mixed-precision quantization of neural networks via constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5350–5359, 2021.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. HAWQ: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 293–302, 2019.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Muyang Li, Yujun Lin, Zhekai Zhang, Tianle Cai, Xiuyu Li, Junxian Guo, Enze Xie, Chenlin Meng, Jun-Yan Zhu, and Song Han. Svdquant: Absorbing outliers by low-rank components for 4-bit diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

Darryl D. Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2849–2858. JMLR.org, 2016.

Xingchao Liu, Mao Ye, Dengyong Zhou, and Qiang Liu. Post-training quantization with multiple points: Mixed precision without mixed precision. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 8697–8705, 2021.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. 2023. URL https://arxiv.org/abs/2310.04378.

Yuexiao Ma, Taisong Jin, Xiawu Zheng, Yan Wang, Huixia Li, Yongjian Wu, Guannan Jiang, Wei Zhang, and Rongrong Ji. OMPQ: Orthogonal mixed precision quantization. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 9029–9037, 2023.

Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning*, pages 7197–7206. PMLR, 2020.

Nilesh Prasad Pandey, Markus Nagel, Mart van Baalen, Yin Huang, Chirag Patel, and Tijmen Blankevoort. A practical mixed precision algorithm for post-training quantization. 2023. URL https://arxiv.org/abs/2302.05397.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. *arXiv*, (2307.01952), 2023.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving latent diffusion models for high-resolution image synthesis. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=di52zR8xgf.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Sangeetha Siddegowda, Marios Fournarakis, Markus Nagel, Tijmen Blankevoort, Chirag Patel, and Abhijit Khobare. Neural network quantization with ai model efficiency toolkit (aimet). 2022. URL https://arxiv.org/abs/2201.08442.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. HAQ: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8612–8620, 2019.

Ziwei Wang, Han Xiao, Jiwen Lu, and Jie Zhou. Generalizable mixed-precision quantization via attribution rank preservation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5291–5300, 2021.

Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018.

Di Wu, Qi Tang, Yongle Zhao, Ming Zhang, Ying Fu, and Debing Zhang. EasyQuant: Post-training quantization via scale optimization. *arXiv preprint arXiv:2006.16669*, 2020.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. 2023. URL https://arxiv.org/abs/2306.09341.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael Mahoney, and Kurt Keutzer. Hawq-v3: Dyadic neural network quantization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11875–11886. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yao21a.html.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. 2018. URL https://arxiv.org/abs/1801.03924.

Tianchen Zhao, Xuefei Ning, Tongcheng Fang, Enshu Liu, Guyue Huang, Zinan Lin, Shengen Yan, Guohao Dai, and Yu Wang. Mixdq: Memory-efficient few-step text-to-image diffusion models with metric-decoupled mixed precision quantization, 2024.

Appendix A. Preliminaries

In this section, we provide background on diffusion models and post-training quantization as well as provide relevant references to prior work.

A.1. Diffusion models

Diffusion models synthesize images through a Markov chain. The forward diffusion process gradually adds Gaussian noise to data point $\boldsymbol{x}_0$ over T iterations, producing noisy samples $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ through the transition given by:

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) := \mathcal{N}(\sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}), \tag{7}$$

where $0 < \beta_t < 1$ denotes the variance schedule, controlling the magnitude of Gaussian noise added at each step. As $T \to \infty$, $\boldsymbol{x}_T$ converges to the standard Gaussian $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

The reverse process progressively denoises a sample initialized from $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ to generate high-quality images step by step. However, since the true reverse conditional distribution $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ is unavailable, Ho et al. (2020) introduced a reparameterization trick that approximates $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$ via a learned Gaussian distribution:

$$\mathcal{N}\left(\frac{\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{\sqrt{1-\beta_t}}, \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t \boldsymbol{I}\right), \tag{8}$$

where $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ is a neural network with parameters $\boldsymbol{\theta}$, and $\bar{\alpha}_t := \prod_{i=1}^{t}(1-\beta_i)$.

In practice, the Unet architecture (Ronneberger et al. (2015)) has become the predominant choice for the noise estimation network $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$ in diffusion models (Ho et al. (2020); Song et al. (2020); Rombach et al. (2022); Saharia et al. (2022); Podell et al. (2023)), though recent works have started investigating Transformer-based alternatives (Peebles and Xie (2023)).

A.2. Post-training quantization

Quantization process. Quantization maps floating-point numbers to values from a finite discrete set. A widely used post-training quantization technique is *uniform quantization*: Given a floating-point tensor $\boldsymbol{x}$, a scale factor $\sigma > 0$, a zero point $z \in \mathbb{R}$, and a bit-width $b \in \mathbb{N}$, the uniform quantization process can be expressed as:

$$\widetilde{\boldsymbol{x}} := \sigma \cdot \left(\text{clip}\left(\left\lfloor \frac{\boldsymbol{x}}{\sigma} \right\rceil + z, 0, 2^b - 1\right) - z\right), \tag{9}$$

where $\lfloor \cdot \rceil$ denotes the rounding operator. These parameters are calibrated using the weight and activation distributions obtained during the post-training quantization process. The rounding operator $\lfloor \cdot \rceil$ can follow different strategies, such as nearest rounding (Wu et al. (2020); Xiao et al. (2023)) or adaptive rounding (Nagel et al. (2020)).

Quantization error. *Quantization error* refers to the impact of quantizing an operation on the overall model accuracy. Lin et al. (2016) uses the Signal-to-Quantization-Noise Ratio (SQNR) as a metric to quantify quantization error. If $\boldsymbol{y}$ and $\widetilde{\boldsymbol{y}}$ are the outputs without and with quantization, respectively, then the SQNR is defined as

$$\text{SQNR} := 10 \log_{10} \frac{\|\boldsymbol{y}\|_2^2}{\|\widetilde{\boldsymbol{y}} - \boldsymbol{y}\|_2^2}. \tag{10}$$

Lin et al. (2016) also theoretically derives a formula that characterizes the relationship between the SQNRs when quantizing individual operations and the SQNR when quantizing the entire model. We leverage their formula in our optimization problem formulation in Section 2.

A.3. Mixed precision quantization

Automatic mixed precision. To achieve a balance between computational efficiency and quantization error, *mixed precision* can be leveraged, where a different bit-width is assigned to different components in a model. The optimal bit-width assignments for a model do not necessarily carry over to other models. *Automatic mixed precision* (AMP) algorithms aim to determine the optimal assignments automatically. Many AMP methods have been proposed in the literature, including optimization-based methods (Lin et al. (2016); Liu et al. (2021); Chen et al. (2021); Wang et al. (2021)), search-based methods (Wang et al. (2019); Wu et al. (2018)), and higher-order methods (Dong et al. (2019); Ma et al. (2023); Pandey et al. (2023)).

Zhao et al. (2024) proposes MixDQ, an integer programming-based method for determining mixed-precision configurations. Like many prior works on mixed precision for large models, MixDQ does not quantize all operations to integers; instead, it retains FP16 implementations for certain operations such as layer-norm and non-linear activations. In contrast, our approach quantizes all operations to either 8-bit or 16-bit integers, completely eliminating floating point operations. This distinction is significant, as operations like normalization and activations (e.g., softmax) are often computationally expensive. Executing them in floating point on mobile devices can substantially degrade latency.

Furthermore, some methods report acceptable SQNR even with 4-bit quantization or lower. However, this is often due to: (1) the use of training-based techniques, which are beyond the scope of this work, or (2) the incorporation of low-rank branches (e.g., Li et al. (2025)) to mitigate quantization error which is an approach that introduces considerable overhead and hardware-specific implementation challenges.

Many AMP algorithms rely on a step where the quantization sensitivities of different operations or operation types are measured. Measuring the sensitivity of an operation to quantization may require multiple inferences. One standard approach to obtaining the quantization sensitivity for an operation is to quantize the given operation while the rest of the operations run in full precision. The goal here is to measure the impact of quantizing that operation on the final model output while the rest of the model remains unchanged. In order to avoid many inferences, existing works focus on operation-type based mixed precision. For a given large model, usually INT and FP precision are mixed to achieve good accuracy, and the decision of how to mix is based on expert knowledge on model structure and empirical analysis.

Fine-grained AMP. Fine-grained AMP seeks to assign different bit-widths to individual operations, offering a better precision–efficiency balance compared to layer-level (or operation-type level) AMP. However, as discussed above, existing AMP methods face scalability challenges due to their dependence on the number of quantized operations. Consequently, existing methods become prohibitively expensive for fine-grained AMP on large diffusion models with thousands of operations. In stark contrast to existing approaches, our method

requires significantly fewer model evaluations than the number of quantized operations, enabling it to efficiently scale to large diffusion models with thousands of operations.

Conversion cost. When a model employs mixed precision, conversions between different precisions incur additional computational cost. Yao et al. (2021) emphasizes the problem with conversion cost causing extra overhead, which is an aspect that is typically neglected by many prior works in the literature. We formulate an integer linear programming (ILP) based solution to address the conversion cost problem in Section B and verify that it brings latency benefits.

On the metrics. It is important to note that SQNR directly measures the similarity between images generated by the quantized model and those produced by the floating point implementation. Unlike many commonly used metrics in prior work, which assign separate scores to the floating point and quantized outputs, SQNR compares the images pixel by pixel. The closeness of individual scores does not necessarily imply visual similarity between the images. Since the primary goal of post-training quantization is to preserve the original model's output as much as possible, comparing the images themselves offers a more reliable assessment than comparing separate evaluation scores. For this reason, we highlight SQNR as the preferred metric for evaluating image quality, while also reporting results for other widely used metrics.

Appendix B. Conversion cost control

The main body of the paper has focused on assigning bit-widths based on the quantization sensitivity of operations. Bit-width conversions themselves can be viewed as separate operations that take up processor cycles. Having too many high-cost conversions leads to a significant latency increase. This section tackles the problem of reducing the conversion cost given a mixed precision profile. By design, the accuracy of the solution produced by the conversion cost control algorithm is as good as or better than the mixed precision solution we start with.

We do not model the accuracy explicitly in the formulation of the conversion cost control algorithm, but rather build our design on top of an existing mixed precision solution, which could be obtained through any mixed precision algorithm such as the one we presented in Section 2. We will introduce an integer linear programming (ILP) formulation to address the conversion cost control. The decision space of the ILP formulation is restricted to moving operations from lower bit-width to higher bit-width. This ensures that the accuracy of the generated solution statistically should be at least as good as the initial mixed precision profile, with an improved latency since we now reduce the bit-width conversion cost. More precisely, the goal of the ILP formulation in this section is to minimize the increase in the cost of the operations due to moving to higher bit-width while keeping the total conversion cost under a fraction of the cost from the initial solution.

The algorithm requires only a computation graph $G = (V, E)$ and an initial bit-width assignment, where nodes $v \in V$ represent operations and edges $(u, v) \in E$ denote data dependencies. The formulation below is general in the sense that it can work with any mixed precision profile and any computation graph. Practitioners can also apply the conversion cost algorithm to manually designed profiles to reduce latency by managing conversion overhead.

For simplicity and consistency with the previous section, we will restrict the formulation to mixed precision between 8 bits and 16 bits. We model the conversion cost as follows. Let $f_{u,v}$ denote the cost of conversion for the tensor between the nodes u and v and let C_f be the total conversion cost for the initial mixed precision profile. We then limit the sum of all tensor bit-width conversion costs to $\alpha \times C_f$, where $\alpha \in [0, 1]$. The ILP formulation is stated as follows:

$$\underset{x,y}{\text{minimize}} \quad \sum_{v \in V} (h_{v,8} x_v + h_{v,16} y_v) \tag{11}$$

$$\text{subject to} \quad x_v + y_v = 1, \, \forall v \in V \tag{12}$$

$$y_v = 1, \, \forall v \in \{u \in V : \text{init}(u) = 16\} \tag{13}$$

$$\sum_{(u,v) \in E} |x_u - x_v| f_{u,v} \leq \alpha C_f \tag{14}$$

$$x_v, y_v \in \{0, 1\}, \, \forall v \in V \, . \tag{15}$$

$h_{v,8}$ and $h_{v,16}$ in the objective shown in (11) denote the cost of running an operation v in 8 and 16 bits, respectively. The objective is the sum of costs for all the nodes $v \in V$ in the graph.

The variables for the ILP are x_v and y_v, defined for each node v. The variables x_v and y_v are modeled as Boolean, i.e., 0-1, decision variables. An assignment of $x_v = 1$ indicates that the operation v uses 8 bits and $y_v = 1$ indicates 16 bits. The constraints in (12) ensure that a node cannot be assigned two different bit-widths. Next, (13) ensures that the only time we change the bit-width for an operation is when we increase the bit-width from 8 to 16. The notation init(u) indicates the initial bit-width assignment for node u. The conversion cost control is achieved through the constraint in (14).

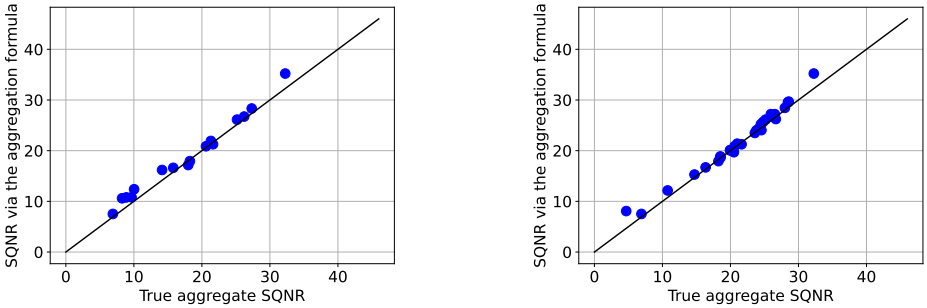
Since overall latency is a nonlinear (and unknown) function of bit-width conversion and operation costs, it is challenging to determine a good balance between the total conversion cost and operation cost. An effective value for α can be identified by evaluating mixed precision profiles generated with various α values between 0 and 1, using on-device latency measurements. The operation cost estimates $h_{v,8}$, $h_{v,16}$ can be obtained from on-device executions. Finally, as in Section 2, we solve the ILP using the CVXPY package with the open-source CBC solver.

Appendix C. Numerical verification of the formula in (6)

Figure 2 is a numerical verification of the formula in (6). For this numerical verification, we have collected data measuring the individual operation sensitivities for all the operations and the aggregate sensitivities. The true aggregate SQNR and the computed SQNR using the formula in (6) are plotted in Figure 2. This figure clearly demonstrates that the SQNR aggregation formula provides a very good fit to the numerical data.

Appendix D. Additional numerical results

Example images. We provide some visual comparisons for models with different precisions in this subsection. We show example images in Figure 3, which are generated by the SDXL model under different quantization configurations for the Unet:



a) Block subsets b) Operation type subsets

Figure 2: The SQNR aggregation formula in (6) closely aligns with the numerical data.

- Image a was generated by the full precision model.
- Image b was generated using the operation-type based mixed precision profile (from the green curve of Figure 1) with 1043 operations quantized to 16 bits.
- Image c was generated using a mixed precision profile that FineAMP produces when the threshold is set to 28. This mixed precision profile has 857 operations quantized to 16 bits.

a) FP32 (no quantization) b) Op-type mixed precision c) FineAMP (28)

Figure 3: Example generated images using SDXL. The prompt is "a hyper realistic photo of a beautiful cabin inside of a forest and full of trees and plants, with large aurora borealis in the sky".

There are fewer distortions and artifacts in image c compared to image b even though the mixed precision profile for image b has more operations at 16 bits compared to image c. This is consistent with the SQNR evaluations in Figure 1. Image c is more similar to the full precision image than image b.

Impact of conversion cost reduction on latency. In this subsection, we investigate the on-device latency impact of the algorithm introduced in Section B. We have sampled α from the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$ and deployed each corresponding mixed precision profile on the hardware. In Table 3, we report the best latency out of the 5 for each mixed precision profile. The first column shows the threshold used in FineAMP and the second column

Table 3: On-device latency results for bit-width conversion cost reduction.

Threshold (dB)	Latency (sec)	% Reduction	Best α
∞ (W8A16)	0.607	-	-
30	0.573	5.8%	0.7
28	0.555	4.8%	0.7
26	0.502	0.4%	0.7
24	0.448	8.4%	0.9

shows the on-device latency for a single step of the Unet component of SDXL. The third column shows latency reduction (%) over the original profile. The last column shows the α value that gives the best latency on device. This shows that not a significant variability exists in the lowest latency achieving α.

Table 3 illustrates that the conversion cost control algorithm introduced in Section B is able to reduce the latency for different mixed precision profiles. Although the latency gain is modest in some cases, it is worth emphasizing that the algorithm takes only a few seconds to run and does not degrade the model performance since it only keeps or increases the bit-width for any operation.

It is worth noting that in Table 2, the mixed precision profile of FineAMP with threshold 30 has a higher latency than W8A16 (i.e. all operations run in 16 bits). This is due to the large number of bit-width conversions that FineAMP(30) profile has. In Table 3, we see that a reduction on the conversion cost causes an overall 5.8% lower latency, reducing from 0.608 seconds to 0.573 seconds.