ZeroQ: A Novel Zero Shot Quantization Framework

Yaohui Cai[1], Zhewei Yao[2], Zhen Dong[2], Amir Gholami[2], Michael W. Mahoney[2], Kurt Keutzer[2]

[1]Peking University; [2]University of California, Berkeley

CVPR 2020

Quantization

- Quantization converts the floating-point (FP32) weights and activations of deep CNNs to low-bit integers.
- It can be written as

$$Q(x) = \texttt{round}\big(x/\Delta + z\big),$$

where, $\Delta = n/(\texttt{max}(x) - \texttt{min}(x)), \quad z = \Delta * \texttt{min}(x).$

with $n = 2^b - 1$ and b denotes low-bit integer precision.

How to compute Δ, z before inference?

- For weights, it can be computed directly.
- For activations, requires access to original training samples in full or subset.
- Once samples are available, generate activations, record the ranges and compute Δ, z of the same.
- One can also set mixed precision for DNNs, *i.e.*, different layers can have different bit precision.
- The main idea behind mixed-precision quantization is to keep more sensitive layers at higher precision, and more aggressively quantize less sensitive layers, without increasing overall model size.

Sensitivity?

- Sensitivity is defined as

$$\Omega_i(k) = \frac{1}{N} \sum_{j=1}^{N_{dist}} \mathrm{KL}(\mathcal{M}(\theta; x_j), \mathcal{M}(\tilde{\theta}_i(k\text{-}bit); x_j)).$$

- If sensitivity is small, the output of the quantized model will not significantly deviate from the output of the full precision model when quantizing the *i-th* layer to *k*-bits.
- Hence, the *i-th* layer is relatively **insensitive** to *k*-bit quantization, and vice versa.

Sensitivity? Cont.

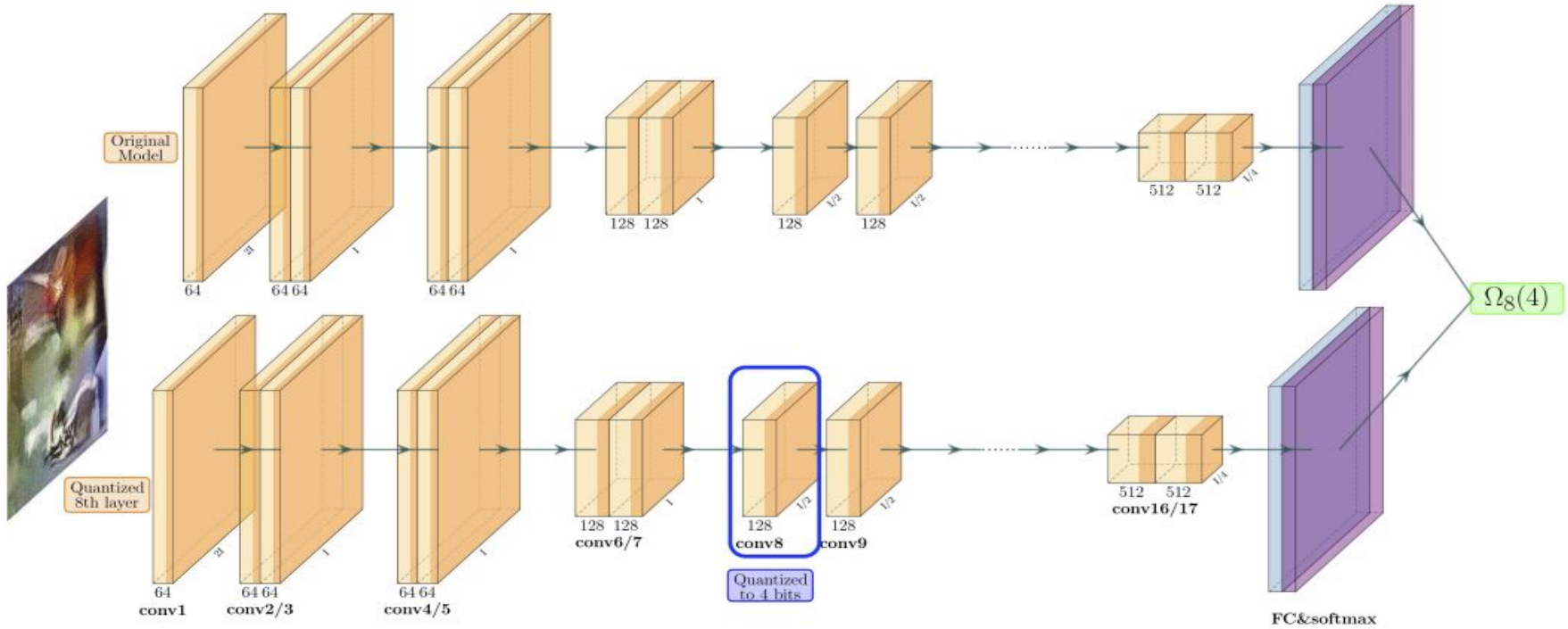


Figure 1: Illustration of sensitivity computation for ResNet18 on ImageNet. The figure shows how we compute the sensitivity of the 8-th layer when quantized to 4-bit ($\Omega_8(4)$) according to Eq. 2. We feed Distilled Data into the full-precision ResNet18 (top), and the same model except quantizing the 8-th layer to 4-bit (bottom) receptively. The sensitivity of the 8-th layer when quantized to 4-bit $\Omega_8(4)$ is defined as the KL-divergence between the output of these two models. For simplicity, we omit the residual connections here, although the same analysis is applied to the residual connections in ZEROQ.

If original training samples aren't available?

$$\min_{x^r} \sum_{i=0}^{L} \|\tilde{\mu}_i^r - \mu_i\|_2^2 + \|\tilde{\sigma}_i^r - \sigma_i\|_2^2, \qquad (3)$$

Figure 3: Visualization of Gaussian data (left) and Distilled Data (right). More local structure can be seen in our Distilled Data that is generated according to Algorithm 1.

Algorithm 1: Generation of Distilled Data

Input: Model: $\mathcal{M}$ with L Batch Normalization layers

Output: A batch of distilled data: x^r

Generate random data from Gaussian: x^r

Get μ_i, σ_i from Batch Normalization layers of $\mathcal{M}$, $i \in 0, 1, \ldots, L$ // Note that $\mu_0 = 0$, $\sigma_0 = 1$

for $j = 1, 2, \ldots$ **do**

- Forward propagate $\mathcal{M}(x^r)$ and gather intermediate activations
- Get $\tilde{\mu}_i$ and $\tilde{\sigma}_i$ from intermediate activations, $i \in 1, \ldots, n$
- Compute $\tilde{\mu}_0$ and $\tilde{\sigma}_0$ of x^r
- Compute the loss based on Eq. 3
- Backward propagate and update x^r

- Authors addressed this by "distilling" a synthetic input data to match the statistics of the original training dataset.
- Referred to as Distilled Data.
- They obtained the Distilled Data by solely analyzing the trained model itself.

Gaussian vs. Distilled vs. Training data

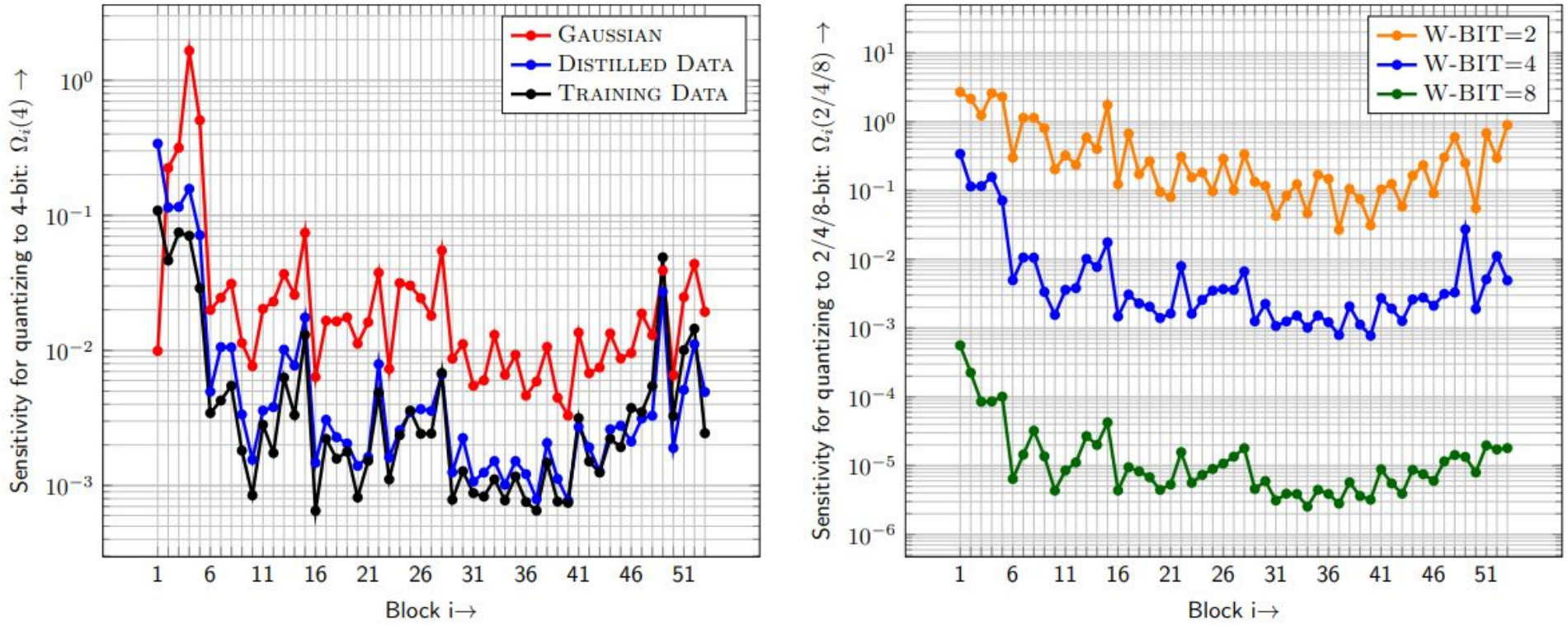


Figure 2: (Left) Sensitivity of each layer in ResNet50 when quantized to 4-bit weights, measured with different kinds of data (red for Gaussian, blue for Distilled Data, and black for training data). (Right) Sensitivity of ResNet50 when quantized to 2/4/8-bit weight precision (measured with Distilled Data).

Sensitivity and Mixed Precision (MP)

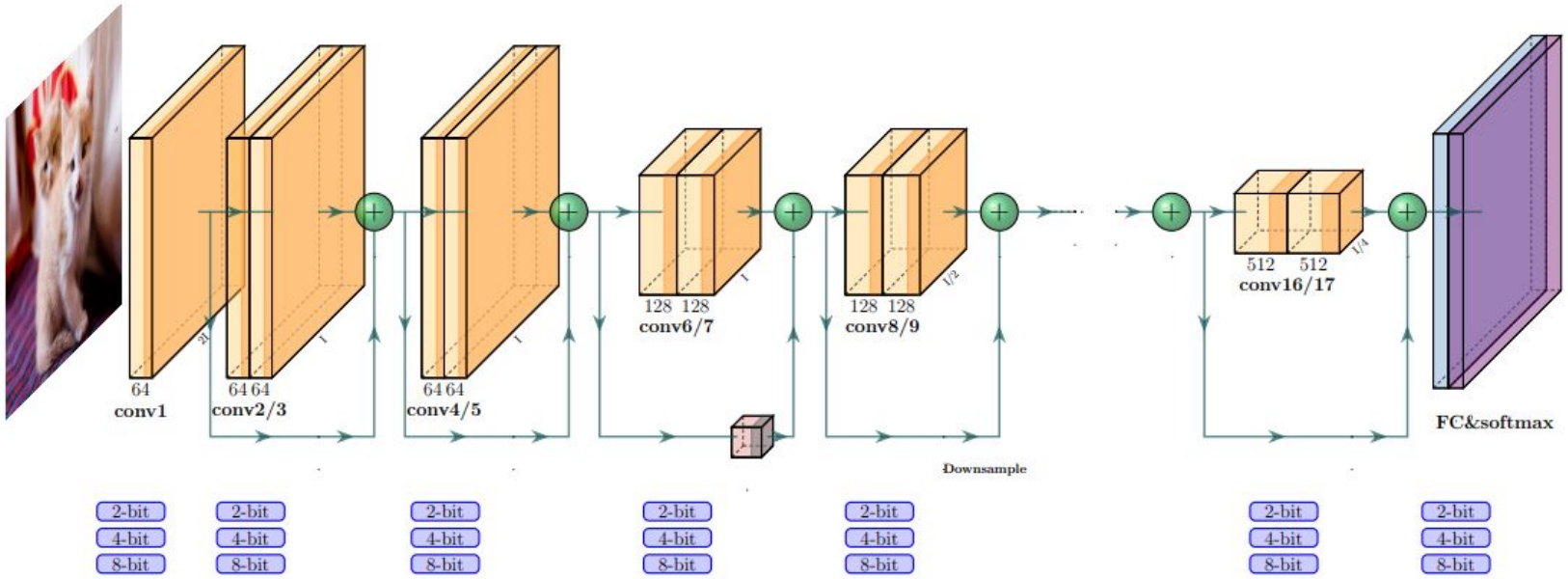


Figure 5: Mixed precision illustration of ResNet18 on ImageNet.

- Optimizes following to set mixed-precision quant.

$$\min_{\{k_i\}_{i=1}^{L}} \Omega_{sum} = \sum_{i=1}^{L} \Omega_i(k_i) \;\; s.t. \sum_{i=1}^{L} P_i * k_i \leq S_{target},$$

Pareto Frontier for MP

- For a model with *L* layers, each having a candidate bit from set of *m* bits, total search space can have m^L .
- *Independence Assumption:* the sensitivity of each layer to quantization is independent to sensitivity of other layers.
- Instead of computing the sensitivity of the entire network at once, we break the network into *L/a* groups, with each group containing *a* layers.
- Furthermore, we break the x-axis (model size) of the Pareto frontier plot into b intervals.
- Start with first *a layers* and compute mxa sensitivities with independence assumption.
- Afterwards, for each interval on the x-axis, we choose top t˜ configurations that have the lowest overall sensitivity when the first *a* layers are quantized.

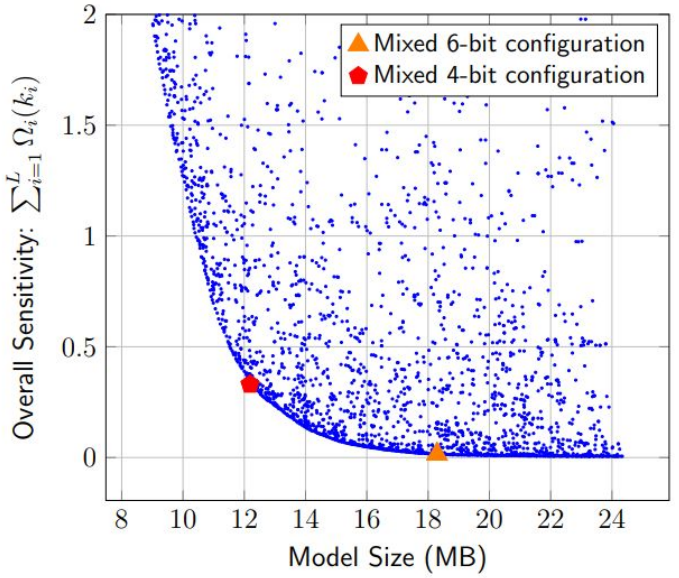


Figure 4: The Pareto frontier of ResNet50 on ImageNet. Each point shows a mixed-precision bit setting. The x-axis shows the resulting model size for each configuration, and the y-axis shows the resulting sensitivity. In practice, a constraint for model size is set. Then the Pareto frontier method chooses a bit-precision configuration that results in minimal perturbation. We show two examples for 4 and 6-bit mixed precision configuration shown in red and orange. The corresponding results are presented in Table 1a.

Pareto Frontier for MP cont.

- It then relaxes the independence assumption for these t˜ configurations and recompute the overall sensitivity, Ω_{sum}, without any approximation.
- This leads to a cost of t˜ of computing that equation.
- We then select the top t configurations out of these.
- We conduct this process for all the b intervals for the model size. Therefore, the total cost will be t˜× b.
- The next step is to consider the next set of *a* layers.
- Similar to above steps, it selects t˜× b configs. for second set of *a* layers out of t × b × m^a configurations. [t × b for first set of *a* layers and m^a for second set].
- It then selects the final top t × b configurations for the first 2a layers based on this.
- This process needs to be performed for all the *L/a* groups.
- The total computational cost becomes (*L/a*) × t˜× b + *m* × *L*.
- We typically set t˜, t, b, a to be 10, 5, 200, 5, respectively.

Results

Table 1: Quantization results of ResNet50, MobileNetV2, and ShuffleNet on ImageNet. We abbreviate quantization bits used for weights as "W-bit" (for activations as "A-bit"), top-1 test accuracy as "Top-1." Here, "MP" refers to mixed-precision quantization, "No D" means that none of the data is used to assist quantization, and "No FT" stands for no fine-tuning (re-training). Compared to post-quantization methods OCS [44], OMSE [19], and DFQ [32], ZEROQ achieves better accuracy. ZEROQ† means using percentile for quantization.

(a) ResNet50

Method	No D	No FT	W-bit	A-bit	Size (MB)	Top-1
Baseline	–	–	32	32	97.49	77.72
OMSE [19]	✓	✓	4	32	12.28	70.06
OMSE [19]	✗	✓	4	32	12.28	74.98
PACT [5]	✗	✗	4	4	12.19	76.50
ZEROQ	✓	✓	MP	8	**12.17**	**75.80**
ZEROQ†	✓	✓	MP	8	**12.17**	**76.08**
OCS [44]	✗	✓	6	8	18.46	74.80
ZEROQ	✓	✓	MP	6	**18.27**	**77.43**
ZEROQ	✓	✓	8	8	24.37	**77.67**

(b) MobileNetV2

Method	No D	No FT	W-bit	A-bit	Size (MB)	Top-1
Baseline	–	–	32	32	13.37	73.03
ZEROQ	✓	✓	MP	8	1.67	**68.83**
ZEROQ†	✓	✓	MP	8	1.67	**69.44**
Integer-Only [18]	✗	✗	6	6	2.50	70.90
ZEROQ	✓	✓	MP	6	2.50	**72.85**
RVQuant [33]	✗	✗	8	8	3.34	70.29
DFQ [32]	✓	✓	8	8	3.34	71.20
ZEROQ	✓	✓	8	8	3.34	**72.91**

(c) ShuffleNet

Method	No D	No FT	W-bit	A-bit	Size (MB)	Top-1
Baseline	–	–	32	32	5.94	65.07
ZEROQ	✓	✓	MP	8	0.74	**58.96**
ZEROQ	✓	✓	MP	6	1.11	**62.90**
ZEROQ	✓	✓	8	8	1.49	**64.94**

Table 2: Object detection on Microsoft COCO using RetinaNet. By keeping activations to be 8-bit, our 4-bit weight result is comparable with recently proposed method FQN [25], which relies on fine-tuning. (Note that FQN uses 4-bit activations and the baseline used in [25] is 35.6 mAP).

Method	No D	No FT	W-bit	A-bit	Size (MB)	mAP
Baseline	✓	✓	32	32	145.10	36.4
FQN [25]	✗	✗	4	4	18.13	32.5
ZEROQ	✓	✓	MP	8	18.13	**33.7**
ZEROQ	✓	✓	MP	6	24.17	**35.9**
ZEROQ	✓	✓	8	8	36.25	**36.4**

Results cont.

Table 3: Uniform post-quantization on ImageNet with ResNet18. We use percentile clipping for W4A4 and W4A8 settings. ZEROQ† means using percentile for quantization.

Method	No D	No FT	W-bit	A-bit	Size (MB)	Top-1
Baseline	–	–	32	32	44.59	71.47
PACT [5]	✗	✗	4	4	5.57	69.20
DFC [12]	✓	✓	4	4	5.58	55.49
DFC [12]	✓	✗	4	4	5.58	68.06
ZEROQ†	✓	✓	MP	4	**5.57**	**69.05**
Integer-Only[18]	✗	✗	6	6	8.36	67.30
DFQ [32]	✓	✓	6	6	8.36	66.30
ZEROQ	✓	✓	MP	6	**8.35**	**71.30**
RVQuant [33]	✗	✗	8	8	11.15	70.01
DFQ [32]	✓	✓	8	8	11.15	69.70
DFC [12]	✓	✗	8	8	11.15	69.57
ZEROQ	✓	✓	8	8	11.15	**71.43**

Table 4: ResNet20 on CIFAR-10

Method	No Data	No FT	W-bit	A-bit	Size (MB)	Top-1
Baseline	–	–	32	32	1.04	94.03
ZEROQ	✓	✓	MP	8	0.13	**93.16**
ZEROQ	✓	✓	MP	6	0.20	**93.87**
ZEROQ	✓	✓	8	8	0.26	**93.94**

Table 5: Additional results on ImageNet

(a) ResNet152

Method	No D	No FT	w-bit	a-bit	Size (MB)	Top-1
Baseline	–	–	32	32	229.62	80.08
ZEROQ	✓	✓	MP	8	28.70	**78.00**
ZEROQ	✓	✓	MP	6	43.05	**77.88**
RVQuant [33]	✗	✗	8	8	57.41	78.35
ZEROQ	✓	✓	8	8	57.41	**78.94**

(b) InceptionV3

Method	No D	No FT	W-bit	A-bit	Size (MB)	Top-1
Baseline	–	–	32	32	90.92	78.88
ZEROQ	✓	✓	MP	8	**11.35**	**77.57**
OCS[44]	✗	✓	6	6	17.22	71.30
ZEROQ	✓	✓	MP	6	**17.02**	**78.76**
RVQuant [33]	✗	✗	8	8	22.47	74.22
ZEROQ	✓	✓	8	8	22.47	**78.81**

(c) SqueezeNext

Method	No D	No FT	W-bit	A-bit	Size (MB)	Top-1
Baseline	–	–	32	32	9.86	69.38
ZEROQ	✓	✓	MP	8	1.23	**59.23**
ZEROQ	✓	✓	MP	6	1.85	**68.17**
ZEROQ	✓	✓	8	8	2.47	**69.17**

Table 6: Ablation study for Distilled Data on ResNet18, ShuffleNet and SqueezeNext. We show the performance of ZEROQ with different sources of data to compute the sensitivity and determine the activation range. All quantized models have the same size as quantized models with 4-bit weights.

Method	W-bit	A-bit	ResNet18	ShuffleNet	SqueezeNext
Baseline	32	32	71.47	65.07	69.38
Gaussian	MP	8	67.87	56.23	48.41
Training Data	MP	8	68.61	58.90	62.55
Distilled Data	MP	8	**68.45**	**57.50**	**59.23**

Mask Guided Matting via Progressive Refinement Network

Qihang Yu[1], Jianming Zhang[2], He Zhang[2], Yilin Wang[2], Zhe Lin[2], Ning Xu[2], Yutong Bai[1], Alan Yuille[1]

[1]The Johns Hopkins University; [2]Adobe

CVPR 2021

What is image matting?

1. Image matting refers to precisely separating the foreground region from the background in an image.

$$\mathbf{I} = \alpha\mathbf{F} + (1-\alpha)\mathbf{B}, \alpha \in [0,1],$$

where, **I** is composited image, **F** is foreground, **B** is background and *α* denotes the alpha matte with each pixel value in [0,1].

2. Given the highly ill-posed nature of the problem and lack of large-scale labeled datasets, existing works leverage various prior-based techniques.

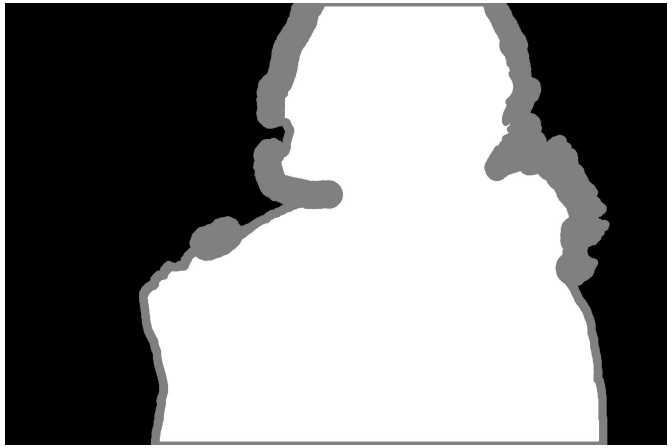
3. A majority of such techniques use ***trimaps***.

4. A trimap divides the image into three regions: foreground, background, and transitions (*a.k.a* unknown or gray) regions.

What is image matting? Cont.

- The matting task is then simplified to estimate the unknown values only in the transition region, reducing the solution space.

Input



Trimap

Estimated Alpha Matte

MG Matting: Observations

- End to end prediction of alpha matte by just using a single image needs large-scale datasets.
- The unavailability of such datasets, may lead to an incapable model due to lack of semantic guidance that may not generalize well on unseen data.
- Hence, learning-based methods require some additional guidance along with input image to precisely estimate its alpha matte.
- However, the specific guidance may restrict the robustness of the model against variety of priors.
- Majority of the existing works use trimap-based guidance.
- The MG Matting, on the other hand, works in a more general setting, where the guidance can be
  - a trimap,
  - a rough binary segmentation mask,
  - a soft alpha matte,
  - or any easy-to-obtain coarse mask irrespective of the user-defined or model-predicted.

MG Matting

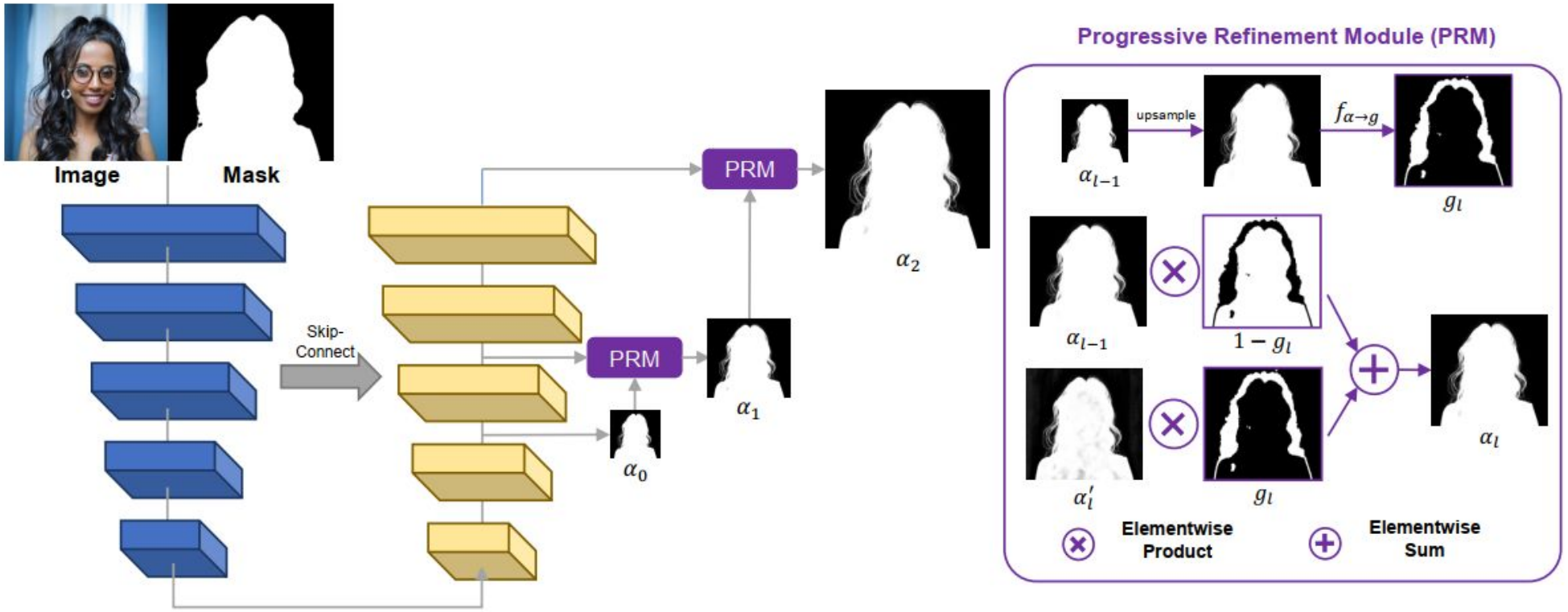


Figure 2: The proposed PRN. The network predicts alpha matte at multiple resolutions, while the one at lower-resolution provides guidance about uncertain region to be refined in the next prediction.

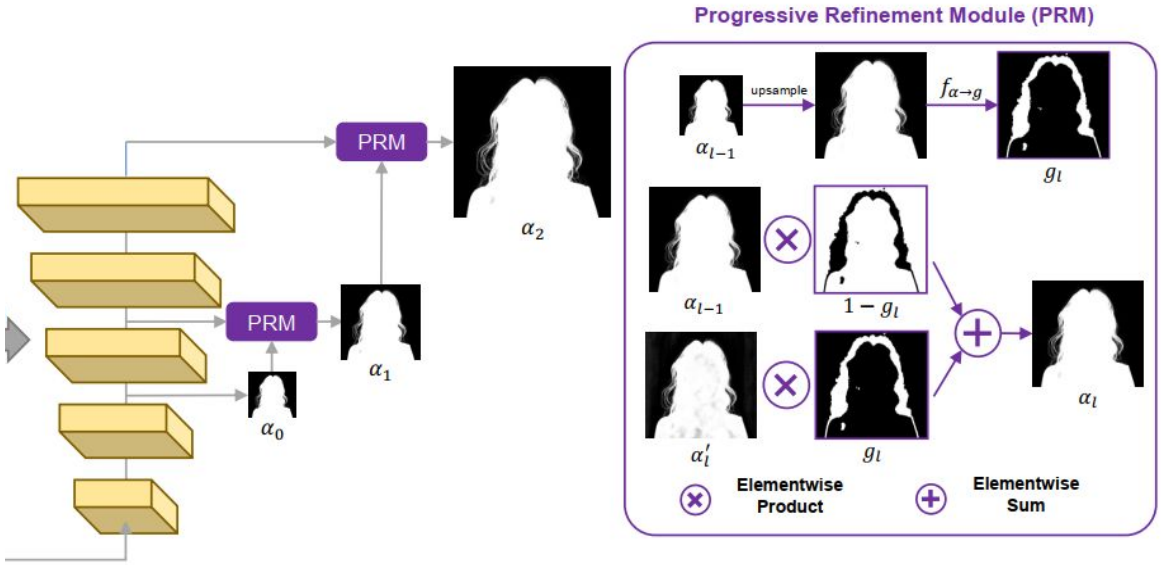
MG Matting cont.

- The PRM, for a level *l* first generates the self-guidance mask g_l from the matting output of previous level α_{l-1} as,

$$f_{\alpha_{l-1}\to g_l}(x,y) = \begin{cases} 1 & \text{if } 0 < \alpha_{l-1}(x,y) < 1, \\ 0 & \text{otherwise.} \end{cases}$$

- The self guidance mask is then used to preserve the confident regions from previous output features and allows current layer to only focus on refining the uncertain regions, as

$$\alpha_l = \alpha'_l g_l + \alpha_{l-1}(1 - g_l).$$



- Base g_0 is filled with 1.
- Loss includes self-guidance from 3-levels

$$\mathcal{L}(\hat{\alpha}, \alpha) = \mathcal{L}_{l1}(\hat{\alpha}, \alpha) + \mathcal{L}_{comp}(\hat{\alpha}, \alpha) + \mathcal{L}_{lap}(\hat{\alpha}, \alpha).$$

$$\mathcal{L}_{final} = \sum_l w_l \mathcal{L}(\hat{\alpha}_l \cdot g_l, \alpha_l \cdot g_l), \quad w_0 : w_1 : w_2 = 1 : 2 : 3$$

Training

1. To make the network robust against the guidance as input, they used gt alpha matte and generated guidance by using series of guidance-perturbations such as
    1. Given the gt alpha matte, we first binarize it with a random threshold uniformly sampled from 0 to 1.
    2. Then, the mask is dilated and/or eroded in random order with random kernel sizes from 1 to 30.
    3. Also, internal guidance is perturbated for 2 levels (8, and 4).
    4. During training, the self-guidance mask from output stride 8 is dilated by K1 random sampled from [1, 30] and the one from output stride 4 is dilated by K2 from [1, 15]. For testing, we fix K1 = 15 and K2 = 7.

2. Used SAD, MSE, Grad and Connectivity as evaluation metrics following all previous methods.

3. Results on Adobe 1K Dataset, Distinction646

4. Also applied re-JEPGing, gaussian blur, and gaussian noises to the input image to make the model better adapt to real-world noises which are rarely seen in the synthetic dataset during training.

5. For real-world testing, produced own dataset of 637 use-cases with wide variety. [Human annotated]

Training cont.

Input **Existing method** **MG Matting**

- Besides predicting the alpha matte, the method also estimates the foreground color (*to allow compositing with non-opaque/transparent foreground objects, where only alpha matte is not enough*).
- The paper argues that training a single model for predicting both alpha matte and foreground color will degrade the matting performance.
- Whereas, decoupling will allow a flexible extension to the cases where alpha mattes are already given.
- The MG Matting adopts the popularly used ResNet34-UNet with an Atrous Spatial Pyramid Pooling (ASPP) as backbone for PRN and color estimation with same loss functions.

Results

Methods	SAD	MSE (10^{-3})	Grad	Conn
Learning Based Matting [46]	113.9	48	91.6	122.2
Closed-Form Matting [21]	168.1	91	126.9	167.9
KNN Matting [6]	175.4	103	124.1	176.4
Deep Image Matting [41]	50.4	14	31.0	50.8
IndexNet Matting [29]	45.8	13	25.9	43.7
AdaMatting [4]	41.7	10.2	16.9	-
Context-Aware Matting [18]	35.8	8.2	17.3	33.2
GCA Matting [24]	35.3	9.1	16.9	32.5
$\text{Ours}_{\text{TrimapFG}}$	**31.5**	**6.8**	**13.5**	**27.3**
$\text{Ours}_{\text{Trimap}}$	32.1	7.0	14.0	27.9

Table 1: Results on Composition-1k test set. The subscripts denote the corresponding guidance inputs, *i.e.* TrimapFG, Trimap. The other evaluated methods all require a trimap as input.

Methods	SAD	MSE (10^{-3})	Grad	Conn
Learning Based Matting* [46]	105.04	21	94.16	110.41
Closed-Form Matting* [21]	105.73	23	91.76	114.55
KNN Matting* [6]	116.68	25	103.15	121.45
Deep Image Matting* [41]	47.56	9	43.29	55.90
HAttMatting* [31]	48.98	9	41.57	49.93
Deep Image Matting [41]	48.73	11.2	42.60	49.55
+ Ours	**36.58**	**7.2**	**27.37**	**35.08**
IndexNet Matting [29]	46.95	9.4	40.56	46.80
+ Ours	**35.82**	**5.8**	**25.75**	**34.23**
Context-Aware Matting [18]	36.32	7.1	29.49	35.43
+ Ours	**35.04**	**5.4**	**24.55**	**33.35**
GCA Matting [24]	39.64	8.2	32.16	38.77
+ Ours	**35.93**	**5.7**	**25.94**	**34.35**

Table 2: Matting refinement results on Distinction-646 test set. Results with $*$ are from methods trained on Distinction-646 train set as reported in [31] for reference. Other results are only trained on composition-1k.

Results cont.

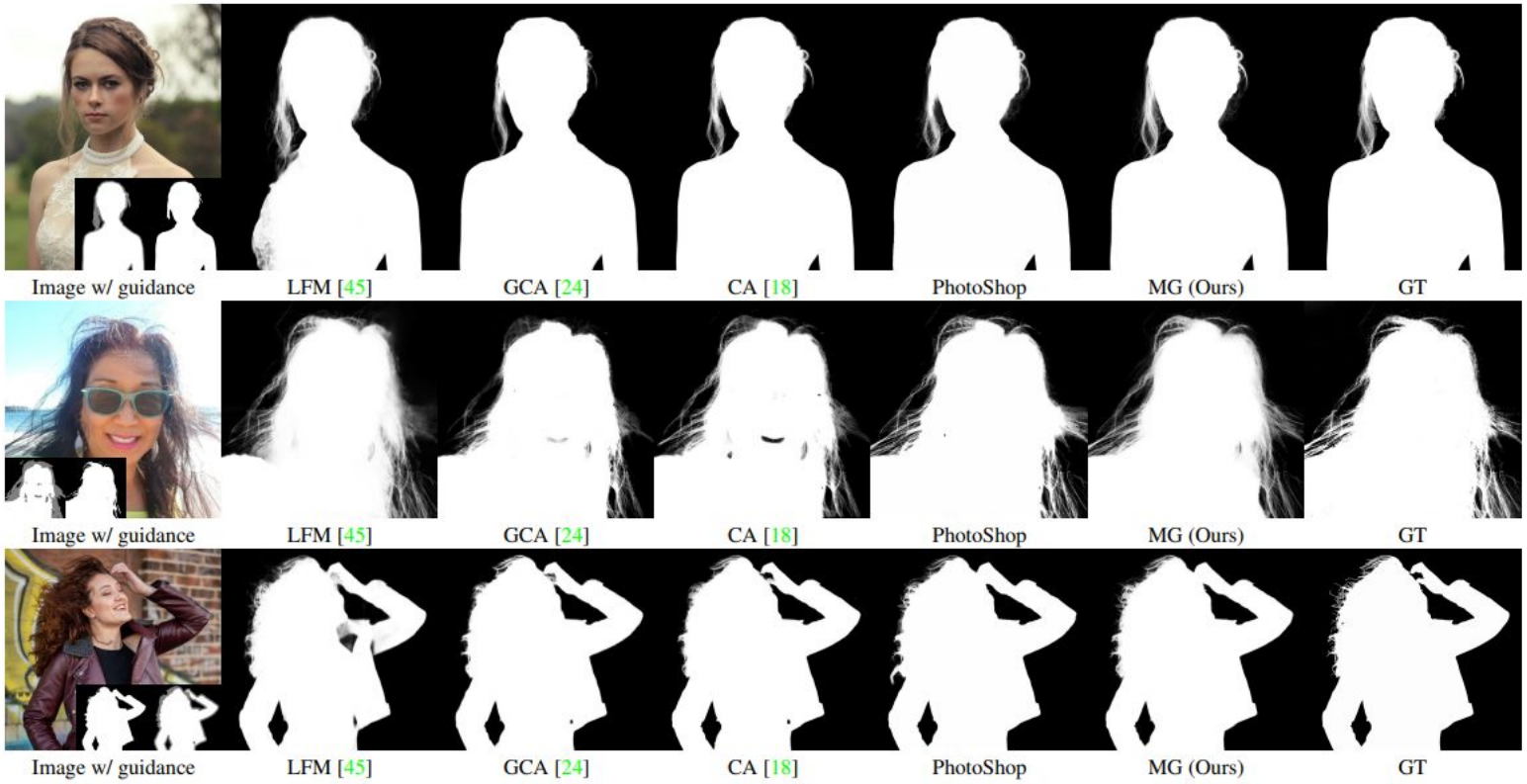


Figure 5: The visual comparison results among different methods on our portrait test set. We visualize representative examples with both high-quality studio-level portraits and selfies with strong noises. MG Mating performs well on different quality images and can maintain details. We note that our results, though only trained on composition-1k, are not only superior to previous state-of-the-art but also produces comparable or better results than commercial methods in PhotoShop.

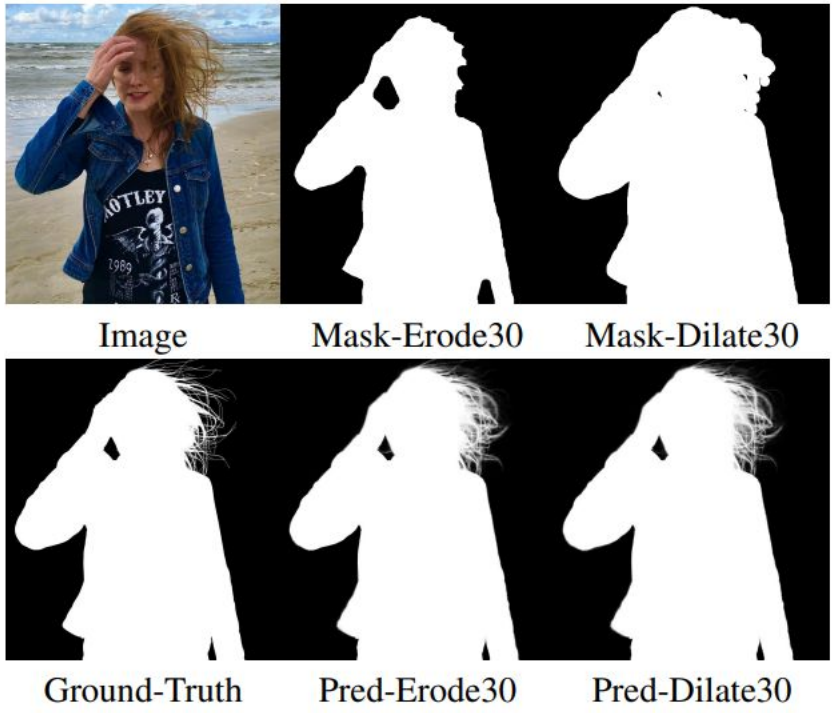


Figure 6: Our model is robust given different quality guidance masks and produces consistent alpha estimation.