Fast Lossless Compression with Integer-Only Discrete Flows

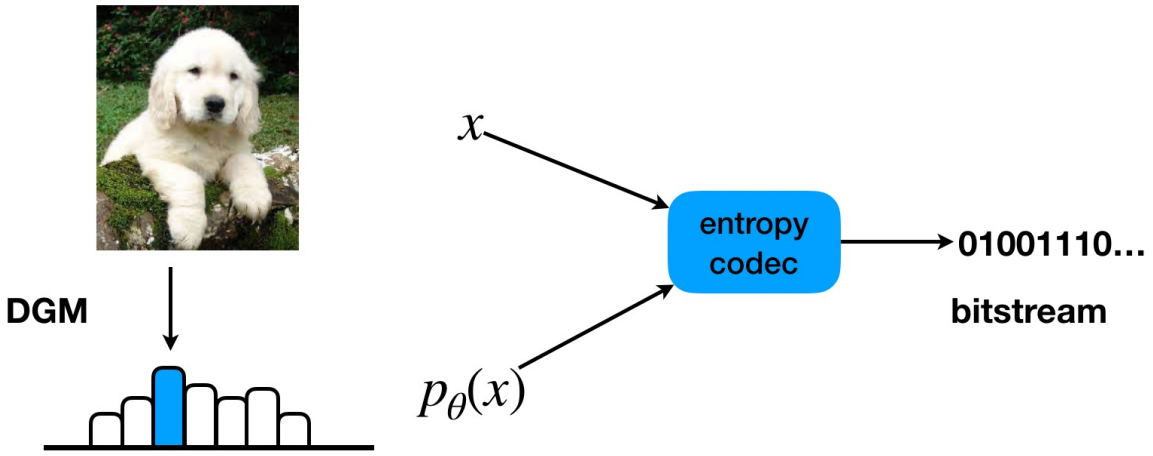
Siyu Wang[1], Jianfei Chen[1], Chongxuan Li[2], Jun Zhu[1], Bo Zhang[1]

[1]Tsinghua University; [2]Renmin University of China

Background

DGM For lossless compression

Neural Compressor: Probabilistic Model + Entropy Coding



High compression rate

Low encoding/decoding speed

Background

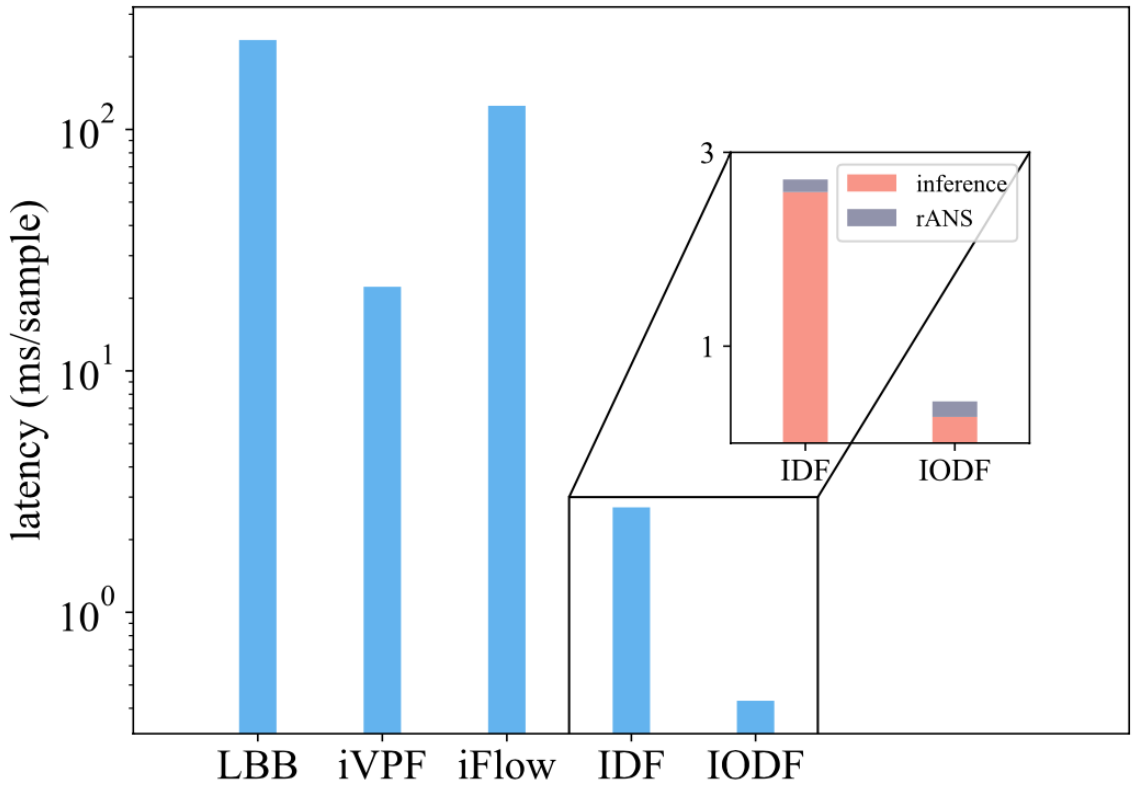
Integer discrete flows (IDF, *Hoogeboom et al.*)

Model data x and latent representation z both in discrete integer space:

$$\mathcal{X} = \mathcal{Z} = \mathbb{Z}^d$$

Design a bijective function $f_\theta(\cdot)$ between x and z. With change-of-variable formula,

$$\mathrm{P}_X(x) = P_Z\big(f_\theta(x)\big)$$



Method

Quantization

Hybrid format representation

Tensors are represented with a hybrid numerical format, which is done by a *quantizer* Q.

For a real-valued tensor $\mathbf{r}$, the quantizer outputs

$$\tilde{\mathbf{r}} = Q(\mathbf{r}) = s_{\mathbf{r}}\hat{\mathbf{r}} \approx \mathbf{r}$$

$\hat{\mathbf{r}}$: 8-bit integers in {-128,...,+127} or {0,...+255}.

$s_{\mathbf{r}}$: scalar, scale parameter.

Method

Quantization

Convolution $\mathbf{y} = \mathrm{Conv}(\mathbf{x}; \mathbf{W}, \mathbf{b})$

$\mathbf{W}$ is a $C \times D \times k \times k$ convolution kernel, $\mathbf{b}$ is a C-dimensional bias vector, $\mathbf{x}$ is a $D \times h \times w$ input tensor, and $\mathbf{y}$ is a $C \times h' \times w'$ output tensor.

$$y_c = \sum_{c'=1}^{D} W_{c,c'} \circledast x_{c'} + b_c, \quad c \in \{1, \ldots, C\}$$

Method

Quantization

Integer-only Convolution

Use hybrid format to represent $\mathbf{y}, \mathbf{x}, \mathbf{W}$ $\mathbf{y} \approx s_{\mathbf{y}}\hat{\mathbf{y}}, \mathbf{x} \approx s_{\mathbf{x}}\hat{\mathbf{x}}, \mathbf{W} \approx s_{\mathbf{W}}\hat{\mathbf{W}}.$

plug them into $$y_c = \sum_{c'=1}^{D} W_{c,c'} \circledast x_{c'} + b_c, \quad c \in \{1, \ldots, C\}$$

reorganize terms, we have

$$\hat{y}_c \approx \frac{s_{\mathbf{W}} s_{\mathbf{x}}}{s_{\mathbf{y}}} \sum_{c'=1}^{D} \hat{W}_{c,c'} \circledast \hat{x}_{c'} + \frac{b_c}{s_{\mathbf{y}}}.$$

Method

Quantization

$$\tilde{\mathbf{r}} = Q(\mathbf{r}) = s\hat{\mathbf{r}} = s \cdot \left\lfloor \text{clip}\left(\frac{\mathbf{r}}{s}, -Q_N, Q_P\right) \right\rceil$$

Learned Step-size Quantization (LSQ, *Esser et al.*)

Back-propogation through quantizer is performed with STE (*Bengio et al.*): $\partial \lfloor \mathbf{u} \rceil / \partial \mathbf{u} = \mathbf{I}$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{r}} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{r}}} \frac{\partial \tilde{\mathbf{r}}}{\partial \mathbf{r}} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{r}}}, \ \mathbf{r} = \mathbf{W} \ or \ \mathbf{x},$$

Gradient of scale parameter s

$$\frac{\partial \tilde{\mathbf{r}}}{\partial s} = \begin{cases} (-\mathbf{r}/s + \lfloor \mathbf{r}/s \rceil) \odot \mathbb{I}(-Q_N < \mathbf{r}/s < Q_P) \\ -Q_N \cdot \mathbb{I}(\mathbf{r}/s < -Q_N) \\ Q_P \cdot \mathbb{I}(\mathbf{r}/s > Q_P) \end{cases} \qquad \frac{\partial \mathcal{L}}{\partial s} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{r}}} \frac{\partial \tilde{\mathbf{r}}}{\partial s}$$

Method

Quantization

Deployment on GPU

After fine-tuning networks with fake quantizers, we deploy the model on GPU with TensorRT library and realize more efficient inference.

Improvement of network architecture

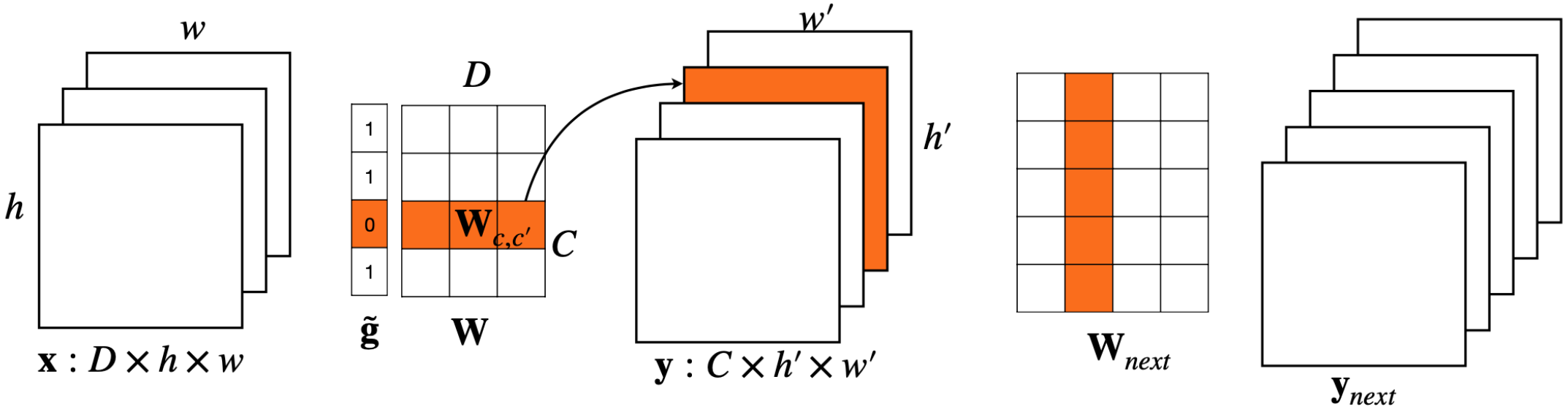
- Dense blocks v.s.residual blocks
- Optimization over residual blocks

Table 1. Latency of floating-point and integer-only inference for convolutions with varying number of input and output channels. Obtained by averaging over 1000 runs (milliseconds).

In Chn	Out Chn	Fp32	Int8	Speedup
128	128	0.040	0.0039	10.2×
64	256	0.035	0.0098	3.6×
32	512	0.037	0.0131	2.8×

Method

Pruning



Method

Pruning

Training binary-gated convolution

$$\mathcal{L}(\mathbf{X};\{\mathbf{W}\},\{\mathbf{g}\}) = \mathcal{L}_{IDF}(\mathbf{X};\{\mathbf{W}\},\{\mathbf{g}\}) + \sum \lambda \|\tilde{\mathbf{g}}\|_1,$$

$\mathcal{L}_{IDF}$ tends to remain all entries of $\tilde{\mathbf{g}}$ close to 1 to remain high performance.

$\|\tilde{\mathbf{g}}\|_1$ pushes the gates to be sparse.

Method

Training workflow

Algorithm 1 Training IODF

Input: r_{target}, remaining target proportion of FLOPs and $\mathbf{X}$, the training dataset.

#Stage 1:

$\mathbf{W} \longleftarrow$ InitializeParameter()

Train $\mathcal{L}_{IDF}(\mathbf{X}; \{\mathbf{W}\}, \{\mathbf{1}\})$ to convergence

$F_0 \longleftarrow$ CalculateFLOPs($\mathbf{W}$)

#Stage 2:

$\mathbf{g} \longleftarrow \alpha\mathbf{1}$, $\lambda \longleftarrow$ InitializeLambda()

Train $\mathcal{L}(\mathbf{X}; \{\mathbf{W}\}, \{\mathbf{g}\})$ until CalculateFLOPs($\mathbf{W}, \mathbf{g}$)$<$ $r_{target}F_0$

#Stage 3:

Fine-tune $\mathcal{L}_{IDF}(\mathbf{X}; \{\mathbf{W}\}, \mathbf{g})$ with fixed $\mathbf{g}$

#Stage 4:

Fine-tune the model with fake quantization applied to activations

#Stage 5:

Fine-tune the model with fake quantization applied to activations and weights

	Analytic bpd	Coding bpd	Inference Latency 4	8	16	32	64	Compression Bandwidth 4	8	16	32	64
ImageNet32												
IDF-Dense	3.890	3.900	8.38	5.08	4.08	*	*	0.31	0.54	0.70	*	*
IDF-Res	3.916	3.926	4.19	3.19	3.59	2.54	2.93	0.56	0.79	0.79	1.12	1.00
8bit IDF-Dense	3.911	3.921	5.38	2.90	1.74	1.20	0.99	0.46	0.86	1.21	2.18	2.67
8bit IDF-Res	3.923	3.934	2.08	1.09	0.64	0.44	0.36	0.91	1.78	2.96	4.76	5.98
Pruned IDF-Res	3.936	3.947	3.27	2.04	1.60	1.33	1.29	0.72	1.14	1.59	2.01	2.15
IODF	3.968	3.979	1.79	0.94	0.54	0.34	0.27	0.98	1.91	3.45	5.41	7.08
Speedup	-	-	4.7×	5.4×	7.6×	*	*	3.2×	3.6×	4.9×	*	*
ImageNet64												
IDF-Dense	3.635	3.638	18.65	15.45	13.93	*	*	0.59	0.73	0.83	*	*
IDF-Res	3.637	3.640	12.50	11.89	9.30	8.84	8.64	0.82	0.93	1.22	1.32	1.35
8bit IDF-Dense	3.663	3.666	8.98	5.57	4.35	3.67	3.34	1.02	1.66	2.32	2.83	3.11
8bit IDF-Res	3.663	3.673	3.03	2.02	1.61	1.41	1.31	1.83	3.47	4.72	5.57	6.83
Pruned IDF-Res	3.657	3.666	7.75	6.45	6.55	5.79	5.71	1.21	1.59	1.70	1.93	2.00
IODF	3.685	3.695	2.79	1.71	1.34	1.15	1.06	2.22	3.72	4.64	7.03	7.93
Speedup	-	-	6.9×	9.0×	10.4×	*	*	3.8×	5.1×	5.6×	*	*

Reference

Duda, J. Asymmetric numeral systems. arXiv preprint arXiv:0902.0271, 2009.

Duda, J. Asymmetric numeral systems: entropy coding com- bining speed of huffman coding with compression rate of arithmetic coding. arXiv preprint arXiv:1311.2540, 2013.

Ho, J., Lohn, E., and Abbeel, P. Compression with flows via local bits-back coding. arXiv preprint arXiv:1905.08500, 2019.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion proba- bilistic models. arXiv preprint arXiv:2006.11239, 2020.

Hoogeboom, E., Peters, J. W., Berg, R. v. d., and Welling, M. Integer discrete flows and lossless compression. arXiv preprint arXiv:1905.07376, 2019.

Huffman, D. A. A method for the construction of minimum- redundancy codes. Proceedings of the IRE, 40(9):1098– 1101, 1952.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. arXiv preprint arXiv:1807.03039, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In International Conference on Learning Repre- sentations, 2013.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. Varia- tional diffusion models. arXiv preprint arXiv:2107.00630, 2, 2021.

NVIDIA. Tensorrt. https://developer.nvidia. com/tensorrt, 2018.

Shannon, C. E. A mathematical theory of communication. The Bell system technical journal, 27(3):379–423, 1948.

Zhang, S., Kang, N., Ryder, T., and Li, Z. iflow: Numeri- cally invertible flows for efficient lossless compression via a uniform coder. Advances in Neural Information Processing Systems, 34, 2021b.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., & Lakshminarayanan, B. (2018). Do deep generative models know what they don't know?. arXiv preprint arXiv:1810.09136.

Zhang, S., Zhang, C., Kang, N., and Li, Z. ivpf: Numerical invertible volume preserving flow for efficient lossless compression. In Proceedings of the IEEE/CVF Confer- ence on Computer Vision and Pattern Recognition, pp. 620–629, 2021c.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In International Conference on Machine Learning, pages 2256–2265, 2015.

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." arXiv preprint arXiv:1605.08803 (2016).

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. Predicting structured data, 1(0), 2006.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. arXiv preprint arXiv:2010.02502, 2020a.

Bao F, Li C, Zhu J, et al. Analytic-DPM: an Analytic Estimate of the Optimal Reverse Variance in Diffusion Probabilistic Models[J]. arXiv preprint arXiv:2201.06503, 2022.

Zhang M, Zhang A, McDonagh S. On the out-of-distribution generalization of probabilistic image modelling[J]. Advances in Neural Information Processing Systems, 2021, 34.

Thanks for listening.