Recent Advances in Generative Models

Zhenguo Li

Huawei Noah's Ark Lab

Nov 4, 2023

Generative Models: why do we care ?

- A full, joint probability distribution aims to model all dependencies within high-dimensional data.

- It enables many applications
  - Compression, storage, and transmission (telecommunication/5.5G)
  - Sampling, generation, and editing (AIGC)
  - Inference, reasoning, and discovery (AI for Math/Science)

- Generative models: flows, VAE, GAN, autoregressive (GPT), diffusion

Outline

- Neural Compression
- Text-to-image Generation
- Neural Theorem Proving

Outline

- Neural Compression
- Text-to-image Generation
- Neural Theorem Proving

iFlow: Numerically Invertible Flows for Efficient Lossless Compression via a Uniform Coder

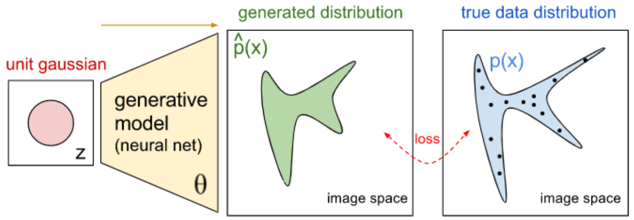
NeurIPS 2021 Spotlight, Huawei Noah Ark's Lab

AI for Lossless Compression

Input $\xrightarrow{x}$ ★ Probabilistic model $\xrightarrow{P(x)}$ ★ Entropy Coder $\xrightarrow{c}$ Encoded bits

Predict $P(x)$ with probabilistic model

- Maximize data likelihood $E_{P(x)}[\log_2 \hat{P}(x)]$
- Minimize code length $E_{P(x)}[-\log_2 \hat{P}(x)]$



Higher $P(x)$ => shorter codes

- Shannon Theorem (optimal codelength)

$$E_{P(x)}[-\log_2 P(x)] = H(X)$$

- Expected codelength

$$E_{P(x)}[-\log_2 \hat{P}(x)] = -\sum_x P(x)\log_2 \hat{P}(x) = H(X) + KL(P||\hat{P})$$

- Better $\hat{P}$ => higher compression ratio
- Entropy coders: AC/ANS.

	True prob	Prob with traditional	Prob with AI
A	0.8	0.5	0.7
B	0.05	0.15	0.05
C	0.01	0.1	0.05
D	0.14	0.25	0.2
length	0.94	1.25	1.00

Asymmetrize binary system for $p(0) = \frac{1}{7}, p(1) = \frac{6}{7}$

	x'	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
x	$s=0$	0							1							2					
	$s=1$		0	1	2	3	4	5		6	7	8	9	10	11		12	13	14	15	16

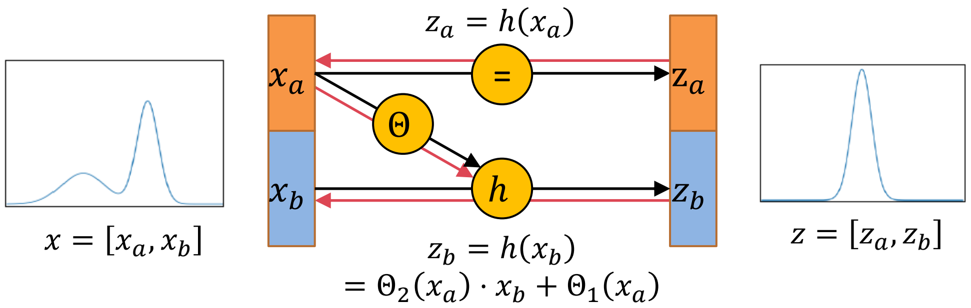
e.g. $x = 1 \xrightarrow{s=0} 7 \xrightarrow{s=1} 9 \xrightarrow{s=1} 11 \xrightarrow{s=1} 13 \xrightarrow{s=1} 16$

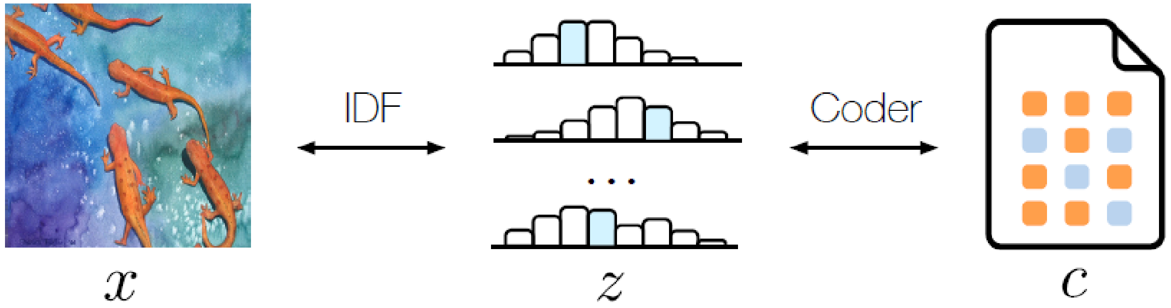
$x' \approx x/p(s)$. Expected codelength

$\log x' - \log x = -\log p(s)$

Lossless Compression with Flow Model

- Flow model
  - Invertible neural network: $f: x \to z$; $f^{-1}: z \to x$
  - Probability mass: $p_X(x) = p_Z(z)\left|\frac{dz}{dx}\right|$
- Lossless compression with flows
  - Compression: convert x to $z = f(x)$, compress z with $p_Z(z)$
  - Decompression: decode z with $p_Z(z)$, recover x with $x = f^{-1}(z)$



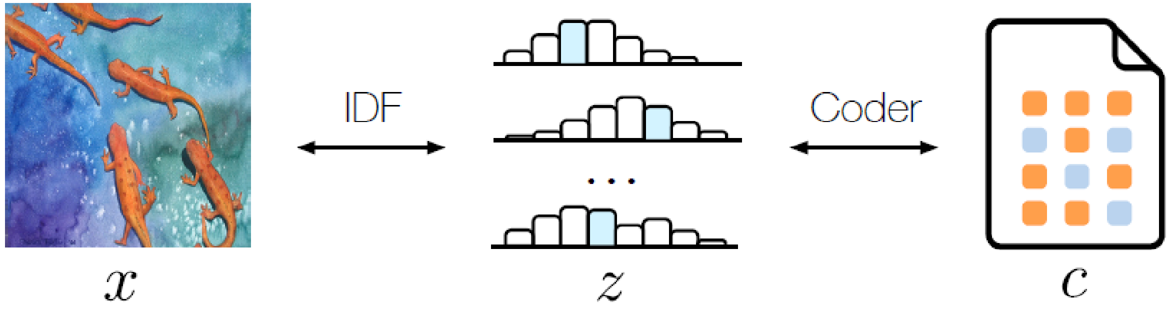


- Advantages
  - Accurate density estimation
  - High compression ratio

	ImageNet32	ImageNet64	CIFAR10
PNG [5]	6.39	5.71	5.87
FLIF [35]	4.52	4.19	4.19
JPEG-XL [2]	6.39	5.74	5.89
L3C [29]	4.76	4.42	-
RC [30]	-	-	-
Bit-Swap [25]	4.50	-	3.82
IDF [18]	4.18	3.90	3.34
IDF++ [4]	4.12	3.81	3.26
iVPF [40]	4.03	3.75	3.20
LBB [17]	**3.88**	**3.70**	**3.12**
iFlow (Ours)	**3.88**	**3.70**	**3.12**

Lossless Compression with Flow Model

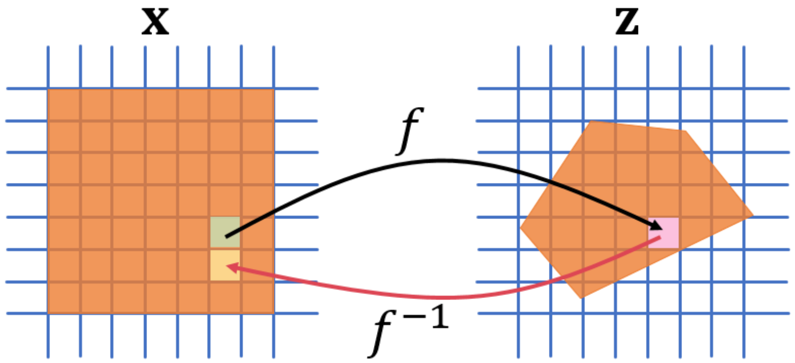
- Flow model
  - Invertible neural network: $f: x \to z; f^{-1}: z \to x$
  - Probability mass: $p_X(x) = p_Z(z)\left|\frac{dz}{dx}\right|$
- Lossless compression with flows
  - Compression: convert x to $z = f(x)$, compress z with $p_Z(z)$
  - Decompression: decode z with $p_Z(z)$, recover x with $x = f^{-1}(z)$



- Challenges: numerical errors
  - Data must be discrete

75	17	61	6	119	97	121	...	62	8

  - Flow models are usually not invertible due to numerical error

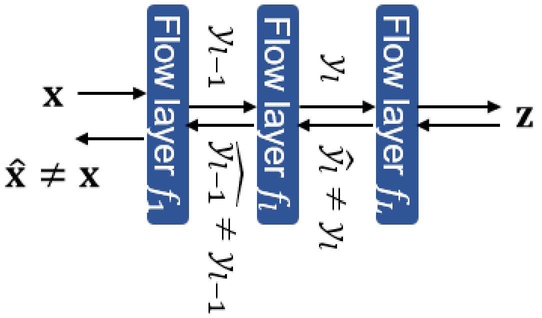



```
x = 9; s = 0.6
z = round(s * x)
print (round(z / s) == x)
```

```
False
```

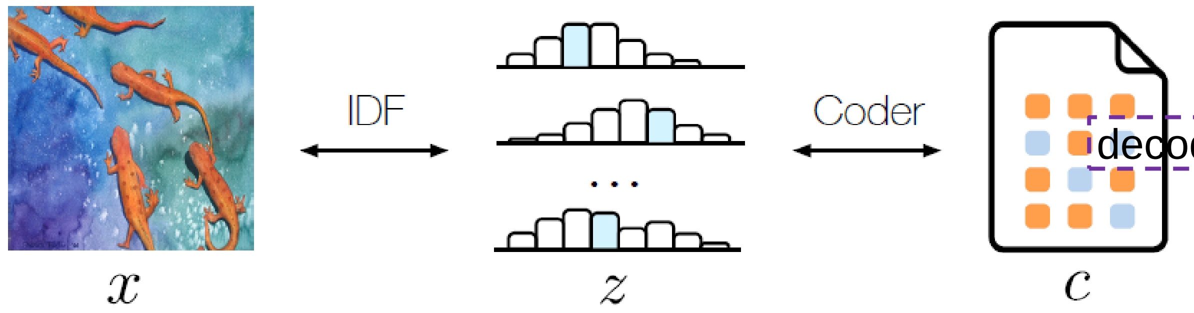
```
x = 0.9; s = 0.6
z = s * x
print (z / s == x)
```

```
False
```



Lossless Compression with Flow Model

- Flow model
  - Invertible neural network: $f: x \to z; f^{-1}: z \to x$
  - Probability mass: $p_X(x) = p_Z(z)\left|\frac{dz}{dx}\right|$
- Lossless compression with flows
  - Compression: convert x to $z = f(x)$, compress z with $p_Z(z)$
  - Decompression: decode z with $p_Z(z)$, recover x with $x = f^{-1}(z)$

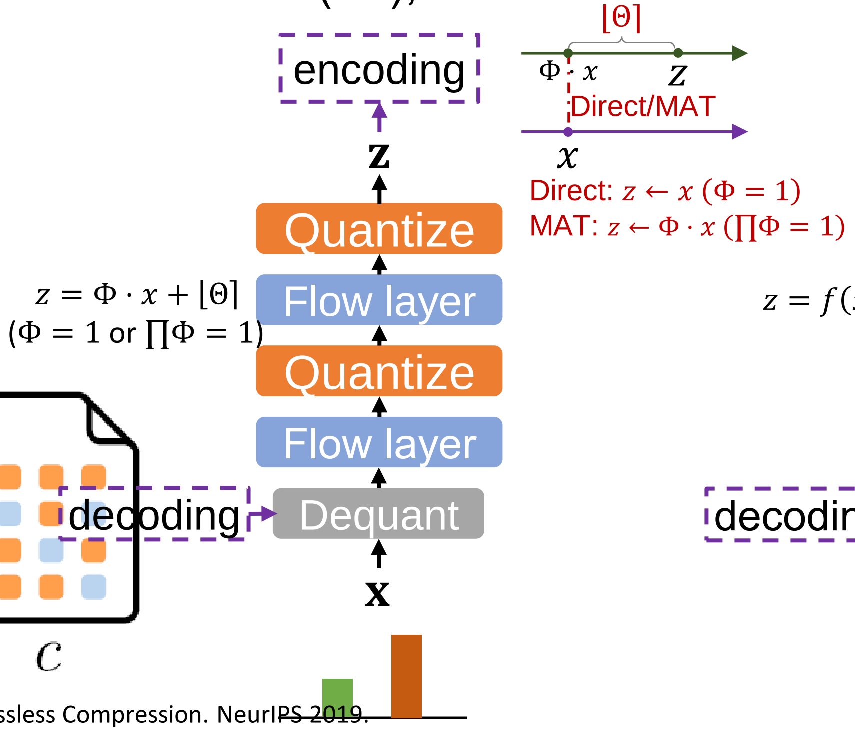


- Related work

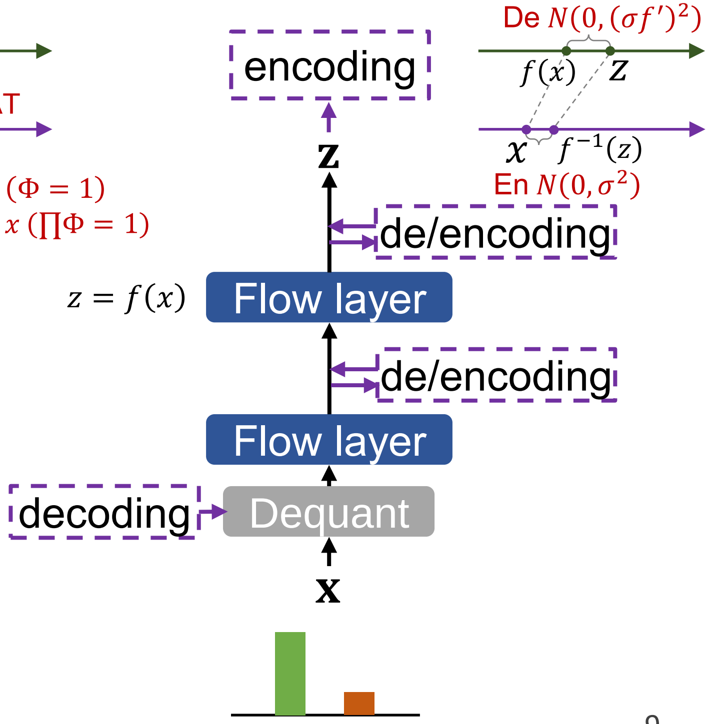
IDF(++) (NeurIPS 2019, ICLR 2021), iVPF (CVPR 2021)
- Invertible operations in integer flow model
- Inferior expressive power

LBB (NeurIPS 2019)
- Any flow models with high compression ratio
- Encoding the numerical error is slow

IDF(++), iVPF



LBB



Emiel Hoogeboom, Jorn W. T. Peters, Rianne van den Berg, Max Welling. Integer Discrete Flows and Lossless Compression. NeurIPS 2019.
Jonathan Ho, Evan Lohn, Pieter Abbeel. Compression with Flows via Local Bits-Back Coding. NeurIPS 2019.

Dynamic Entropy Coders

- AI model captures all dependencies within data: $p(x_1, x_2) = p(x_1)p(x_2|x_1)$
- Traditional models often use the same distribution for each dimension
- Dynamic entropy coder should be introduced in AI compression

- Related work: rANS. Coding with PMF l_s/m and CDF b_s/m

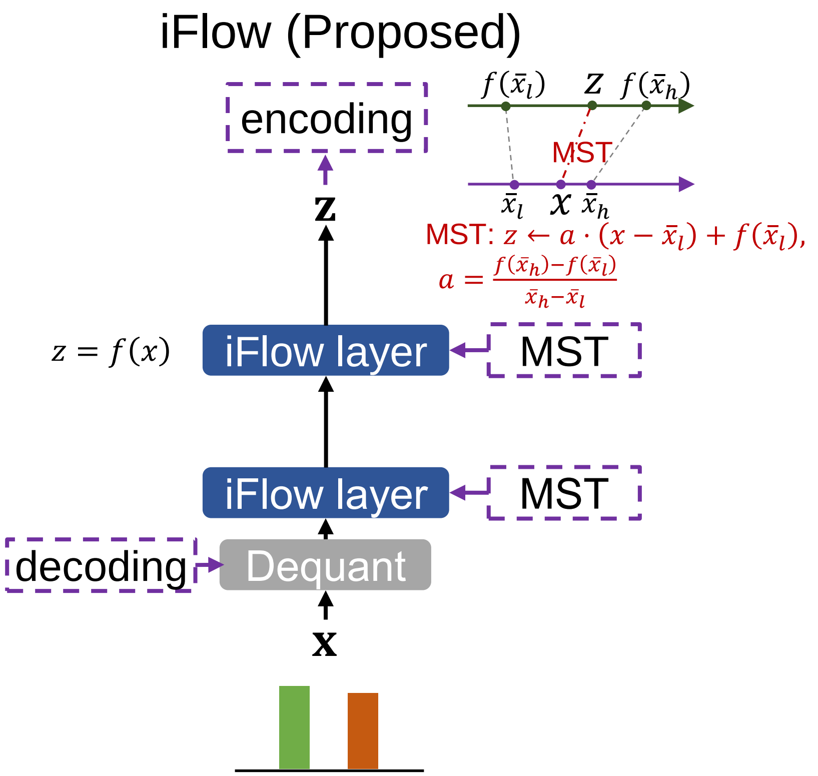
$$c'(c, s) = \lfloor c/l_s \rfloor \cdot m + (c \mod l_s) + b_s \qquad c(c', s) = \lfloor c'/m \rfloor \cdot l_s + (c' \mod m) - b_s$$

- Drawbacks: low compression bandwidth
  - Many atomic operations
  - Binary search in decoding

	# threads	rANS
Encoder	1	$5.1_{\pm 0.3}$
	4	$10.8_{\pm 1.9}$
	8	$15.9_{\pm 1.4}$
	16	$21.6_{\pm 1.1}$
Decoder	1	$0.80_{\pm 0.02}$
	4	$2.8_{\pm 0.1}$
	8	$5.5_{\pm 0.2}$
	16	$7.4_{\pm 0.5}$

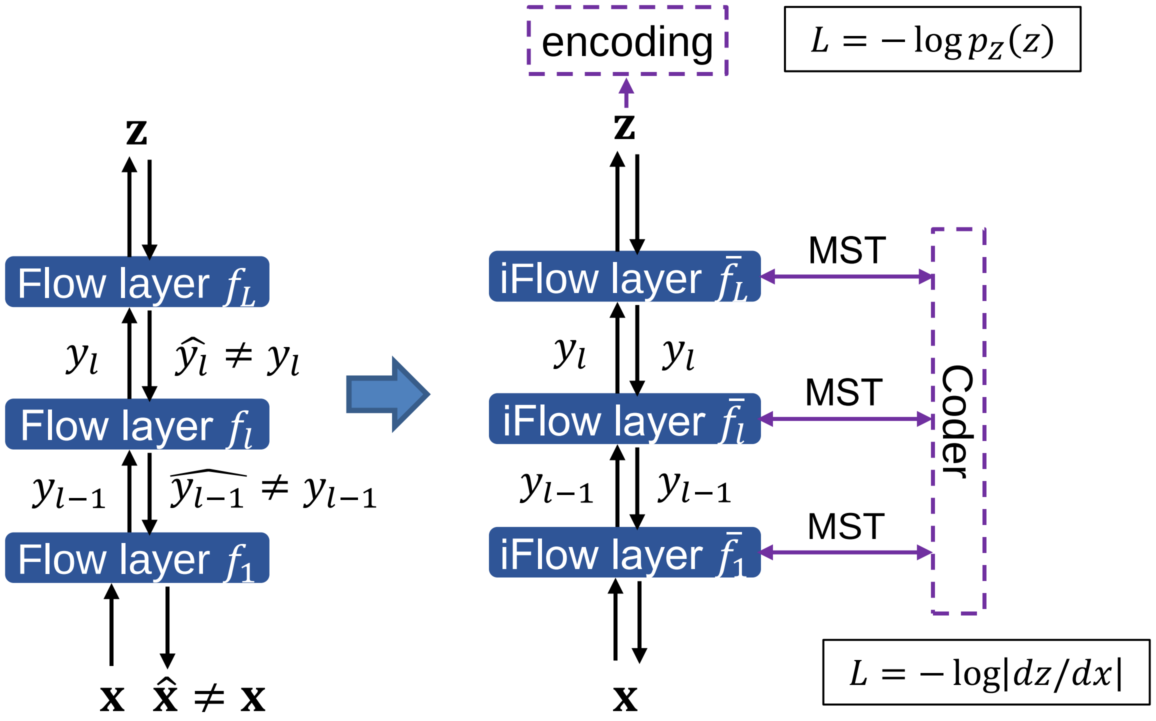
iFlow: Contributions

- Numerically Invertible Flows (iFLow)
  - MST: the fast and efficient numerically invertible flows with bits-back coding
- Dynamic Entropy Coders
  - UBCS: efficient dynamic entropy coder on uniform distribution for fast computation of iFlow
- Lossless compression with iFlow
  - Coding with ANY types of flows

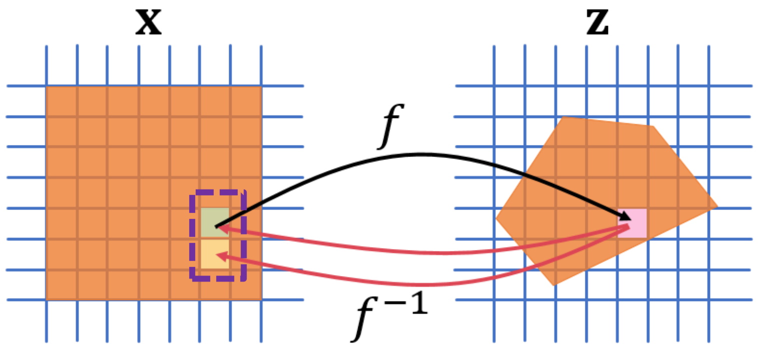


iFlow: Numerically Invertible Flows

- iFlow pipeline
  - Flow f: stacking flow layers $f = f_L \circ \cdots \circ f_1$
  - iFlow $\bar{f} = \bar{f}_L \circ \cdots \circ \bar{f}_1$: each layer is numerically invertible $y_{l-1} = \bar{f}_l^{-1}\left(\bar{f}_l(y_{l-1})\right)$
  - Inputs/outputs of each layer is k-precision quantization: $y \leftarrow \lfloor 2^k \cdot y \rceil / 2^k$



- Coder may be involved in iFlow
  - One discrete z may correspond to multiple x's
  - Code for duplicate positions
  - $\boxed{z = \bar{f}(x) \text{ with } -\log \bar{f}'(x) \text{ bits encoded}}$



iFlow: Numerically Invertible Flows

- Numerically invertible linear flows
  - $z = f(x) = a \cdot x, a \to R/S$。MST algorithm

Codelength: $L_f(x) = \log S - \log R = -\log f'(x)$

$$y \underset{y = z \cdot S + r_e}{\overset{y \div S = z \bmod r_e}{\rightleftarrows}} (z, r_e)\ (r_e < S)$$

$$\overbrace{(x, r_d)\ (r_d < R) \underset{\div R}{\overset{\times R + r_d}{\rightleftarrows}} y \underset{\times S + r_e}{\overset{\div S}{\rightleftarrows}} (z, r_e)}^{z \approx R/S \cdot x \approx a \cdot x}\ (r_e < S)$$

Decode r_d with $U(0, R)$ ↑ Coder

Encode r_e with $U(0, S)$ ↓ Coder

Algorithm 1 Modular Scale Transform (MST): Numerically Invertible Scale Flow $f(x) = R/S \cdot x$.

Forward MST: $\bar{z} = \bar{f}(\bar{x})$.

$\hat{x} \leftarrow 2^k \cdot \bar{x}$;
Decode r_d from $U(0, R)$; $\hat{y} \leftarrow R \cdot \hat{x} + r_d$;
$\hat{z} \leftarrow \lfloor \hat{y}/S \rfloor, r_e \leftarrow \hat{y} \mod S$;
Encode r_e with $U(0, S)$;
return $\bar{z} \leftarrow \hat{z}/2^k$.

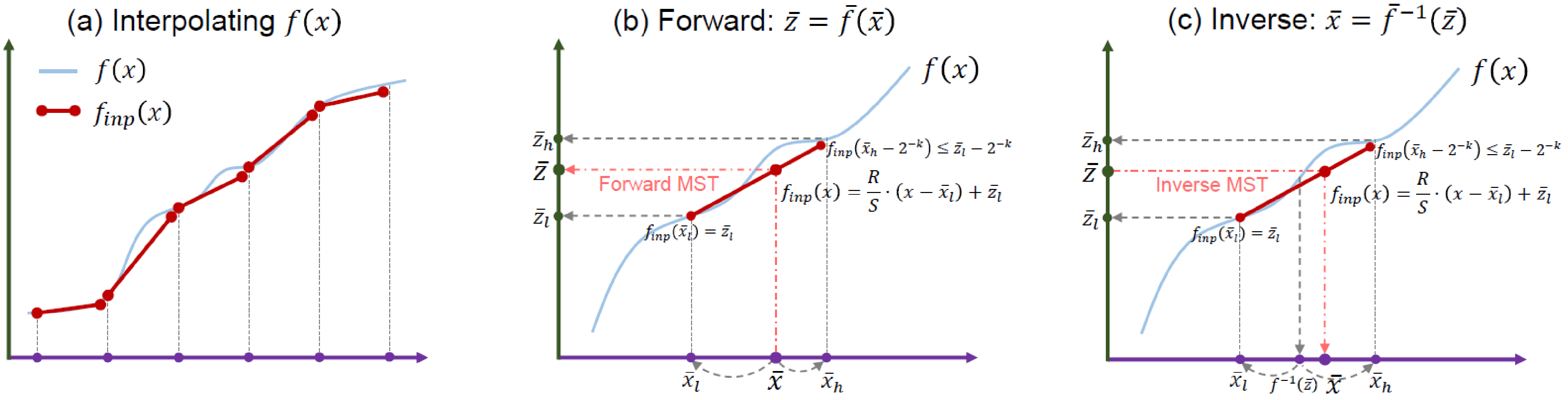
Inverse MST: $\bar{x} = \bar{f}^{-1}(\bar{z})$.

$\hat{z} \leftarrow 2^k \cdot \bar{z}$;
Decode r_e from $U(0, S)$; $\hat{y} \leftarrow S \cdot \hat{z} + r_e$;
$\hat{x} \leftarrow \lfloor \hat{y}/R \rfloor, r_d \leftarrow \hat{y} \mod R$;
Encode r_d with $U(0, R)$;
return $\bar{x} \leftarrow \hat{x}/2^k$.

iFlow: Numerically Invertible Flows

- Numerically invertible non-linear flows
  - Interpolating f, use MST on each interval

Codelength: $L_f(x) = -\log R/S \approx -\log f'(x)$



iFlow: Numerically Invertible Flows

- Numerically invertible coupling layer

  Codelength: $L_f(x) = -\log f'(x_b) = -\log dz/dx$

  – $z = [z_a, z_b] = [x_a, f(x_b)]$。Use non-linear iFlow layer in $z_b = f(x_b)$
- Numerically invertible 1x1 convolutional layer

  Codelength: $L_f(x) = -\det \Lambda = -\log dz/dx$

Upper triangular $\boldsymbol{U}$: Iterative computation

Diagonal $\boldsymbol{\Lambda}$: $\boldsymbol{z} = \boldsymbol{\lambda} \cdot \boldsymbol{x}$: use MST

Lower triangular $\boldsymbol{L}$: Iterative computation

Permutation $\boldsymbol{P}$: permute elements

$$\boldsymbol{x} \rightleftarrows \boldsymbol{Ux} \rightleftarrows \boldsymbol{\Lambda Ux} \rightleftarrows \boldsymbol{L\Lambda Ux} \rightleftarrows \boldsymbol{z} = \boldsymbol{PL\Lambda Ux}$$

$\boldsymbol{U}^{-1}$: Iterative computation

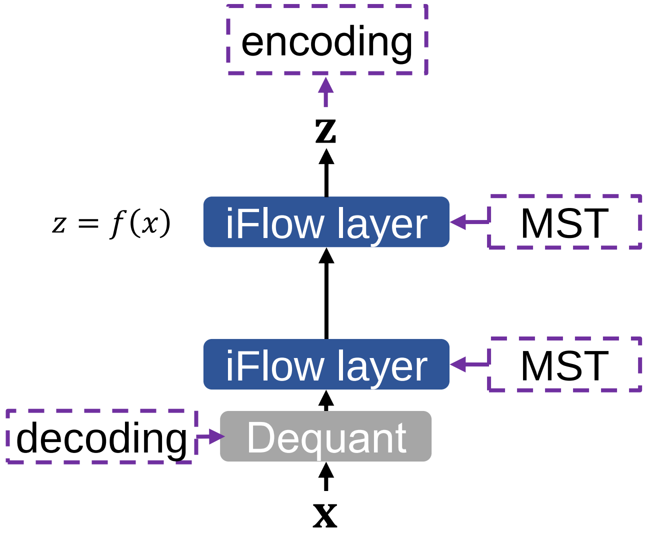
$\boldsymbol{\Lambda}^{-1}$: $\boldsymbol{z} = \boldsymbol{x}/\boldsymbol{\lambda}$, use inverse MST

$\boldsymbol{L}^{-1}$: Iterative computation

$\boldsymbol{P}^{-1}$: permute elements inversely

- Lossless compression with iFlow
  – Construct iFlow with iFlow layers
  – Bits-back coding with dequantization

Codelength: $L_f(x) =$
$-\log p(z) - \sum_l L_{f_l}(y_{l-1}) = -\log p(x)$



UBCS: Fast Dynamic Uniform Coder

- Related work: Range-based Asymmetric Numerical System (rANS)

$$c'(c,s) = \lfloor c/l_s \rfloor \cdot m + (c \mod l_s) + b_s \quad \Big| \quad c(c',s) = \lfloor c'/m \rfloor \cdot l_s + (c' \mod m) - b_s$$

- Proposed: Uniform Base Conversion System (UBCS)
  - Coding with any uniform distribution: $P(s) = \frac{1}{R}, s \in \{0,1,\ldots,R-1\}$

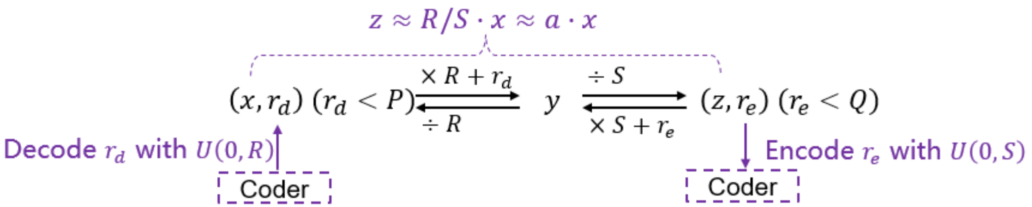
$$c' = E(c,s) = c \cdot R + s \quad \Big| \quad s = c' \mod R, \qquad c = D(c',s) = \lfloor \frac{c'}{R} \rfloor$$

- Advantages

	rANS	UBCS
Encoding bandwidth	21.6 MB/s	2075 MB/s
Decoding bandwidth	7.4 MB/s	552 MB/s
Encoding process	One division, one mod, one multiplication, two additions	One multiplication, one addition
Decoding process	Find s with binary search, one shift operation, one or operation, two additions	One division, one mod

iFlow: Numerically Invertible Flows

- Discussions
  - Coding with ANY flow: high compression ratio
  - Fast uniform coder in MST: high bandwidth

<table>
<tr><th>iFlow</th><th>LBB</th></tr>
<tr><td>
</td><td>Decode $\bar{\mathbf{z}} \sim \mathcal{N}(f(\bar{\mathbf{x}}), \sigma^2 \mathbf{J}\mathbf{J}^\top)\, \delta_z$
Encode $\bar{\mathbf{x}}$ using $\mathcal{N}(f^{-1}(\bar{\mathbf{z}}), \sigma^2 \mathbf{I})\, \delta_x$
Encode $\bar{\mathbf{z}}$ using $p(\bar{\mathbf{z}})\, \delta_z$</td></tr>
<tr><td>Code Uniform distribution with UBCS</td><td>Code Gaussian distribution with rANS</td></tr>
<tr><td>FAST</td><td>SLOW</td></tr>
</table>

Experiments: Coding Bandwidth

- UBCS: achieving high compression bandwidth
  - 50x speedup compared with rANS, achieving 2GB/s

	# threads	rANS	UBCS
Encoder	1	$5.1_{\pm 0.3}$	$\mathbf{380_{\pm 5}}$
	4	$10.8_{\pm 1.9}$	$\mathbf{709_{\pm 56}}$
	8	$15.9_{\pm 1.4}$	$\mathbf{1297_{\pm 137}}$
	16	$21.6_{\pm 1.1}$	$\mathbf{2075_{\pm 353}}$
Decoder	1	$0.80_{\pm 0.02}$	$\mathbf{66.2_{\pm 1.7}}$
	4	$2.8_{\pm 0.1}$	$\mathbf{248_{\pm 8}}$
	8	$5.5_{\pm 0.2}$	$\mathbf{460_{\pm 16}}$
	16	$7.4_{\pm 0.5}$	$\mathbf{552_{\pm 50}}$

Experiments: Lossless Compression

- iFlow achieves SoTA compression ratio and bandwidth
  - The compression ratio achieves the theoretical upper bound
  - Coding time only occupies 30% of the model inference time, which is no longer the bottleneck for lossless compression
  - Coding bandwidth is 5x faster than LBB (as the coding time is 5x compared with LBB)
  - Coding bandwidth is 30% faster than iVPF (as UBCS performs faster than MAT in iVPF)

flow arch.	compression technique	nll	bpd	aux. bits	encoding time (ms) inference	encoding time (ms) coding	decoding time (ms) inference	decoding time (ms) coding
Flow++	LBB [17]	3.116	**3.118**	39.86	16.2±0.3	116±1.0	32.4±0.2	112±1.5
	iFlow (Ours)		**3.118**	**34.28**		**21.0**±0.5		**37.7**±0.5
iVPF	iVPF [40]	3.195	3.201	**6.00**	5.5±0.1	11.4±0.2	5.2±0.1	13.5±0.3
	iFlow (Ours)		**3.196**	7.00		**7.1**±0.2		**9.7**±0.2

Shifeng Zhang, Chen Zhang, Ning Kang, Zhenguo Li. iVPF: Numerical Invertible Volume Preserving Flow for Efficient Lossless Compression. CVPR 2021

Experiments: Lossless Compression

- Achieving SoTA on benchmarking image datasets

	ImageNet32	ImageNet64	CIFAR10	CLIC.mobile	CLIC.pro	DIV2K
PNG [5]	6.39	5.71	5.87	3.90	4.00	3.09
FLIF [35]	4.52	4.19	4.19	2.49	2.78	2.91
JPEG-XL [2]	6.39	5.74	5.89	2.36	2.63	2.79
L3C [29]	4.76	4.42	-	2.64	2.94	3.09
RC [30]	-	-	-	2.54	2.93	3.08
Bit-Swap [25]	4.50	-	3.82	-	-	-
IDF [18]	4.18	3.90	3.34	-	-	-
IDF++ [4]	4.12	3.81	3.26	-	-	-
iVPF [40]	4.03	3.75	3.20	-	-	-
LBB [17]	**3.88**	**3.70**	**3.12**	-	-	-
iFlow (Ours)	**3.88**	**3.70**	**3.12**	-	-	-
HiLLoC [36]†	4.20	3.90	3.56	-	-	-
IDF [18]†	4.18	3.94	3.60	-	-	-
iVPF† [40]	4.03	3.79	3.49	2.47/2.39‡	2.63/2.54‡	2.77/2.68‡
iFlow (Ours)†	**3.88**	**3.65**	**3.36**	**2.26/2.26‡**	**2.45/2.44‡**	**2.60/2.57‡**

Experiments: Lossless Compression

- Achieving good generalization performance: SoTA compression ratio on real-world high resolution images
  - Train flow with Imagenet32/64 dataset
  - Crop the image to 32x32/64x64 patches

	ImageNet32	ImageNet64	CIFAR10	CLIC.mobile	CLIC.pro	DIV2K
PNG [5]	6.39	5.71	5.87	3.90	4.00	3.09
FLIF [35]	4.52	4.19	4.19	2.49	2.78	2.91
JPEG-XL [2]	6.39	5.74	5.89	2.36	2.63	2.79
L3C [29]	4.76	4.42	-	2.64	2.94	3.09
RC [30]	-	-	-	2.54	2.93	3.08
Bit-Swap [25]	4.50	-	3.82	-	-	-
IDF [18]	4.18	3.90	3.34	-	-	-
IDF++ [4]	4.12	3.81	3.26	-	-	-
iVPF [40]	4.03	3.75	3.20	-	-	-
LBB [17]	**3.88**	**3.70**	**3.12**	-	-	-
iFlow (Ours)	**3.88**	**3.70**	**3.12**	-	-	-
HiLLoC [36]†	4.20	3.90	3.56	-	-	-
IDF [18]†	4.18	3.94	3.60	-	-	-
iVPF† [40]	4.03	3.79	3.49	2.47/2.39‡	2.63/2.54‡	2.77/2.68‡
iFlow (Ours)†	**3.88**	**3.65**	**3.36**	**2.26/2.26‡**	**2.45/2.44‡**	**2.60/2.57‡**

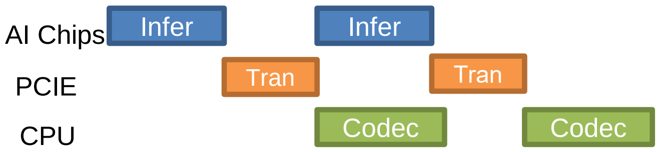
PILC: Practical Image Lossless Compression with an End-to-end GPU Oriented Neural Framework

CVPR 2022, Huawei Noah's Ark Lab

PILC: >100 MB/s AI Lossless Compression

Type	Performance	# inference	# transfer	Entropy coder
Auto-Regressive		Deep model; 1 network inference per symbol	1 transfer per symbol	Special entropy coder required: 1. Dynamic 2. Distribution calculated for each symbol
AE	Inferior compression ratio	Deep model required	1 transfer per latent layer	

- Principles for building real-time AI lossless codecs
  - Inference time of AI model should be small
    - Auto-regressive models achieve better compression ratio with smaller parameters
    - AE models is faster and able to model global information
  - AI codecs should not suffer from bandwidth issues
  - PCIE transfer between AI chips and CPU should be reduced



AI Chips for inference + CPU codec

AI Chips for inference & codec

PILC: >100 MB/s AI Lossless Compression

Type	Performance	# inference	# transfer	Entropy coder
Auto-Regressive		Deep model; 1 network inference per symbol	1 transfer per symbol	Special entropy coder required: 1. Dynamic 2. Distribution calculated for each symbol
AE	Inferior compression ratio	Deep model required	1 transfer per latent layer	

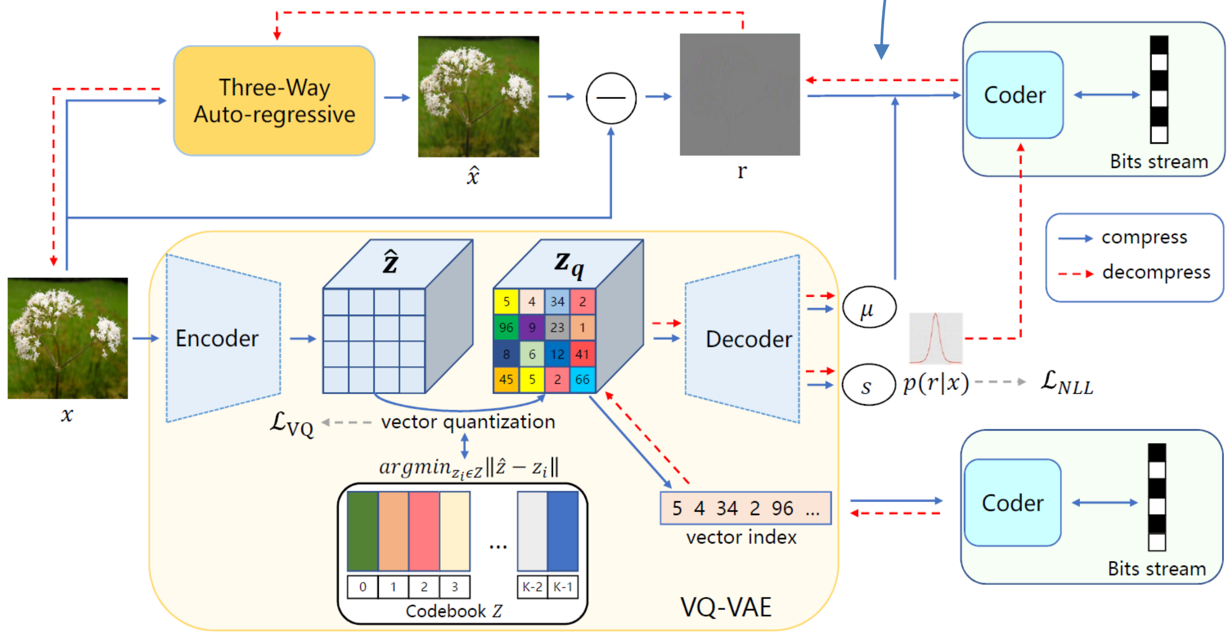
Parallel Auto-regressive decoding

All process in AI accelerated chips

Auto-Regressive + VQ-VAE

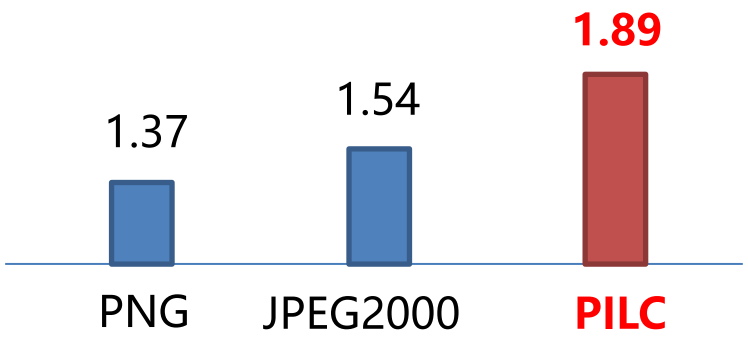
New distribution estimation

Semi-dynamic coder

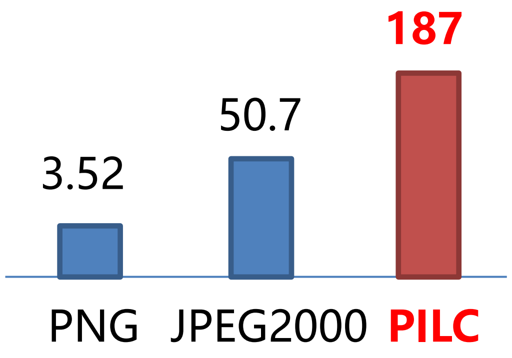


PILC: Result

- 30% better CR than PNG
- ~200 MB/s with single NVIDIA Tesla V100 chip
  - 15x faster than L3C, comparable CR



Compression ratio



Bandwidth (MB/s)

	Threads	Throughput rANS (MB/s)	Throughput ANS-AI (MB/s)
Encode	1	5.1	81.7
	4	10.8	239.0
	8	15.9	433.9
	16	21.6	598.8
Decode	1	0.8	122.0
	4	2.8	467.9
	8	5.5	925.9
	16	7.4	1190.0

Bandwidth for dynamic entropy coder

BPD	CIFAR10	ImageNet32	ImageNet64	DIV2K	CLIC.pro	CLIC.mobile	Throughput (MB/s) Compress	Throughput (MB/s) Decompress
PNG [2] (fastest)	6.44	6.78	6.09	4.64	4.23	4.39	55.9	118.2
PNG [2] (best)	5.91	6.41	5.77	4.23	3.90	3.80	3.0	83.5
WebP [31] (-z 0)	4.77	5.44	4.92	3.43	3.22	3.03	29.8	99.1
FLIF [23] (–effort 0)	4.27	**5.06**	4.70	3.24	3.03	2.82	6.2	4.2
JPEG2000 [24]	6.75	7.50	6.08	4.11	3.79	3.94	7.6	9.1
L3C [17]	4.55	5.19	**4.57**	**3.13**	**2.96**	**2.65**	12.3	6.3
PILC (Ours)	**4.23**	5.10	4.76	3.41	3.23	3.00	**180.3**	**217.2**

BPD result on different datasets. The lower BPD value, the better

	Phase	Throughput (MB/s)	Time (μs)
Compress	RAM $\rightarrow$ GPU	9246	0.33
	Model Inference	276	11.11
	Coder Encode	675	4.55
	GPU $\rightarrow$ RAM	2985	1.03
	Total	**180**	**17.02**
Decompress	RAM $\rightarrow$ GPU	11101	0.28
	Coder Decode	11091	0.28
	VQ-VAE Decode	721	4.26
	Code Decode	672	4.57
	AR Decode	869	3.53
	GPU $\rightarrow$ RAM	2521	1.20
	Total	**217**	**14.12**

Speed composition for each process

Further References

- High AI compression bandwidth
  - **iFlow (NeurIPS 21 Spotlight):** High-efficiency AI entropy codec with SoTA flows
  - **PILC (CVPR 22):** 200MB/s bandwidth on single V100 GPU, 10-100x faster than previous AI compression model. 30% compression ratio improvement over PNG
  - **SHVC (CVPR 22):** near SoTA compression ratio with 1/10 model size

- Other works
  - **DAMix (AAAI 23):** Combining AI models for SoTA compression ratio on generalized data
  - **iVPF (CVPR 21):** Achieving 5-15x speedup compared with LBB
  - **OSOA (NeurIPS 21):** Dynamic AI model while compression, 47% compression ratio improvement on generation dataset
  - **NelLoc (NeurIPS 21):** Theoretical generation ability analysis, 37% compression ratio improvement with 1/7 model size

Outline

- Neural Compression
- Text-to-image Generation
- Neural Theorem Proving

PixArt-α: Fast Training of Diffusion Transformer for Photorealistic Text-to-Image Synthesis

https://arxiv.org/abs/2310.00426

Huawei Noah's Ark Lab

Background

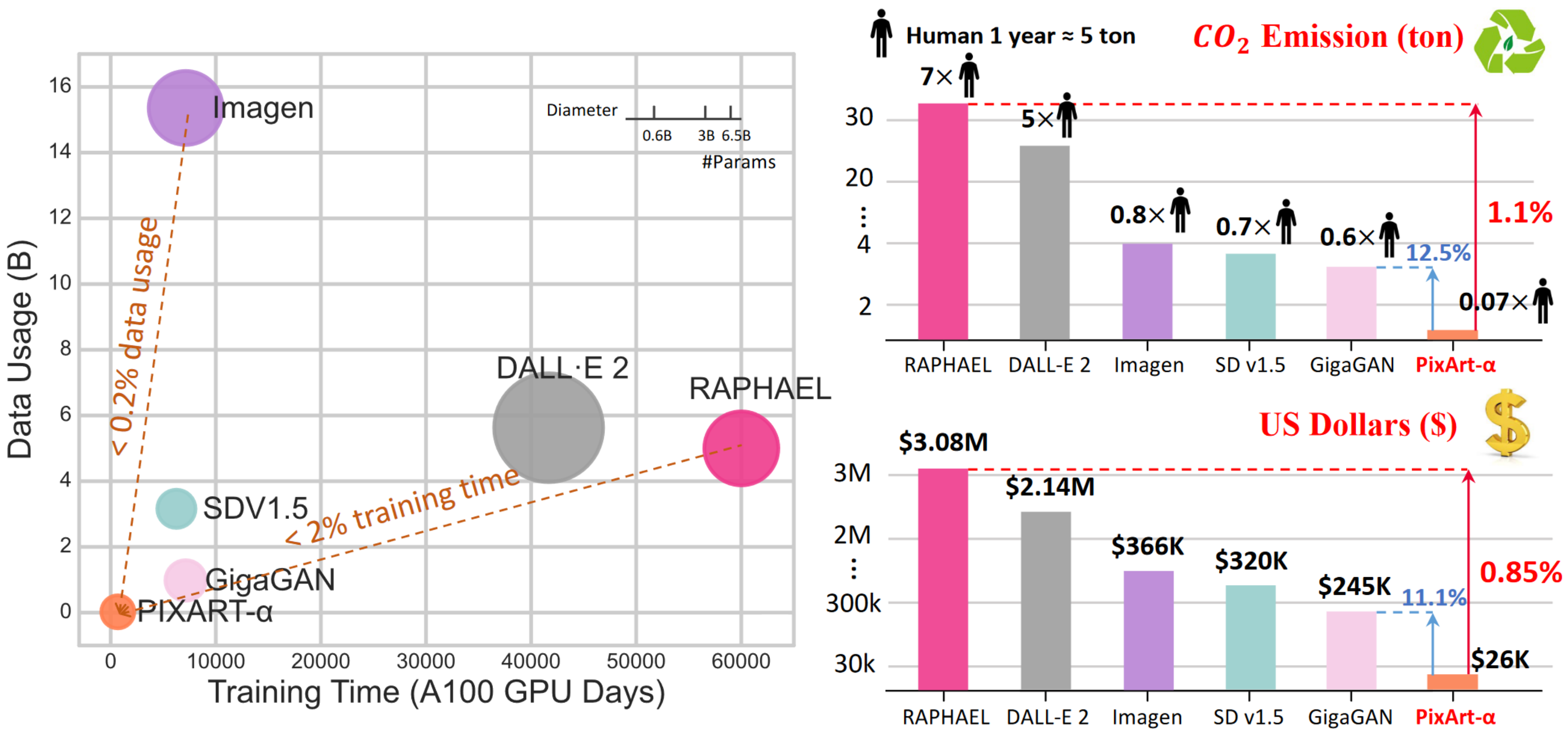
- SoTA text-to-image (T2I) generative models: DALL·E 3, Imagen, Stable Diffusion
- AIGC Applications: image editing, video generation, 3D assets creation, and many more.

DALL·E 3

Stable Diffusion XL

Background

- Challenges in advanced T2I models: enormous training costs/millions of GPU hours/CO2 emissions
- *Can we develop a high-quality image generator with affordable resource consumption?*



(a) Comparison of data usage and training time
(b) Comparison of CO_2 emission and training cost

PixArt-α - Low training cost, Powerful Synthesis Models

- PIXART-α's training speed markedly surpasses existing large-scale T2I models, far more affordable.
- PIXART-α only takes 10.8% of Stable Diffusion v1.5's training time (~675 vs. ~6,250 A100 GPU days), saving nearly \$300,000 (\$26,000 vs. \$320,000) and reducing 90% CO_2 emissions.

Problems with current popular generative training datasets

- Text-image misalignment
- Deficient description
- Infrequent vocabulary
- Low image quality

Table 1: Statistics of noun concepts for different datasets. **VN**: valid distinct nouns (appearing more than 10 times); **DN**: total distinct nouns; **Average**: average noun count per image.

Low density

Dataset	VN/DN	Total Noun	Average
LAION	210K/2461K = 8.5%	72.0M	6.4/Img
LAION-LLaVA	85K/646K = 13.3%	233.9M	20.9/Img
SAM-LLaVA	23K/124K = 18.6%	327.9M	29.3/Img
Internal	152K/582K = 26.1%	136.6M	12.2/Img

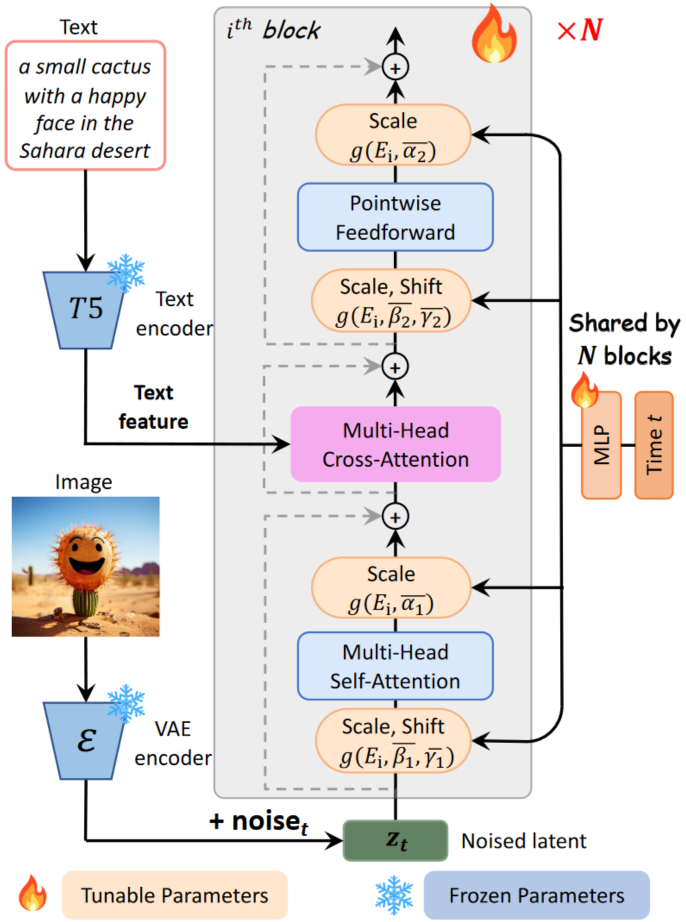
High density

Problems	Text-image misalignment	Deficient descriptions	Infrequent vocabulary
Samples			
Raw caption	What science says about pu'erh tea?	AH1370/1950 Saudi Arabia Gold One Guinea MS-63 NGC	2018 Kawasaki Jet Ski Ultra 310LX in Unionville, Virginia
LLaVA refined caption	The image features a close-up of a cup of tea with a saucer on a wooden table. The tea is described as "pu'erh tea," which is a type of Chinese tea known for its health benefits. The scene is set in a dimly lit room. The presence of a potted plant in the background adds a touch of nature and freshness to the scene.	The image shows a man working on scuba diving equipment at Blue Water Divers. The man is sitting at a table, working on a piece of equipment, possibly fixing or adjusting it. The scene is set in a workshop or a store, with various tools and equipment visible in the background.	The image features a man riding a jet ski on a body of water. The jet ski is green and white, and it is being used for recreational purposes. The man is smiling, indicating that he is enjoying his time on the water. The scene is set in a beach area.

PixArt-α: three core designs

Training strategy decomposition

Efficient T2I Transformer



High-informative data

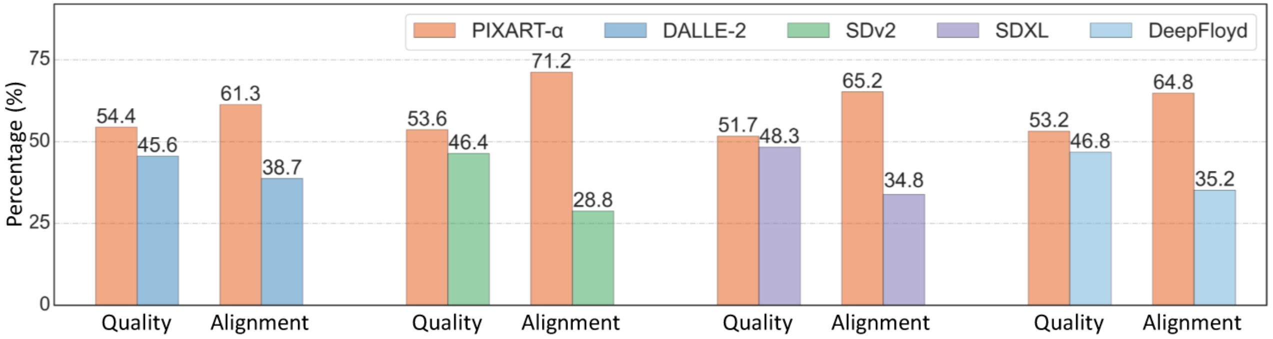
State-of-the-art Results

- Our method has the following two advantages:

1. Better quality and alignment: PIXART-α excels in both higher fidelity and superior alignment.
2. Better controllability: PIXART-α demonstrated exceptional performance in attribute binding, object relationships, and complex compositions, achieving superior compositional generation ability.

Model	Attribute Binding			Object Relationship		Complex↑
	Color ↑	Shape↑	Texture↑	Spatial↑	Non-Spatial↑	
Stable v1.4	0.3765	0.3576	0.4156	0.1246	0.3079	0.3080
Stable v2	0.5065	0.4221	0.4922	0.1342	0.3096	0.3386
Composable v2	0.4063	0.3299	0.3645	0.0800	0.2980	0.2898
Structured v2	0.4990	0.4218	0.4900	0.1386	0.3111	0.3355
Attn-Exct v2	0.6400	0.4517	0.5963	0.1455	0.3109	0.3401
GORS	0.6603	0.4785	0.6287	0.1815	0.3193	0.3328
Dalle-2	0.5750	0.5464	0.6374	0.1283	0.3043	0.3696
SDXL	0.6369	0.5408	0.5637	0.2032	0.3110	0.4091
PIXART-α	0.6886	0.5582	0.7044	0.2082	0.3179	0.4117

T2ICompBench



User study on Ernie-vilg 2.0-300

OpenAI also uses T2ICompBench to evaluate DALLE. 3 !

Huang et al. T2I-CompBench: A Comprehensive Benchmark for Open-world Compositional Text-to-image Generation, NeurIPS 2023

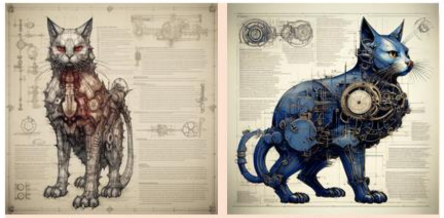
Compare with Midjourney

Art collection style and fashion shoot, in the style of made of glass, dark blue and light pink, paul rand, solarpunk, camille vivier, beth didonato hair, barbiecore, hyper-realistic.

A small cactus with a happy face in the Sahara desert

The image features a woman wearing a red shirt with an icon. She appears to be posing for the camera, and her outfit includes a pair of jeans. The woman seems to be in a good mood, as she is smiling. The background of the image is blurry, focusing more on the woman and her attire.

Pirate ship trapped in a cosmic maelstrom nebula, rendered in cosmic beach whirlpool engine, volumetric lighting, spectacular, ambient lights, light pollution, cinematic atmosphere, art nouveau style, illustration art artwork by SenseiJaye, intricate detail.

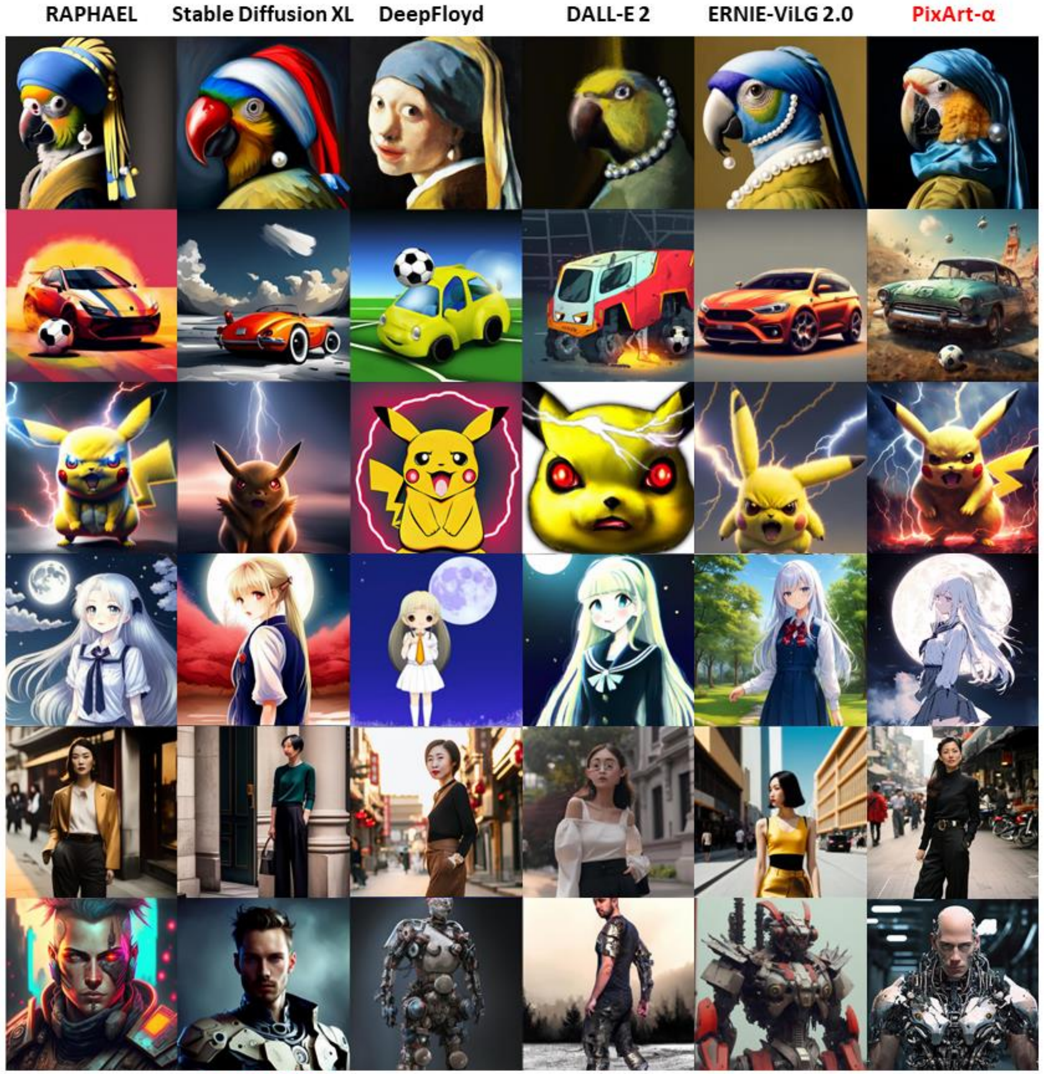


poster of a mechanical cat, technical Schematics viewed from front and side view on light white blueprint paper, illustration drafting style, illustration, typography, conceptual art, dark fantasy steampunk, cinematic, dark fantasy.

A dog that has been meditating all the time

Beautiful scene

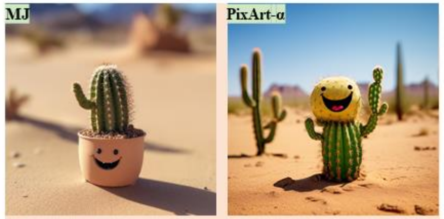
Compare with Other Methods



1. A parrot with a *pearl earring*, Vermeer style.
2. A car *playing soccer*, digital art.
3. A Pikachu with an *angry* expression and red eyes, with *lightning* around it, hyper realistic style.
4. Moonlight Maiden, cute girl in school uniform, long *white hair*, standing under the *moon*, celluloid style, *Japanese manga style*.
5. Street shot of a fashionable *Chinese lady* in Shanghai, wearing *black* high-waisted *trousers*.
6. Half *human*, half *robot*, repaired human, human flesh warrior, mech display, man in mech, *cyberpunk*.

Compare with Midjourney

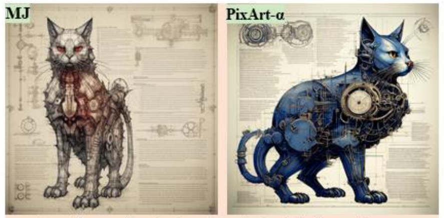
Art collection style and fashion shoot, in the style of made of glass, dark blue and light pink, paul rand, solarpunk, camille vivier, beth didonato hair, barbiecore, hyper-realistic.



A small cactus with a happy face in the Sahara desert

The image features a woman wearing a red shirt with an icon. She appears to be posing for the camera, and her outfit includes a pair of jeans. The woman seems to be in a good mood, as she is smiling. The background of the image is blurry, focusing more on the woman and her attire.

Pirate ship trapped in a cosmic maelstrom nebula, rendered in cosmic beach whirlpool engine, volumetric lighting, spectacular, ambient lights, light pollution, cinematic atmosphere, art nouveau style, illustration art artwork by SenseiJaye, intricate detail.

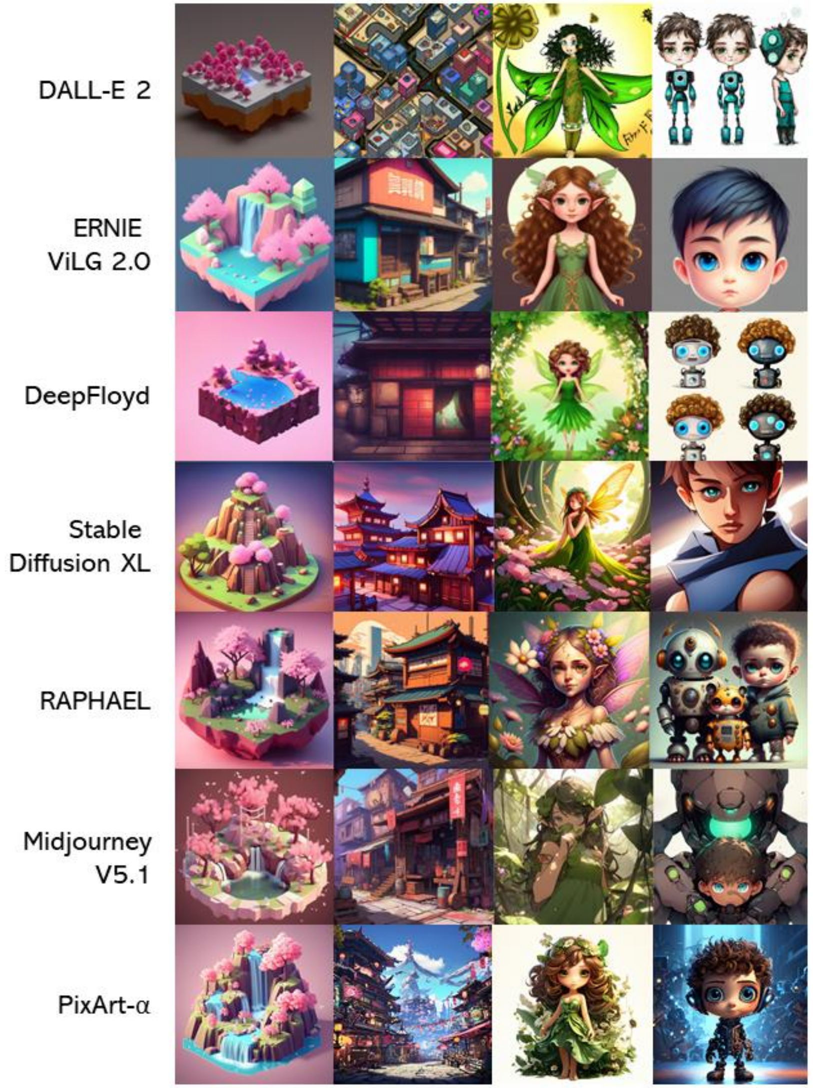


poster of a mechanical cat, technical Schematics viewed from front and side view on light white blueprint paper, illustration drafting style, illustration, typography, conceptual art, dark fantasy steampunk, cinematic, dark fantasy.

A dog that has been meditating all the time

Beautiful scene

Compare with Other Methods



1. A cute little matte low poly isometric *cherry blossom forest island*, *waterfalls*, lighting, soft shadows, trending on Artstation, 3d render, monument valley, fez video game.
2. A shanty version of Tokyo, new rustic style, *bold colors with all colors palette*, video game, genshin, tribe, fantasy, overwatch.
3. Cartoon characters, mini characters, figures, illustrations, flower fairy, green dress, *brown hair, curly long hair, elf-like wings, many flowers and leaves*, natural scenery, *golden eyes*, detailed light and shadow , a high degree of detail.
4. Cartoon characters, mini characters, hand-made, illustrations, *robot kids*, color expressions, boy, *short brown hair*, *curly hair*, *blue eyes*, technological age, *cyberpunk*, big eyes, cute, mini, detailed light and shadow, high detail.

More Samples

A female painter with a *brush* in hand, *white background*, *painting*, looking very *powerful*.

marvel movie character, *iron man*, dress up to match movie character, full body photo, *American apartment*, lying down, life in distress, *messy*, lost hope, food, wine, hd, 8k, real, reality, super detail, 8k post photo manipulation, real photo

A *worker* that looks like *a mixture of cow and horse* is working hard to *type code*

A *baby painter* trying to draw very simple picture, white background

knolling of a *drawing tools and books*, knowledge, white background

real beautiful *woman, Chinese*

Chinese painting of *grapes*

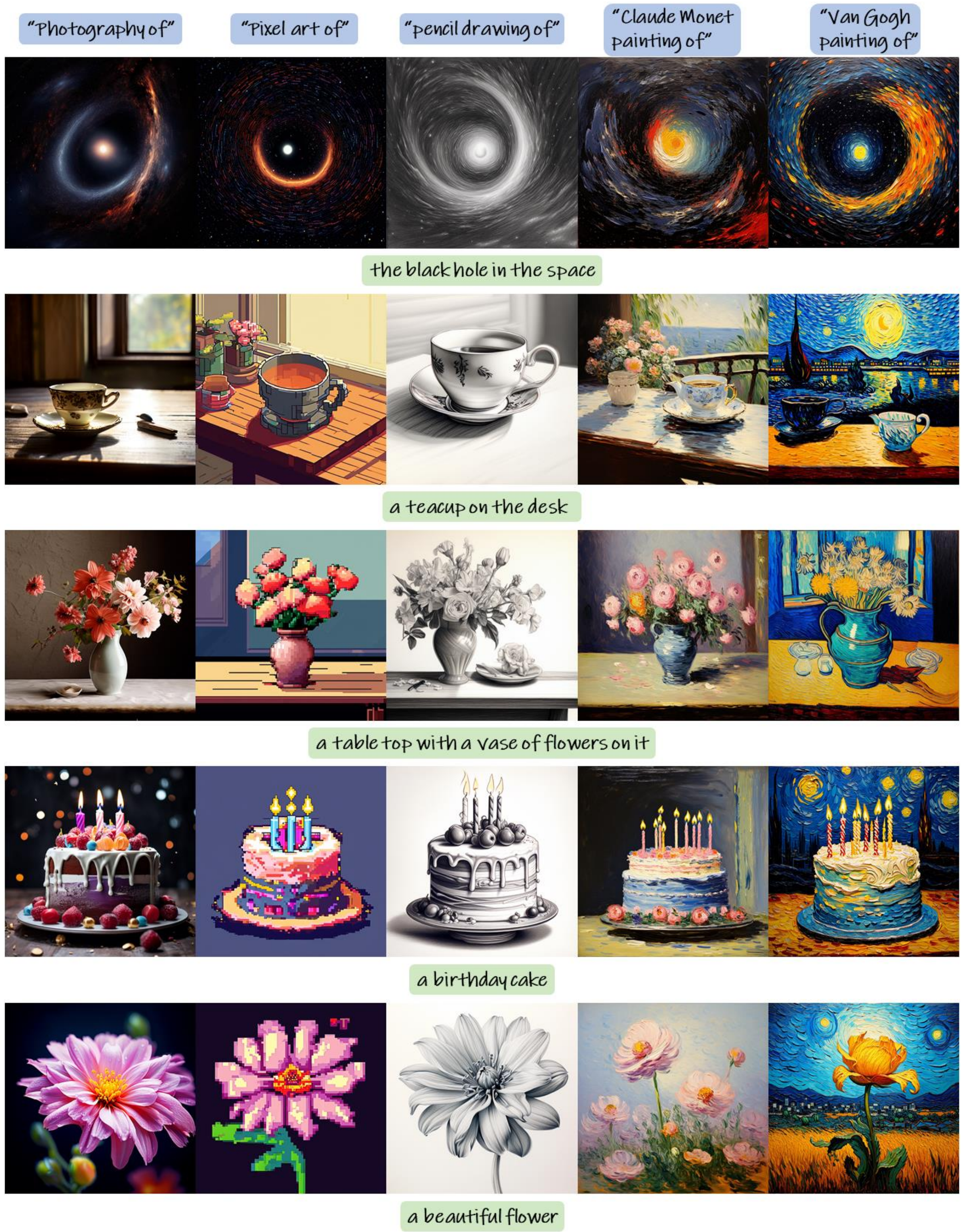
A *snowy mountain*

happy

I *want to supplement vitamin c*, please help me paint related food.

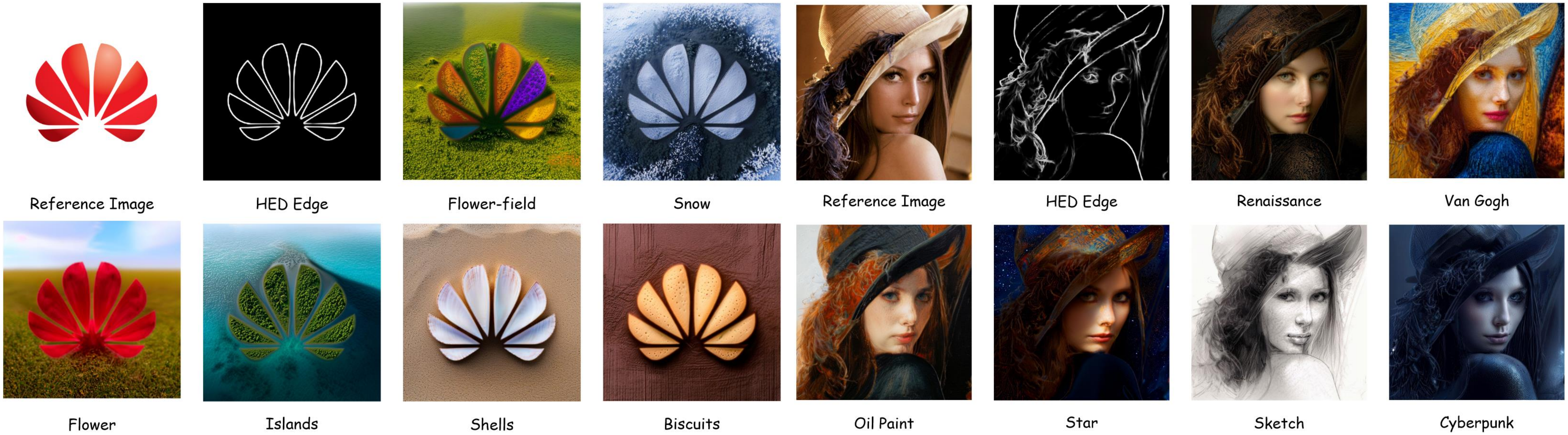
An *alien octopus* floats through a portal *reading a newspaper*

Style control with text



Application 1: PixArt-α + DreamBooth

Application 2: PixArt-α + ControlNet



Further References

2D Generation

- DiffFit: Unlocking Transferability of Large Diffusion Models via Simple Parameter-efficient Fine-Tuning, **ICCV 2023 Oral**
- Complexity Matters: Rethinking the Latent Space for Generative Modeling, **NeurIPS 2023 Spotlight**
- SA-Solver: Stochastic Adams Solver for Fast Sampling of Diffusion Models, **NeurIPS 2023**
- Diff-Instruct: A Universal Approach for Transferring Knowledge From Pre-trained Diffusion Models, **NeurIPS 2023**

3D Generation

- DiT-3D: Exploring Plain Diffusion Transformers for 3D Shape Generation, **NeurIPS 2023**
- DiffComplete: Diffusion-based Generative 3D Shape Completion, **NeurIPS 2023**

Generation Evaluation Benchmark

- T2I-CompBench: A Comprehensive Benchmark for Open-world Compositional Text-to-image Generation, **NeurIPS 2023 Datasets and Benchmarks Track**

Outline

- Neural Compression
- Text-to-image Generation
- Neural Theorem Proving

LEGO-Prover: Neural Theorem Proving with Growing Libraries

https://arxiv.org/abs/2310.00656

Huawei Noah's Ark Lab

Automated Theorem Proving

Problem statement

prove that $\sqrt{2}$ is irrational

```
lemma "sqrt 2 ∉ ℚ"
```

Prover

Proof

Assuming $\sqrt{2} \in \mathbb{Q}$, we have $\sqrt{2}=a/b$, and a, b is coprime. then have $2 = a^2/b^2$ and $2 \times b^2 = a^2$. thus, we know a is even, $a = 2c$. substitute a into previous equation, we have $b^2 = (2 * c)^2$. Thus, we know b is also even, and a, b is not coprime. This is contradiction to the origin assumption. ■

```
proof
  assume "sqrt 2 ∈ ℚ"
  then obtain a b::int where "sqrt 2 = a/b"
    "coprime a b" "b ≠ 0" sledgehammer
  then have c: "2 = a^2 / b^2"
    sledgehammer
  then have "b^2 ≠ 0" sledgehammer
  then have *: "2*b^2 = a^2"
    sledgehammer
  then have "even a"
    sledgehammer
  then obtain c::int where "a=2*c"
    sledgehammer
  with * have "b^2 = 2*c^2"
    sledgehammer
  then have "even b"
    sledgehammer
  with ‹coprime a b› ‹even a› ‹even b›
    show False sledgehammer
qed
```

Formal system

LM + Search (**gpt-f** OpenAI 2021, **Thor** Cambridge 2021, **DT-Solver** Ours 2023):

- **Language model** suggests **action** given **current state.**
- **Formal system** executes action and updates state.
- **Search algorithm** finds correct action path.

```
lemma "sqrt 2 ∉ ℚ"
```

goals: 1. sqrt 2 ∉ ℚ

```
proof
```

goals: 1. sqrt 2 ∈ ℚ ⟹ False

```
assume "sqrt 2 ∈ ℚ"
```

premise: sqrt 2 ∈ ℚ
goals: 1. sqrt 2 ∈ ℚ ⟹ False

```
then obtain a b::int where "sqrt 2 = a/b"
  "coprime a b" "b ≠ 0" sledgehammer
```

premise: sqrt 2 = real_of_int a / real_of_int b
coprime a b
b ≠ 0
goals: 1. sqrt 2 ∈ ℚ ⟹ False

```
then have c: "2 = a^2 / b^2"
  sledgehammer
```

…

…

Automated theorem proving:

LLM with ICL (**DSP** Cambridge 2022, **Subgoal-based** HKU 2023):

- **ChatGPT** generates **entire proof in one go**.
- Use **in-context learning** to prompt the LLM
- **Formal system** verifies the proof

```
lemma "sqrt 2 ∉ ℚ"
  proof
    assume "sqrt 2 ∈ ℚ"
    then obtain a b::int where "sqrt 2 = a/b"
      "coprime a b" "b ≠ 0" sledgehammer
    then have c: "2 = a^2 / b^2"
      sledgehammer
    then have "b^2 ≠ 0" sledgehammer
    then have *: "2*b^2 = a^2"
      sledgehammer
    then have "even a"
      sledgehammer
    then obtain c::int where "a=2*c"
      sledgehammer
    with * have "b^2 = 2*c^2"
      sledgehammer
    then have "even b"
      sledgehammer
    with ‹coprime a b› ‹even a› ‹even b›
      show False sledgehammer
  qed
```

No goals !

Error: xxx

- Verifiable
- Longer reasoning chain
- Data scarcity

Motivation

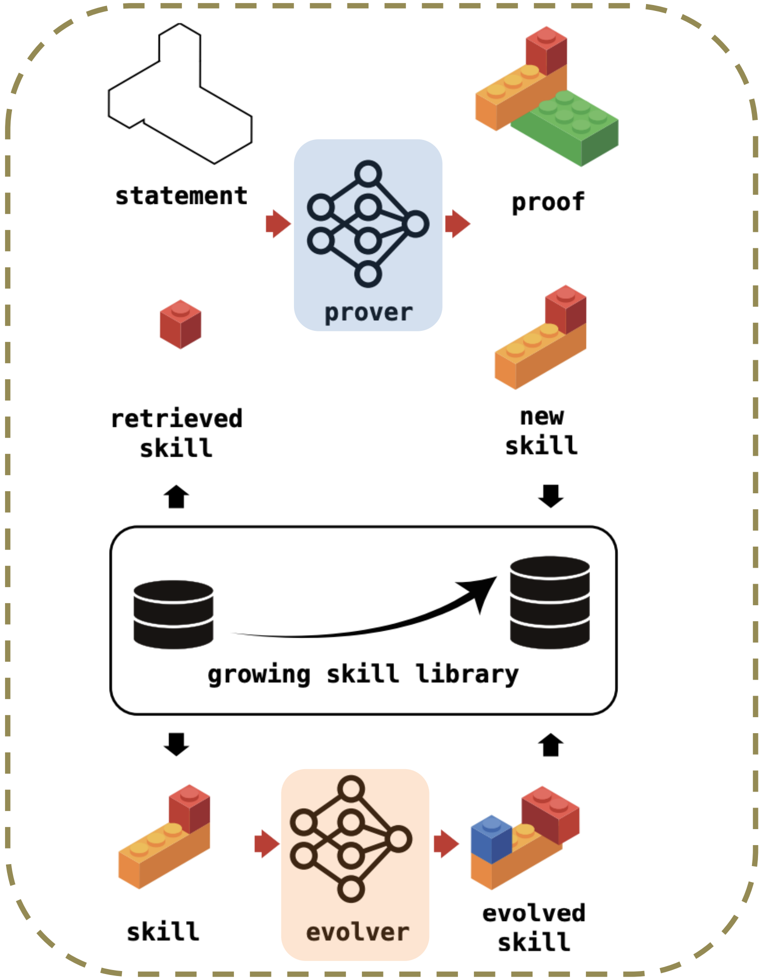
- Problems with existing provers:
  - Each theorem is proved **independently**.
  - Proven conjectures are **not shared** among problems.
  - LLM struggles to generate **correct long-chain proof** (hallucination).

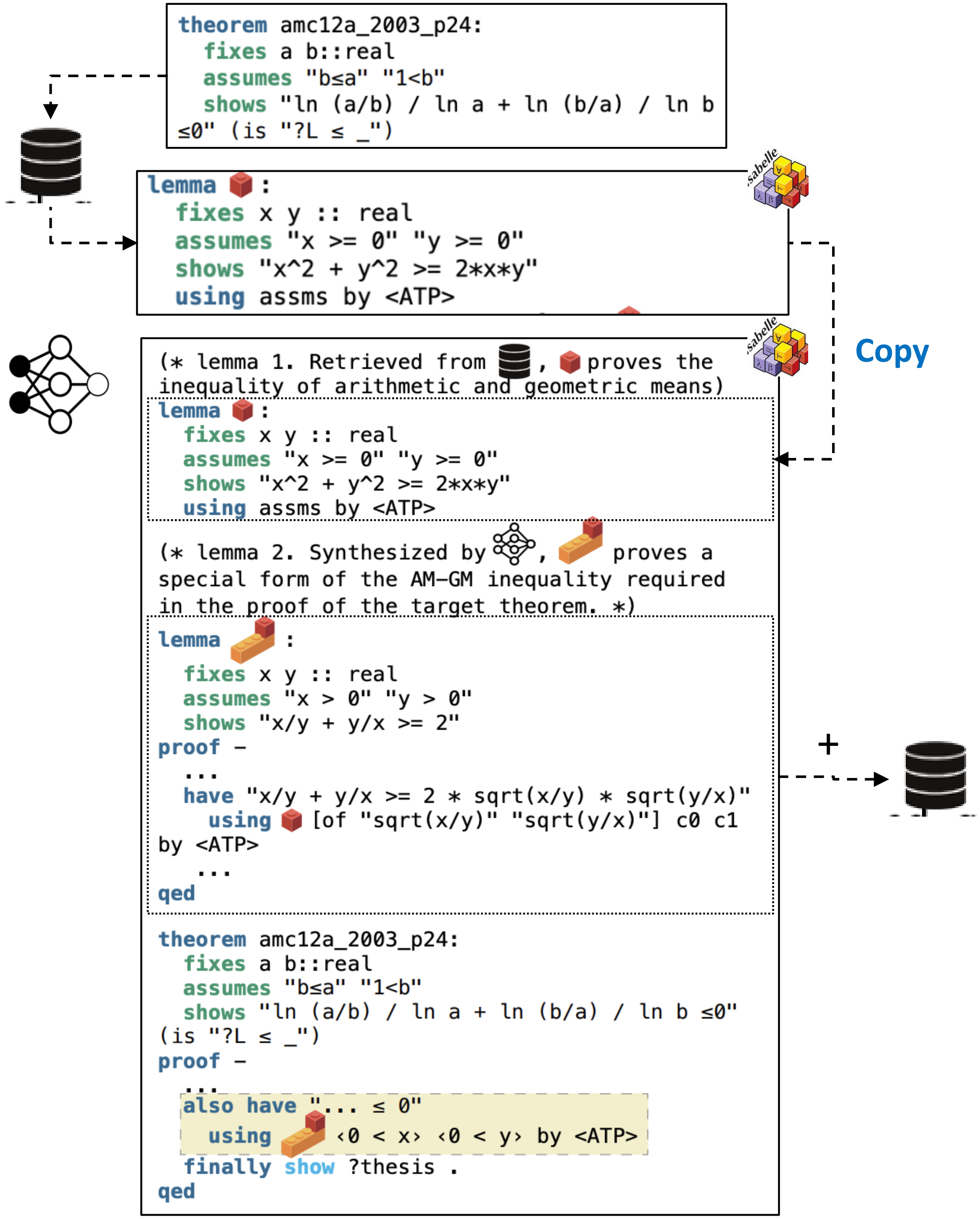
- Ideal provers:
  - Extract & **reuse** useful lemmas during each theorem proving, **to reduce reasoning length**
  - Maintain & **grow** a library of proven theorems/lemmas (online & offline)
  - Leverage the power of LLM (prover)
  - Leverage the verification capability of formal systems (Lean, Isabelle)
  - Imitate or surpass human proving process

LEGO-Prover: prove theorem like building LEGO

Prove in a **block-by-block manner**

- Prove **sub-goal lemmas**
- Prove theorem using sub-goal lemmas.
- Sub-goal Lemmas: retrieved from skill library, or constructed online



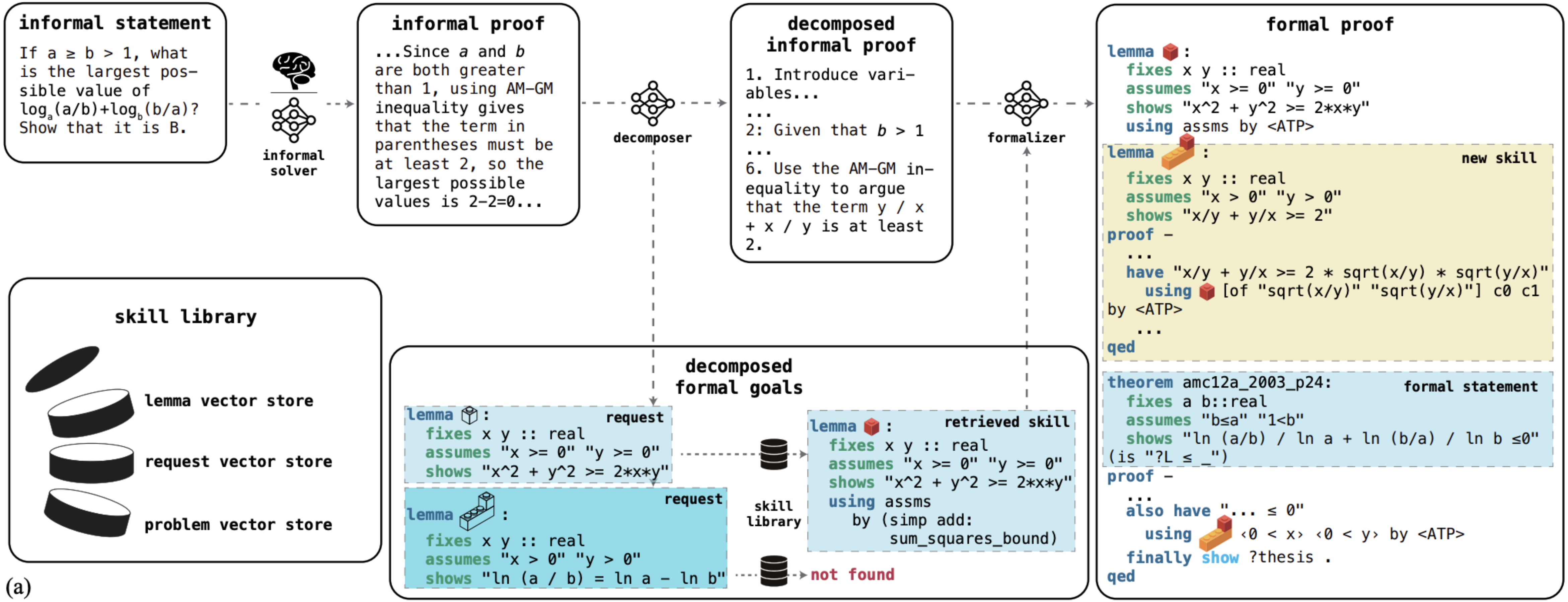


LEGO-Prover consists of a prover, an evolver, and a growing skill library

LEGO-Prover: prover

Three proof steps

- **Informal solver:** produce an informal proof
- **Decomposer:** produce step-by-step informal proof and sub-goals lemma statements, which are used to retrieve useful lemma from the skill library.
- **Formalizer:** prove theorem with step-by-step informal proof and retrieved lemmas block-by-block.



(a)

LEGO-Prover: prover

How to prompt LLM to generate lemmas

- System instructions

Prompt text

- You are **strongly encouraged** to create or use **useful and reusable lemmas** to solve the problem.
- The lemmas should be as general as possible (**generalizable**), and be able to cover a large step in proofs (**non-trivial**). Please ensure that your proof is well-organized and easy to follow, with each step building upon the previous one.

- Special block-by-block structure in in-context learning examples

In-context learning example

Informal proof:
step1. introduce variables...
…

Formal statement:
theorem amc12a_2003_p24:
  fixes a b :: real
  …

Formal proof:
lemma am_gm: fixes x y :: real …

lemma am_gm_extenstion: fixes x y :: real …

theorem amc12a_2003_p24:
  fixes a b :: real

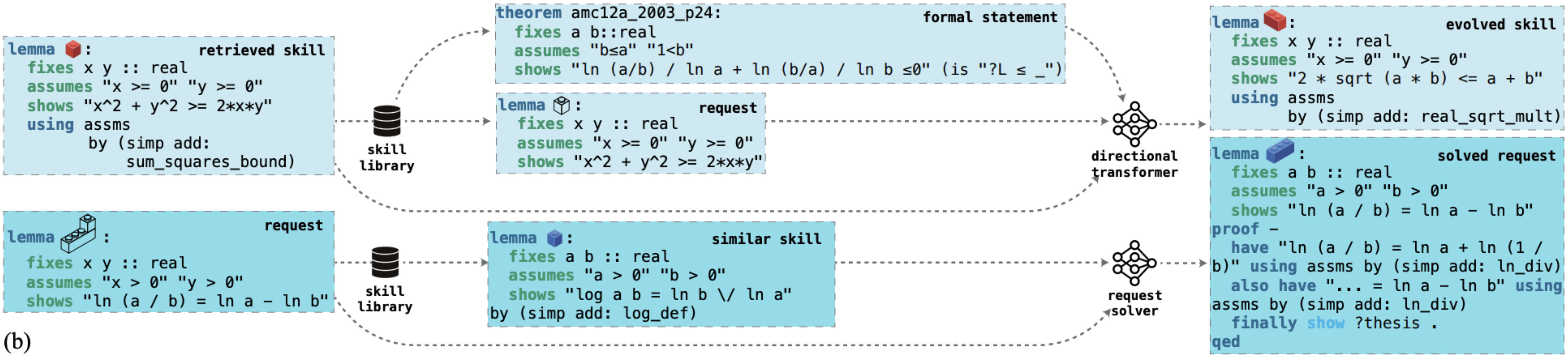
LEGO-Prover: evolver

Transforms existing skills into a more general and reusable form, or directly solves requested subgoals proposed by the prover.

- **Directional transformer** evolves skill using four type of specific direction
- **Request solver** directly solves the request proposed by the decomposer.

Different types of directional transformer

Evolve type	Description
Identify key concepts	Determine the essential ideas, methods, or theorems that are crucial to solving the initial problem.
Parameterize	If the problem involves specific numbers, generalize it by replacing these with variables.
Scale complexity	Try both simpler and more complicated versions of the problem to see how the approach adapts.
Extend dimensions	If the problem is defined in a specific number of dimensions, consider if it holds in more or fewer dimensions.



(b)

Experiments

- **Thor (Cambridge, NeurIPS 2022)**: LM + Search. LM trained on single step state-action pairs. Find proof with best first search.
- **Thor + expert iteration (Google + Cambridge, NeurIPS 2022)**: LM + Search. Extend Thor with extensive data by Codex.
- **DSP (Cambridge, ICLR 2023)**: LLM with ICL, use informal proof to guide Codex to generate formal sketch.
- **Subgoal-Learning (HKU + Cambridge, NeurIPS 2023)**: LLM with ICL, extends DSP with step-by-step informal proof.

Success rate	LLM	miniF2F-valid	miniF2F-test
Baselines			
Thor (Jiang et al., 2022a)	-	28.3%	29.9%
Thor + expert iteration (Wu et al., 2022)	Codex	37.3%	35.2%
Draft, sketch, and Prove (Jiang et al., 2022b)	Codex	42.6%	39.3%
Subgoal-Learning (Zhao et al., 2023)	ChatGPT	48.0%	45.5%
Ours (100 attempts)			
LEGO-Prover (model informal proof)	ChatGPT	52.4%	45.5%
LEGO-Prover (human informal proof)	ChatGPT	55.3%	**50.0%**
LEGO-Prover*	ChatGPT	**57.0%**	**50.0%**
Ablations (50 attempts)			
LEGO-Prover	ChatGPT	50.4%	-
- Skill Library	ChatGPT	47.1%	-

Experiments: case study

retrieved skill

```
lemma am_gm:
  fixes x :: real
  assumes "x > 0"
  shows "x + 1/(2*x) ≥ sqrt 2"
proof -
  have "(sqrt x - sqrt (1/(2*x)))^2 ≥ 0" by simp
  then have "x + 1/(2*x) - 2 * sqrt x * sqrt (1/(2*x)) ≥ 0"
    by (smt (verit) add_le_imp_le_diff assms divide_nonneg_nonneg less_eq_-
real_def real_sqrt_pow2 sum_squares_bound)
  then have "x + 1/(2*x) - sqrt 2 ≥ 0"
    by (metis add_divide_distrib assms div_by_1 divide_divide_eq_right di-
vide_self less_numeral_extra(3) mult_2 one_add_one real_div_sqrt re-
al_sqrt_divide real_sqrt_eq_1_iff real_sqrt_gt_0_iff real_sqrt_one times_di-
vide_eq_right zero_le_numeral)
  then have "x + 1/(2*x) ≥ sqrt 2" by simp
  then show ?thesis by simp
qed
```

synthesized proof

```
lemma am_gm:
  fixes x :: real
  assumes "x > 0"
  shows "x + 1/(2*x) ≥ sqrt 2"
proof -
  have "(sqrt x - sqrt (1/(2*x)))^2 ≥ 0" by simp
  then have "x + 1/(2*x) - 2 * sqrt x * sqrt (1/(2*x)) ≥ 0"
    by (smt (verit) add_le_imp_le_diff assms divide_nonneg_nonneg less_eq_-
real_def real_sqrt_pow2 sum_squares_bound)
  then have "x + 1/(2*x) - sqrt 2 ≥ 0"
    by (metis add_divide_distrib assms div_by_1 divide_divide_eq_right di-
vide_self less_numeral_extra(3) mult_2 one_add_one real_div_sqrt re-
al_sqrt_divide real_sqrt_eq_1_iff real_sqrt_gt_0_iff real_sqrt_one times_di-
vide_eq_right zero_le_numeral)
  then have "x + 1/(2*x) ≥ sqrt 2" by simp
  then show ?thesis by simp
qed

theorem algebra_amgm_faxinrrp2msqrt2geq2mxm1div2x:
  "⋀x. (x>0) ⟹ 2 - sqrt 2 ≥ 2 - x - 1/ (2 * x)"
proof -
  fixes x :: real
  assumes h0: "x>0"
  (*Step 1: We want to show that $x + \frac{1}{2x} \geq \sqrt{2}$ is true.*)
  (* Step 2: Apply the AM-GM inequality to $x$ and $\frac{1}{2x}$ to get $x
+ \frac{1}{2x} \geq 2\sqrt{\frac{x}{2x}}$. *)
  (* Step 3: Simplify $2\sqrt{\frac{x}{2x}}$ to $\sqrt{2}$. *)
  (* Step 4: Conclude that $x + \frac{1}{2x} \geq \sqrt{2}$ is true. *)
have c1: "f (500 * (6/5)) = f 500 / (6/5)"
  have "x + 1/(2*x) ≥ sqrt 2" using am_gm[OF h0] by simp
  (* Step 5: Since $2 - \sqrt{2} \geq 2 - x - \frac{1}{2x}$ is equivalent to
$x + \frac{1}{2x} \geq \sqrt{2}$, we can conclude that $2 - \sqrt{2} \geq 2
- x - \frac{1}{2x}$ is true. *)
  then show "2 - sqrt 2 ≥ 2 - x - 1/ (2 * x)" by simp
qed
```

(a) Directly Use

Retrieved skill:

lemma am_gm: For a real number x, $x > 0$, prove that $x + \frac{1}{2x} \geq \sqrt{2}$.

Proof. We have $\left(\sqrt{x} + \sqrt{\frac{1}{2x}}\right)^2 \geq 0$. Expanding the inequality, we obtain $x + \frac{1}{2x} - 2 * \sqrt{x} * \sqrt{\frac{1}{2x}} \geq 0$. From which we have $x + \frac{1}{2x} - \sqrt{2} \geq 0$, and thus $x + \frac{1}{2x} \geq \sqrt{2}$. ∎

copy paste by LLM

Synthesized proof:

lemma am_gm: For a real number x, $x > 0$, prove that $x + \frac{1}{2x} \geq \sqrt{2}$.

Proof. We have $\left(\sqrt{x} + \sqrt{\frac{1}{2x}}\right)^2 \geq 0$. Expanding the inequality, we obtain $x + \frac{1}{2x} - 2 * \sqrt{x} * \sqrt{\frac{1}{2x}} \geq 0$. From which we have $x + \frac{1}{2x} - \sqrt{2} \geq 0$, and thus $x + \frac{1}{2x} \geq \sqrt{2}$. ∎

theorem algrebra_amgm_faxinrrp: Given a real number x, prove that the expression $2 - \sqrt{2} \geq 2 - x - \frac{1}{2x}$ holds true for all $x > 0$.

Proof. Using the proven lemma **am_gm,** we can show that $x + \frac{1}{2x} \geq \sqrt{2}$. Multiplying both sides with -1 and add 2, we obtain $2 - \sqrt{2} \geq 2 - x - \frac{1}{2x}$. ∎

Case **directly use**:

- A verified lemma am_gm is retrieved from skill libraries (with proof).
- Formalizer synthesized final proof using retrieved skill directly.

  1) Copy pasted the lemma **am_gm** in the proof code directly.

  2) Prove main theorem using the proven **am_gm** lemma.

Experiments: case study

Case **propose lemma by imitation**:

- A verified lemma **prod_1n_4n** is retrieved from skill libraries (proof).
- Formalizer synthesized final proof by solving the lemma imitating the retrieved skill.

  1) Imitate the lemma **prod_1n_4n**. The formalizer uses induction to prove **prod_frac_common_factor.**

  2) Prove main theorem using the proven **prod_frac_common_factor** lemma**.**

Retrieved skill:

lemma prod_1n_4n: for a natural number n, prove that $\prod_1^n 4 * k = 4^n * n!$

Proof. Let's prove by induction on n. For the base case we have $\prod_1^1 4 * k = 4^1 * 1!$, we have $4 = 4$. For induction step, assuming $\prod_1^j 4 * k = 4^j * j!$, we prove that $\prod_1^{j+1} 4 * k = 4^{j+1} * (j + 1)!$. Since $\prod_1^{j+1} 4 * k = 4^j * j! * (4 * (j + 1))$, thus $\prod_1^{j+1} 4 * k = 4^{j+1} * (j + 1)!$. ■

Propose lemma by imitation

Synthesized proof:

lemma prod_frac_common_factor: for a natural number n and a real number a, prove that $\prod_1^n \frac{ak+a}{ak} = n + 1$

Proof. Let's prove by induction on n. For the base case we have $\prod_1^1 \frac{ak+a}{ak} = \frac{a*1+a}{a*1} = 1 + 1$. For induction step, we have $\prod_1^{n+1} \frac{ak+a}{ak} = \frac{a*(n+1)+a}{a*(n+1)} * \prod_1^n \frac{ak+a}{ak} = \frac{a*(n+1)+a}{a*(n+1)} * n + 1 = n + 1 + 1$.■

theorem amc12a_2008_p4: Prove that $\prod_1^{501} \frac{4k+4}{4k} = 502$.

Proof. Using the proven lemma **prod_frac_common_factor,** we can show that $\prod_1^{501} \frac{4k+4}{4k} = 501 + 1 = 502$. ■

retrieved skill

```
lemma prod_1n_4n:
  fixes n :: nat
  shows "(∏k = 1..n. (4 ∗ k)) = 4^n ∗ fact n"
proof(induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  then show ?case by (simp add: algebra_simps)
qed
```

synthesized proof

```
lemma prod_frac_common_factor:
  fixes n:: nat and a::real
  assumes "a ≠ 0"
  shows "(∏k = 1..n. (a ∗ k + a) / (a ∗ k)) = n + 1"
proof(induct n)
  case 0
  then show ?case by simp
next
  case (Suc n)
  have "(∏k = 1..Suc n. (a ∗ k + a) / (a ∗ k)) = ((a ∗ Suc n + a) / (a ∗ Suc
n)) ∗ (∏k = 1..n. (a ∗ k + a) / (a ∗ k))"
    by simp
  also have "... = ((a ∗ Suc n + a) / (a ∗ Suc n)) ∗ (n + 1)"
    using Suc by simp
  also have "... = Suc n + 1"
    using assms by (smt (verit) add.commute add_divide_distrib divisors_zero
nonzero_mult_div_cancel_left nonzero_mult_divide_mult_cancel_right of_nat_1
of_nat_add of_nat_neq_0 plus_1_eq_Suc right_inverse_eq times_divide_eq_left)
  finally show ?case by simp
qed

theorem amc12a_2008_p4:
  "(∏k::nat=1..501. ((4::real) ∗ k + 4) / (4 ∗ k)) = 502"
proof -
  (∗ Step 1: Rewrite the given product as $\frac {8}{4}\cdot\frac {12}{8}\c-
dot\frac {16}{12}\cdots\frac {4n + 4}{4n}\cdots\frac {2008}{2004}$. ∗)
  have "(∏k::nat=1..501. ((4::real) ∗ k + 4) / (4 ∗ k)) = (∏k::nat=1..501.
(4 ∗ (k + 1)) / (4 ∗ k))"
    by eval
  (∗ Step 2: Simplify the product by canceling out common factors. Notice
that each term in the numerator cancels with the corresponding term in the
denominator, leaving only the last term $\frac {2008}{4}$. ∗)
  also have "... = (∏k::nat=1..501. (k + 1) / k)"
    by eval
  (∗ Use lemma 1 to simplify the product ∗)
  also have "... = 501 + 1"
    using prod_frac_common_factor[of "1::real" "501"] by eval
  (∗ Step 3: Calculate the value of $\frac {2008}{4}$ to find that it is
equal to $502$. ∗)
  also have "... = 502"
    by simp
  (∗ Step 4: Conclude that the given product is equal to $502$. ∗)
  finally show ?thesis by simp
qed
```

(b) Propose Lemma by Imitation

Takeaway

- Generative models are a powerful and rewarding way to model data that enables various applications.

- AI is reshaping many fields: data compression, information communication, art generation, mathematical reasoning, science discovery, …
  - Data is growing rapidly. Time to rethink how to store and transmit data.
  - AIGC is on the way.
  - Math is becoming a new area for AI.

- Fundamental challenges in generative AI
  - How to guide the generation towards desired objectives
  - How to verify the generated contents
  - Learning theory for generative AI