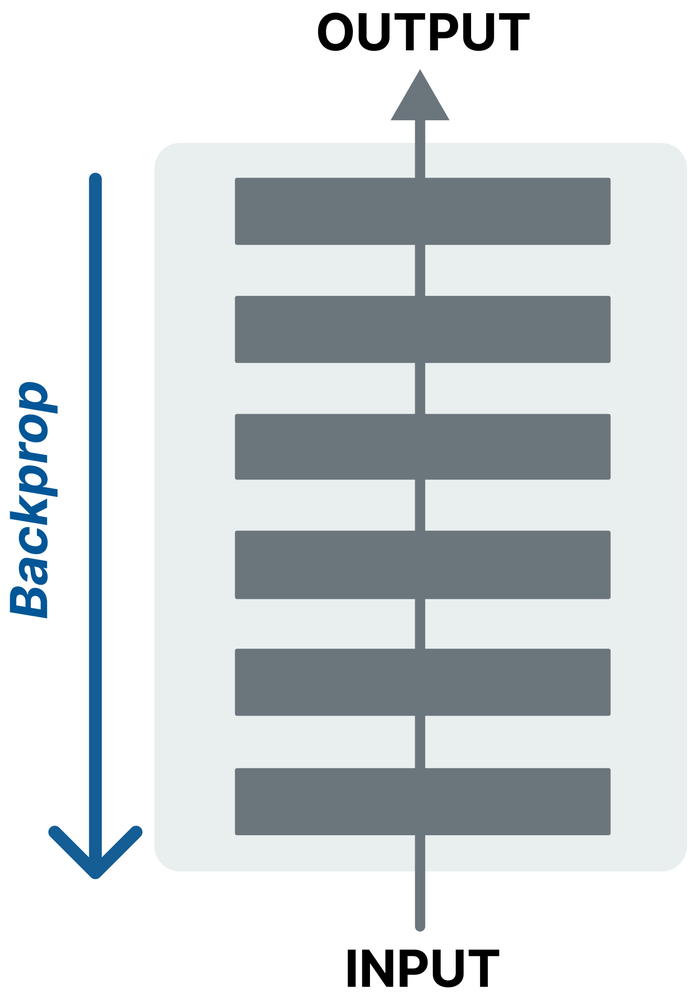
DiffusionBlocks:

Block-wise Neural Network Training via Diffusion Interpretation

Makoto Shing, Masanori Koyama, Takuya Akiba

sakana.ai

The Memory Bottleneck in Deep Learning



- End-to-end backprop requires storing activations across all layers
- Memory grows linearly with depth
- This limits the scale of models we can train on a given hardware

→ Block-wise training: train blocks independently

But Existing Block-Wise Methods Fail

Two critical limitations

- ❌ **Rely on ad-hoc local objectives without theoretical grounding**
- ❌ **Limited to classification tasks**

We need:

- ✔ **Theoretically grounded framework**
- ✔ **Generalizable beyond classification**

→ **Our proposal: DiffusionBlocks**

Key Insight

Residual connections correspond to Euler steps of an ODE

(Haber & Ruthotto, 2017; Chen et al., 2018)

$$z_\ell = z_{\ell-1} + f_\theta(z_{\ell-1}) \leftrightarrow \text{Euler step}$$

This naturally extends to discretized steps of the reverse diffusion process

Diffusion Property

The loss can be optimized at any noise level independently, without knowledge of other noise levels

We leverage this to build...

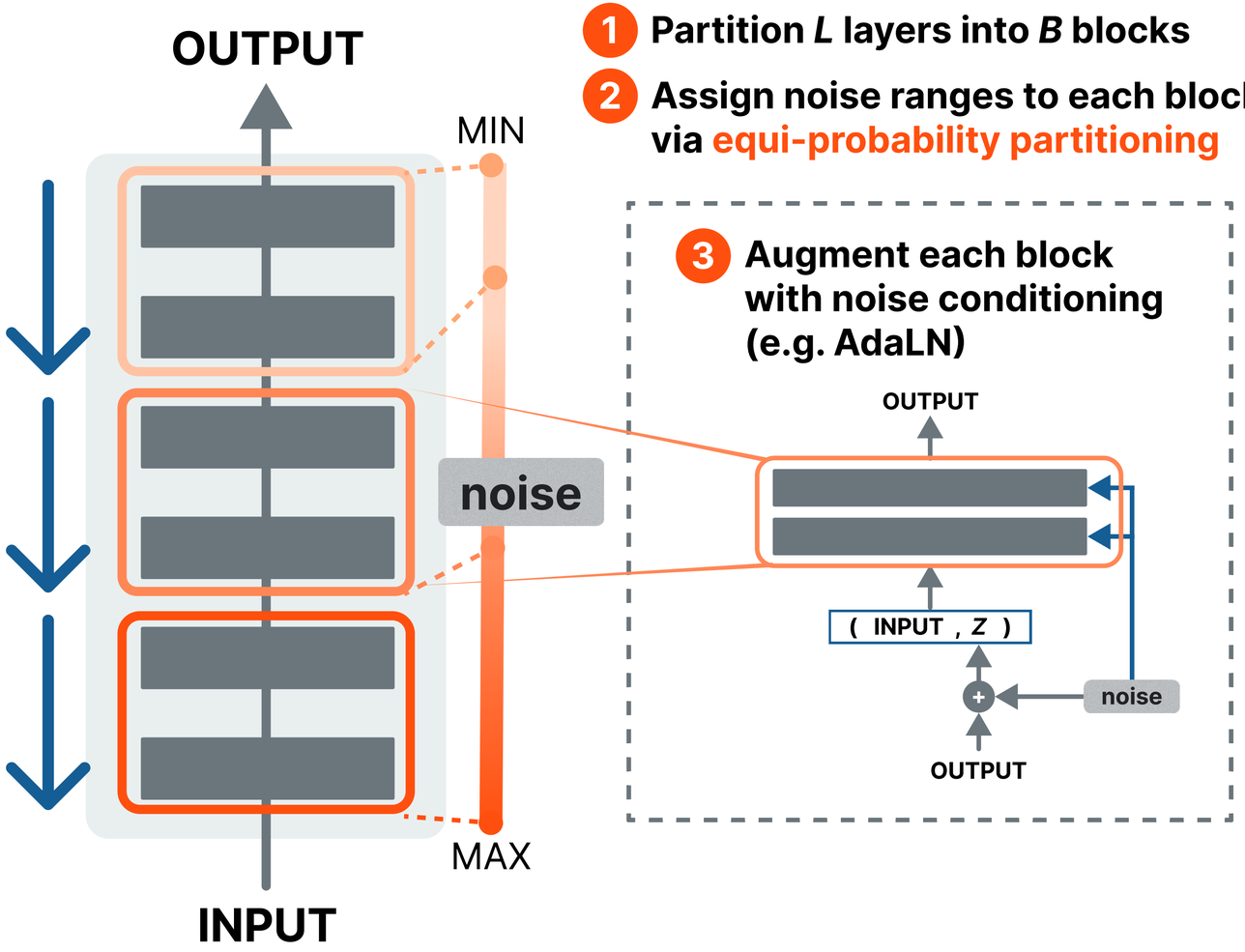
Ours: DiffusionBlocks

By assigning different noise ranges to different blocks:

→ Each block trained independently

→ B× memory reduction

3-Step Conversion + Equi-Probability Partitioning



1. Partition *L* layers into *B* blocks
2. Assign noise ranges to each block via equi-probability partitioning
3. Augment each block with noise conditioning (e.g. AdaLN)

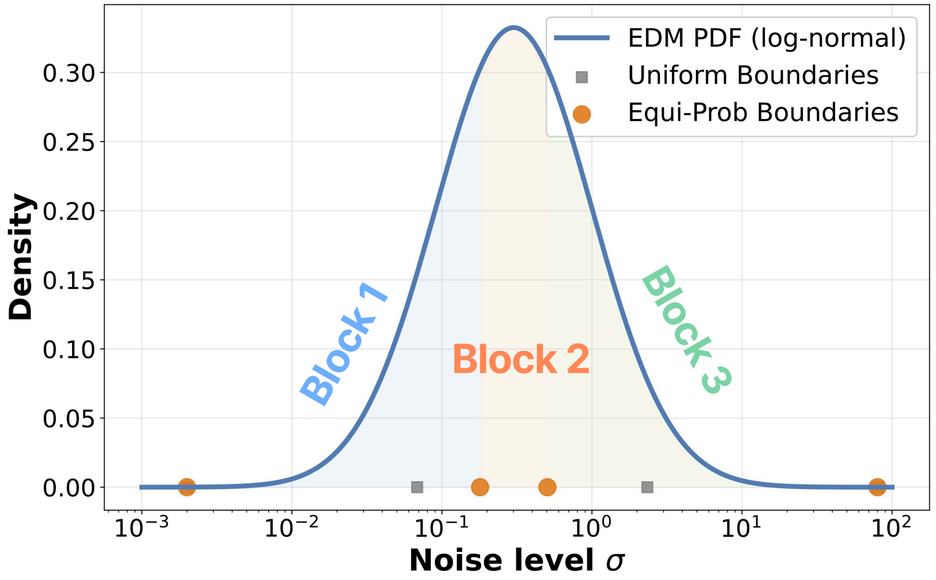
Equi-probability partitioning

Intermediate noise levels are most challenging!

Each block handles equal probability mass under the log-normal noise distribution.

✓ balanced parameter utilization

Log-normal noise distribution with B=3 blocks



Algorithm: Training

See Figure 3 and Figure 5 for more details

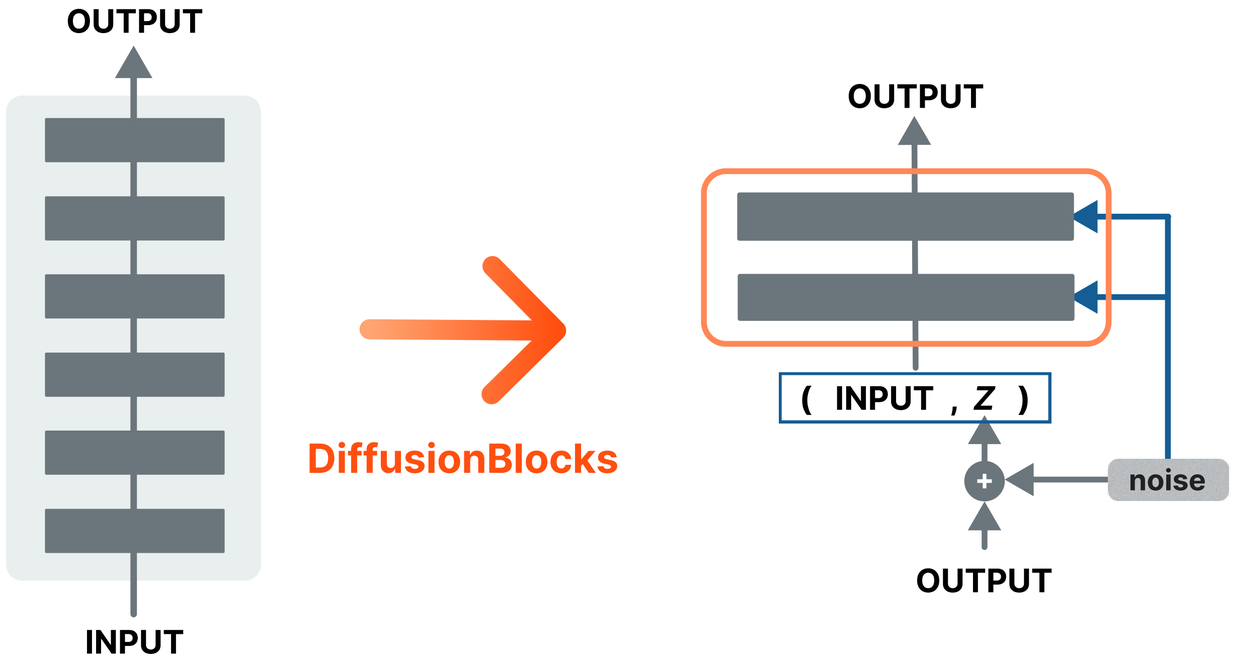
Training

One block at a time — only L/B layers require gradients

1. Sample one block
2. Sample noise from its range
3. Denoise: predict y from (x, z)
4. Compute diffusion loss
5. Update only block params

B× memory reduction

Gradients for only 1 block at a time



Algorithm: Inference

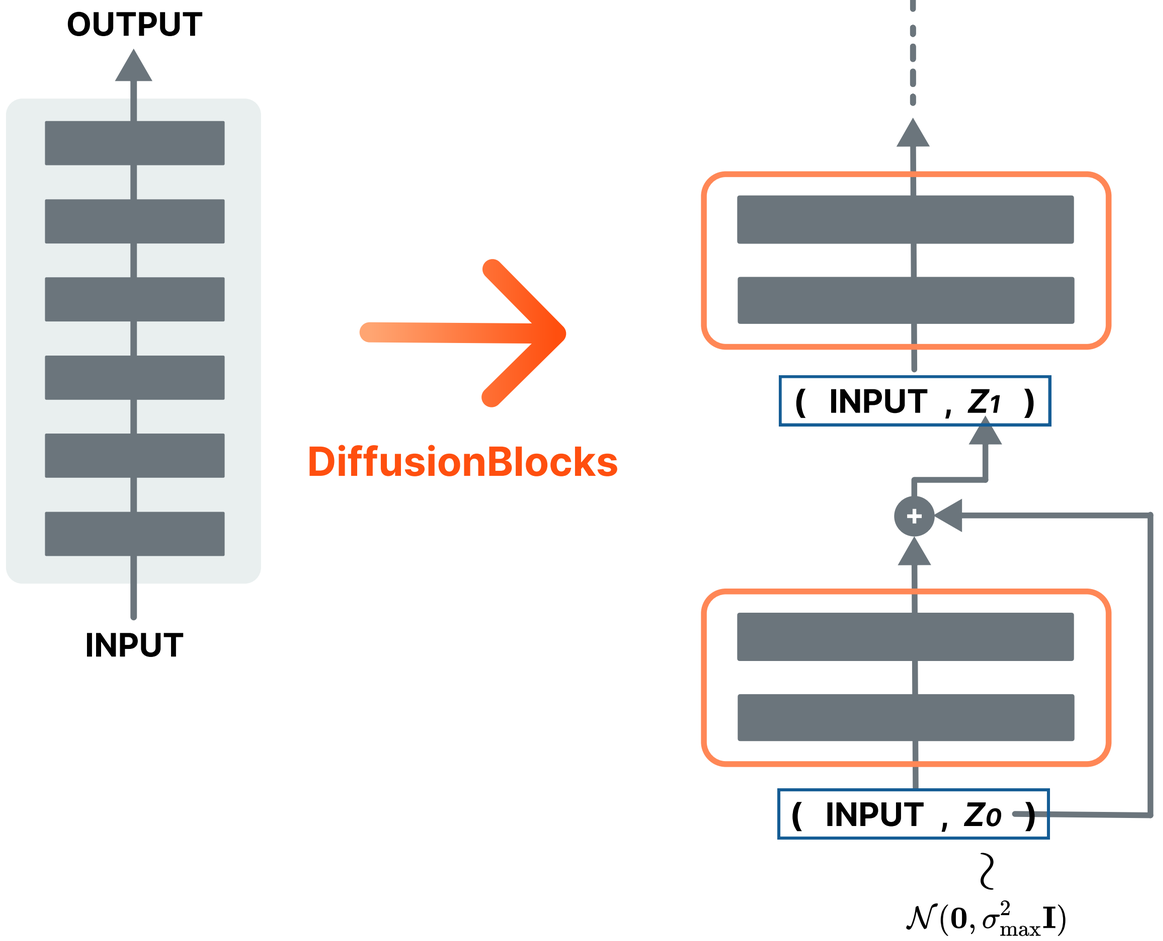
See Figure 3 and Figure 6 for more details

Inference

Sequential blocks from pure Gaussian noise

1. Start from pure Gaussian noise
2. For each noise level:
   Select relevant block
   Apply Euler step
3. Output the last denoised result

For diffusion models: **B× inference speedup**
(only 1 block per denoising step)



Results across 4 architectures

Image Classification ViT on CIFAR-100

Method	Accuracy ↑
ViT (end-to-end)	60.25
+ DiffusionBlocks	59.30

Image Generation DiT on ImageNet 256×256

Method		FID ↓
DiT (end-to-end)		12.09
+ DiffusionBlocks	3x faster 🔥 →	10.63

Text Generation Masked Diffusion on text8

Method	BPC ↓
MD4 (end-to-end)	1.56
+ DiffusionBlocks	1.45

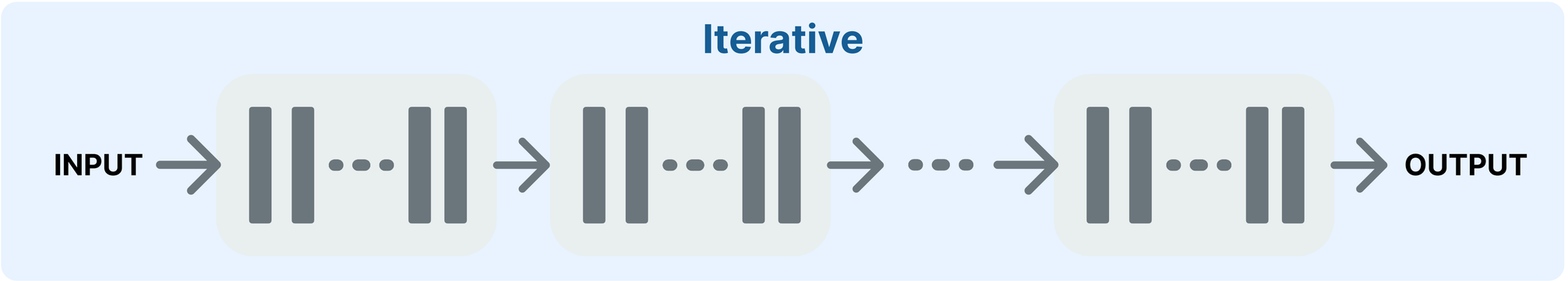
Text Generation AR Transformer on OWT

	MAUVE ↑
AR (end-to-end)	0.85
+ DiffusionBlocks	0.82

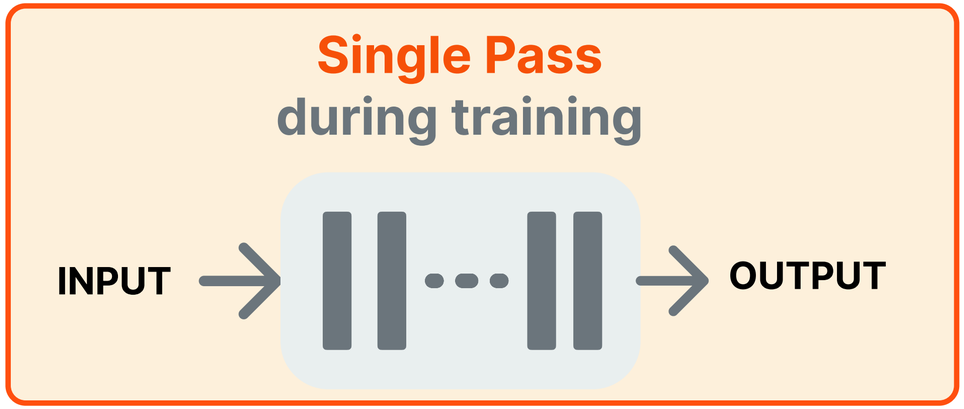
- **4 diverse architectures** across classification and generation tasks
- Competitive with end-to-end training **using only one block's gradients at a time**

Beyond Block-wise: Recurrent-Depth Models

- **Recurrent-depth models apply the same network multiple times iteratively,**
  suffering from expensive *BPTT* (backpropagation through time) during training.



DiffusionBlocks



Text Generation Huginn on LM1B

	MAUVE ↑
Huginn (Geiping et al., 2025)	0.49
+ DiffusionBlocks	0.70

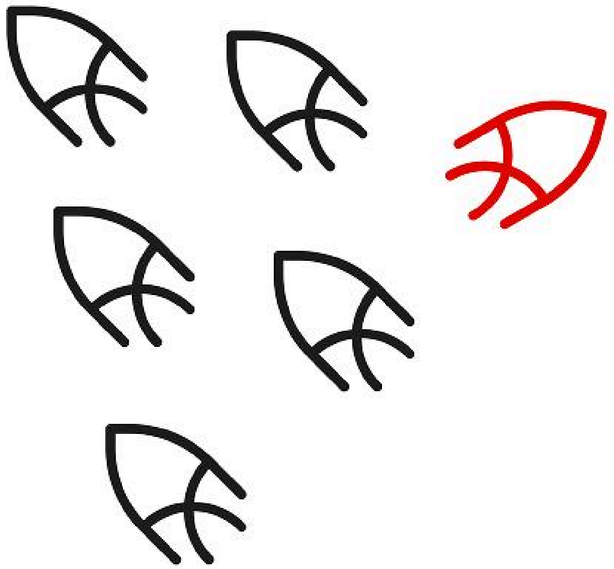
Summary

1. **Theoretically grounded block-wise training** derived from the denoising score matching framework

2. **Broadly applicable**: 5 architectures (ViT, DiT, AR, MDM, recurrent-depth) across classification and generation

3. **B× memory reduction** during training with competitive performance

4. Additional benefits: **B× inference speedup** for diffusion models, **single-pass training** for recurrent-depth



sakana.ai