Temporal Dynamic Quantization for Diffusion Models

Junhyuk So* [1], **Jungwon Lee*** [1], Daehyun Ahn [2], Hyungjun Kim [2] and Eunhyeok Park [1]

[1] Pohang University of Science and Technology
[2] SqueezeBits Inc

POSTECH

Diffusion Models

SDXL : Stable Diffusion XL [1]

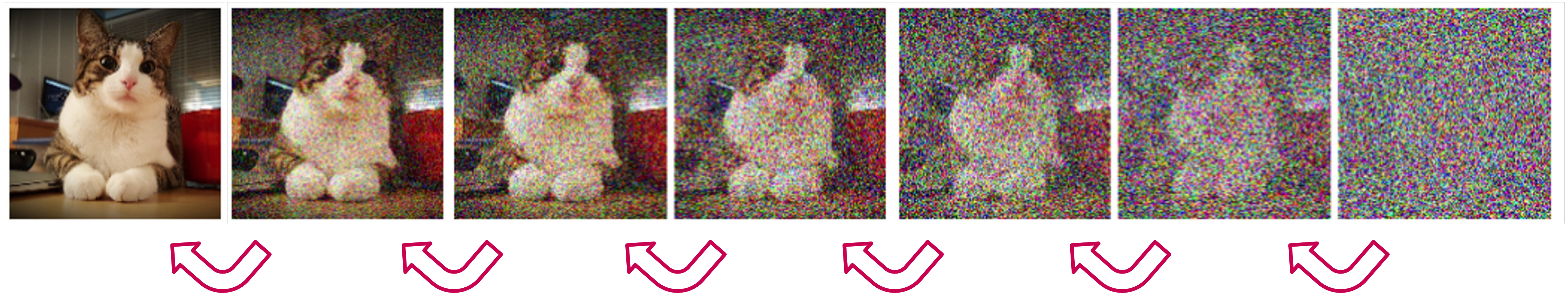
- Recently, Diffusion models have gained popularity due to its remarkable performance.

[1] Podell, Dustin, et al. "Sdxl: Improving latent diffusion models for high-resolution image synthesis."

Mechanism of Diffusion Model

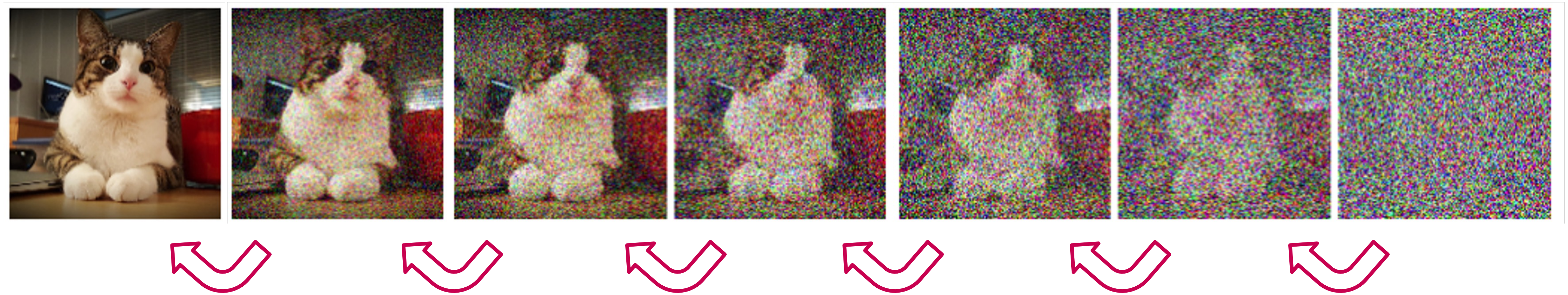
- Diffusion model is **denoising** model. It removes small amount of noise from noisy image.

Mechanism of Diffusion Model



- Diffusion model is **denoising** model. It removes small amount of noise from noisy image.
  - By iteratively denoising from pure noise, we can generate new image.

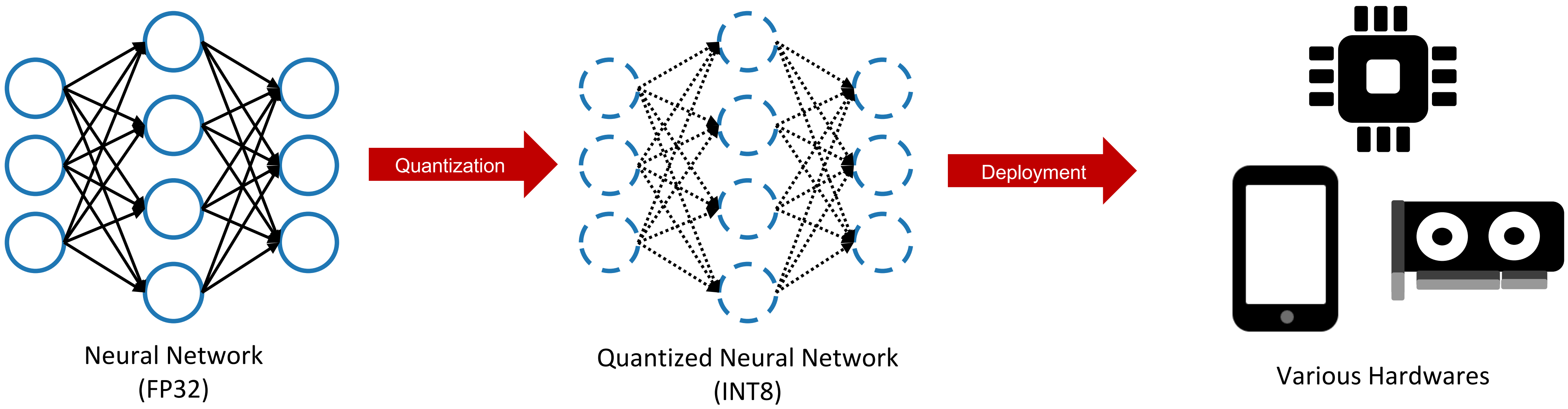
Mechanism of Diffusion Model



- Diffusion model is **denoising** model. It removes small amount of noise from noisy image.
  - By iteratively denoising from pure noise, we can generate new image.

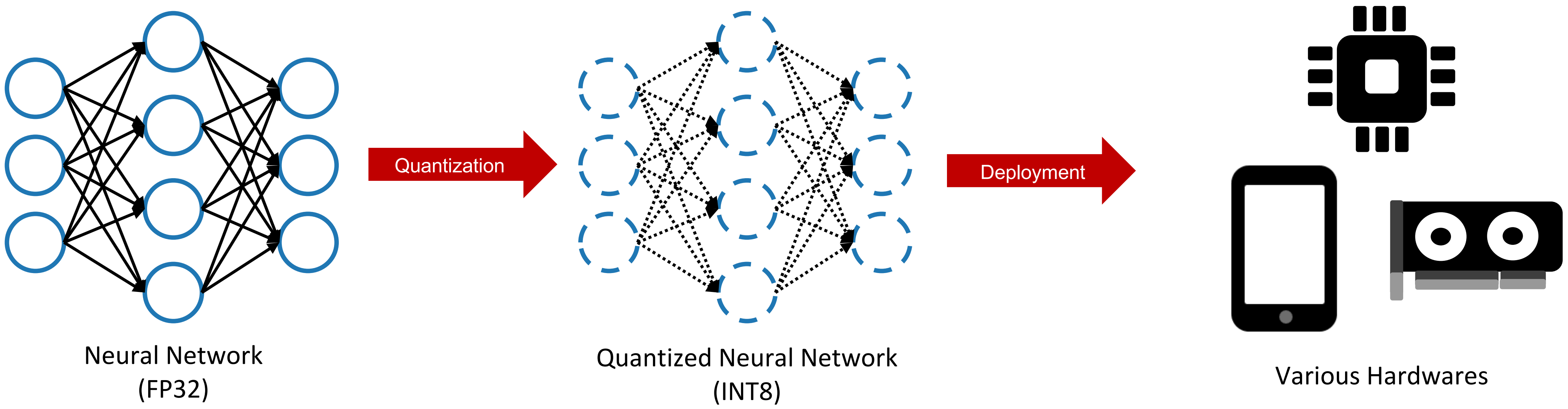
- **Problem** : Diffusion model is **too slow** because it requires **hundreds** of denoising steps for generation.

Quantization



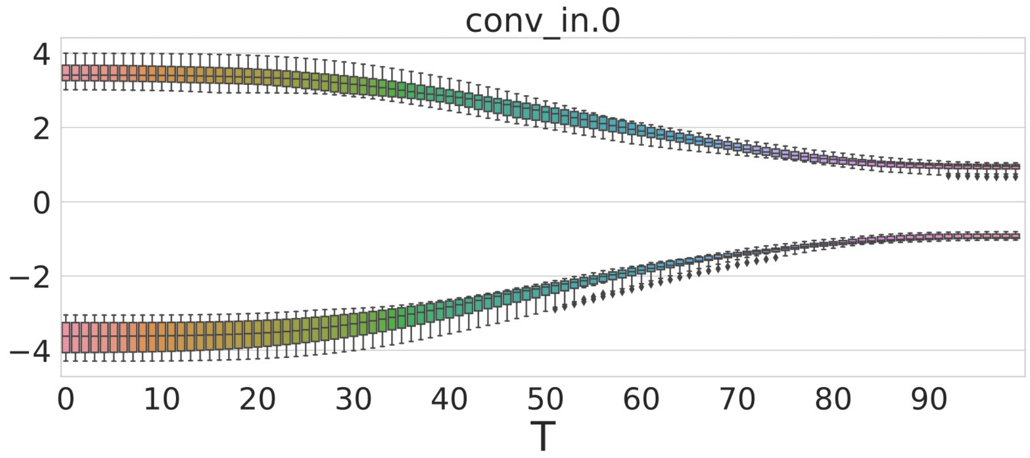
- **Quantization** is one of the most widely adopted optimization techniques.

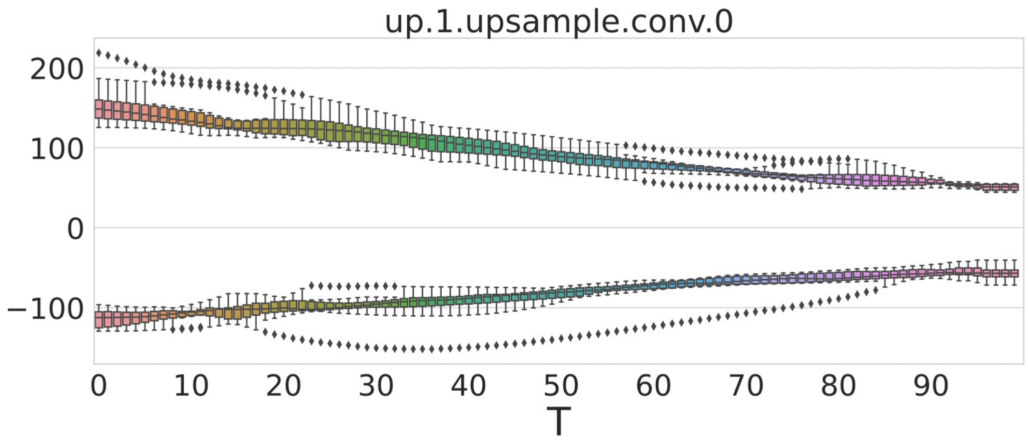
Quantization



- **Quantization** is one of the most widely adopted optimization techniques.
  - Activations and weights are stored in a **low-precision domain.**
  - Reduce memory usage & enable acceleration.

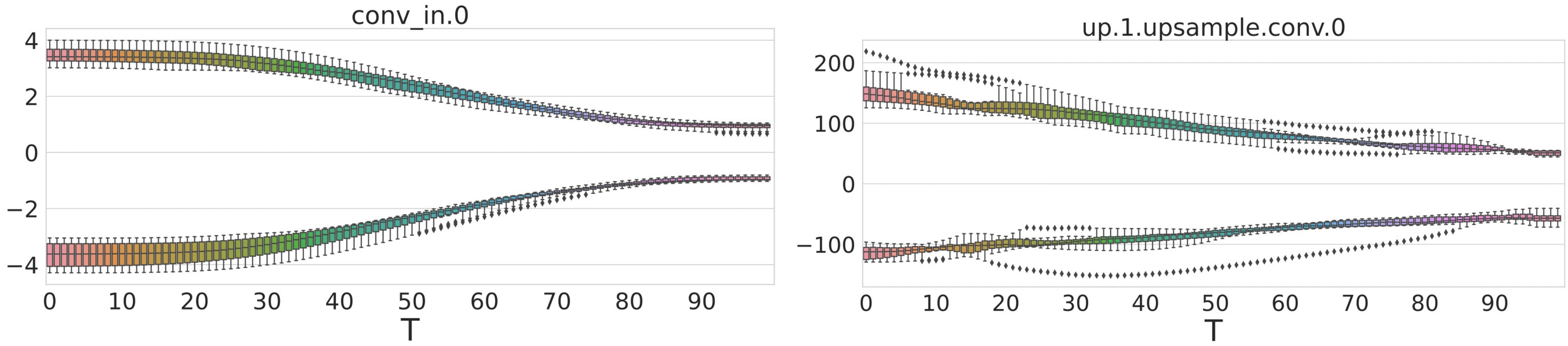
Diffusion Models are Hard to Quantize





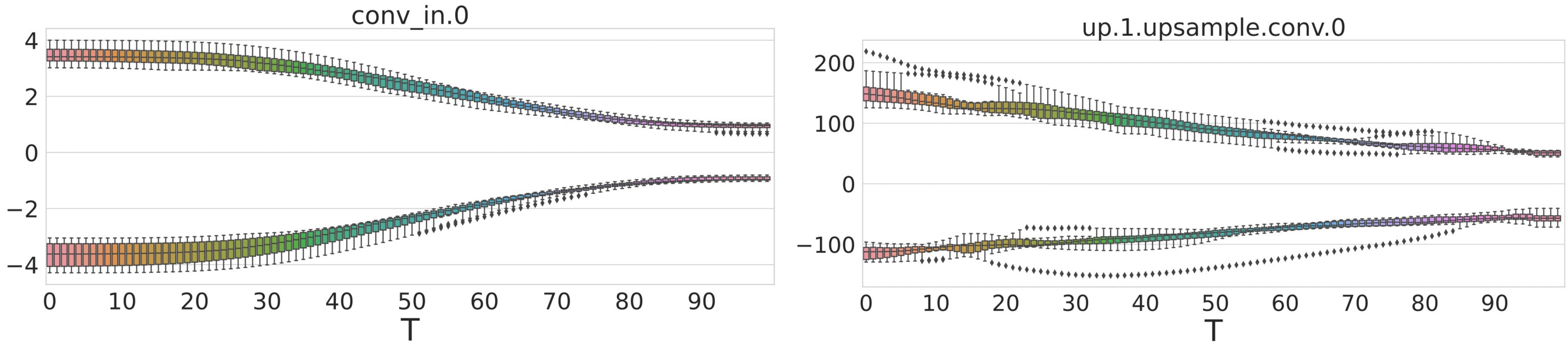
- However, applying quantization to diffusion models is known to be very challenging.

Diffusion Models are Hard to Quantize



- However, applying quantization to diffusion models is known to be very challenging.
  - We discovered that this is due to **unique property of diffusion model**'s denoising process.

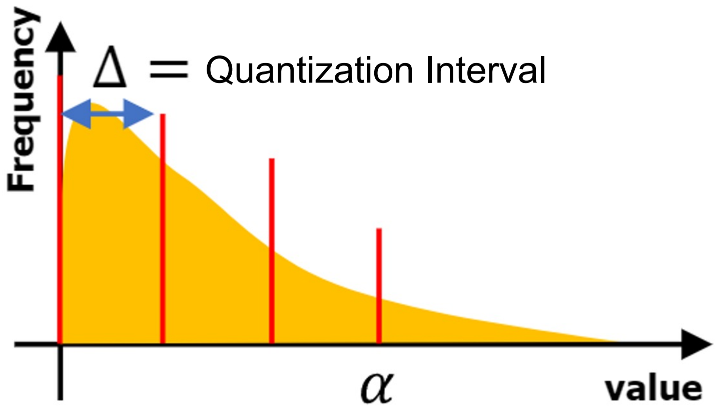
Diffusion Models are Hard to Quantize



- However, applying quantization to diffusion models is known to be very challenging.
  - We discovered that this is due to **unique property of diffusion model**’s denoising process.

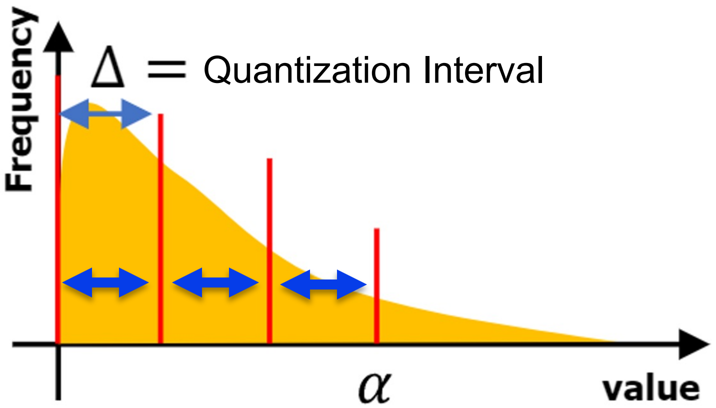
- Activation distribution of each layer **varies significantly** depending on the time step.

Error Source of Quantization



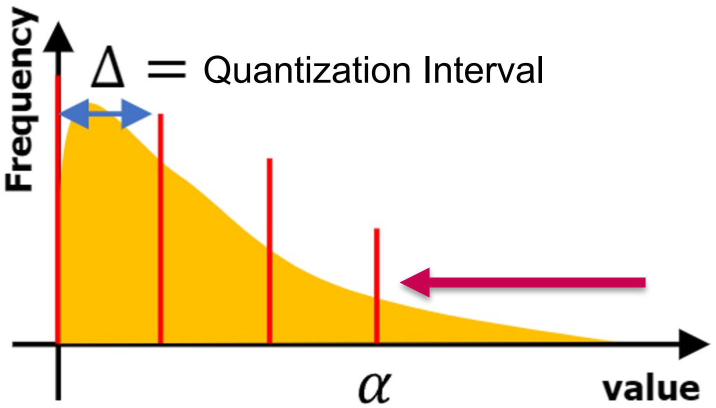
■ There are two types of error source in quantization.

Error Source of Quantization



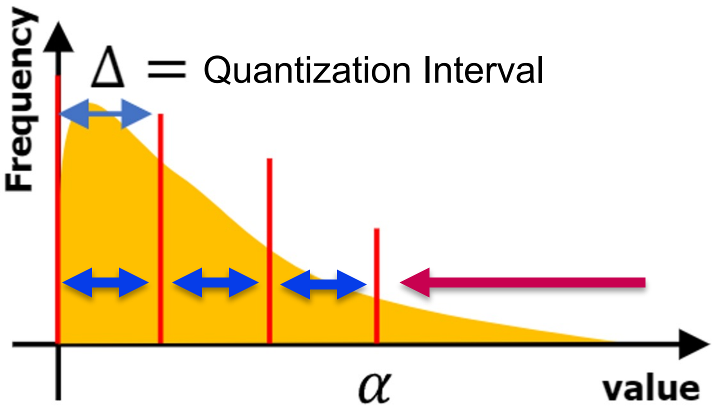
- There are two types of error source in quantization.
  - **Rounding Error** : Values within quantization range are mapped to the nearest quantization bin.

Error Source of Quantization



- There are two types of error source in quantization.
  - **Truncation Error** : Values greater than the last quantization bin are truncated to it.

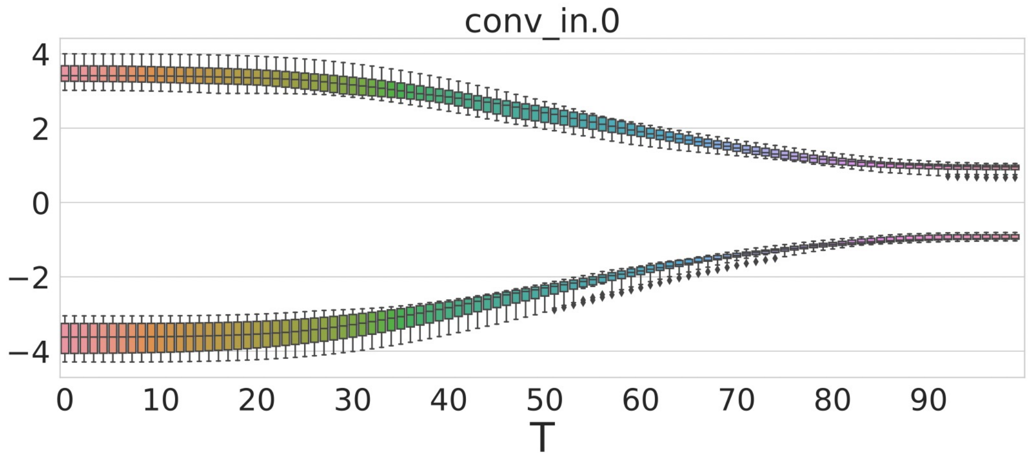
Error Source of Quantization



- There are two types of error source in quantization.
  - **Rounding Error** : Values within quantization range are mapped to the nearest quantization bin.
  - **Truncation Error** : Values greater than the last quantization bin are truncated to it.

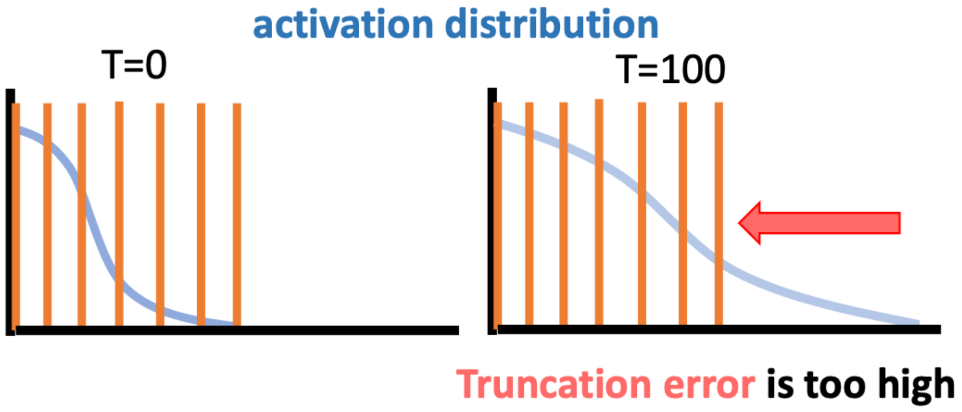
- There is **trade-off** between these two error sources.

Diffusion Models are Hard to Quantize



- In this case, static quantizer cannot handle **Quantization Error Trade-off** effectively.

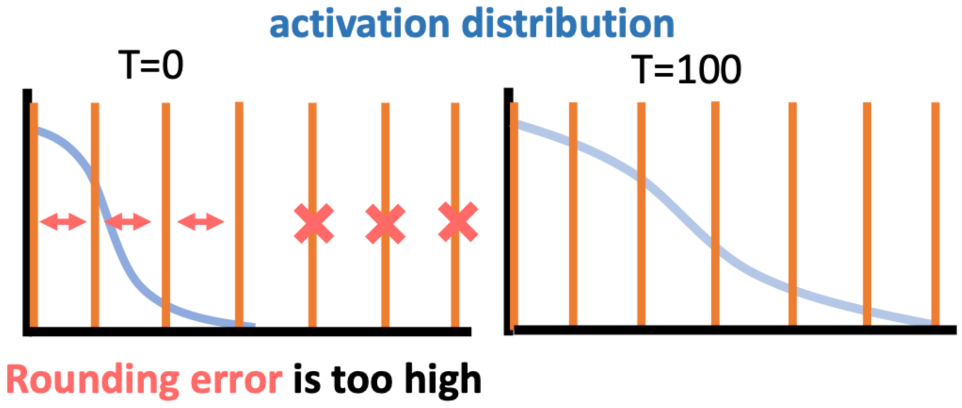
Diffusion Models are Hard to Quantize



(a) The interval is calibrated at T=0

- In this case, static quantizer cannot handle **Quantization Error Trade-off** effectively.
  - Calibrating Quantizer to T=0 → **Large Truncation error** when T=100

Diffusion Models are Hard to Quantize

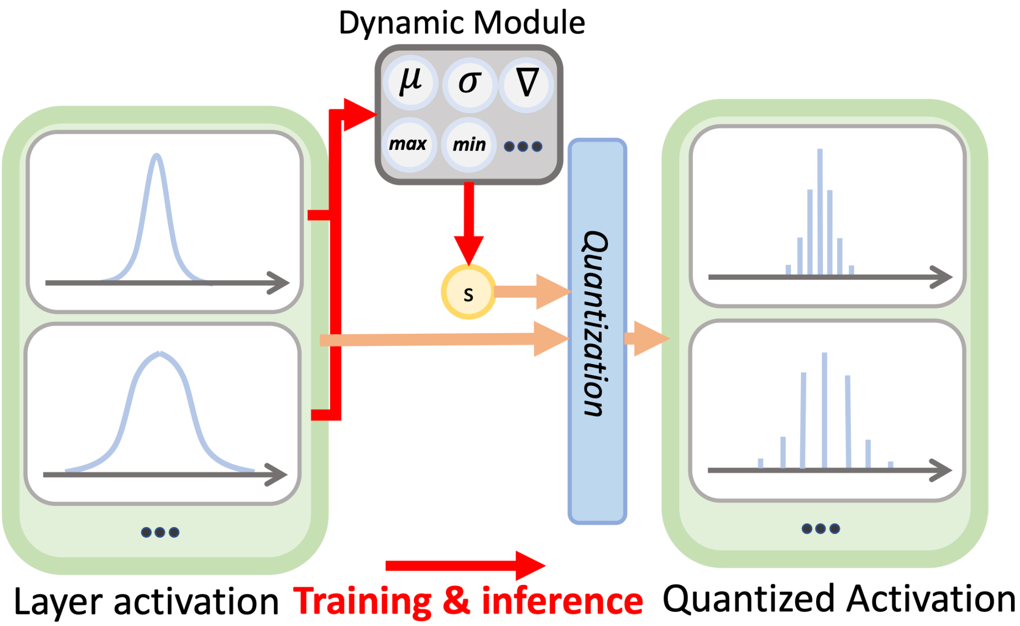


(b) The interval is calibrated at T=100

- In this case, static quantizer cannot handle **Quantization Error Trade off** effectively.
  - Calibrating Quantizer to T=100 → **Large Rounding error** when T=0

Dynamic Quantization

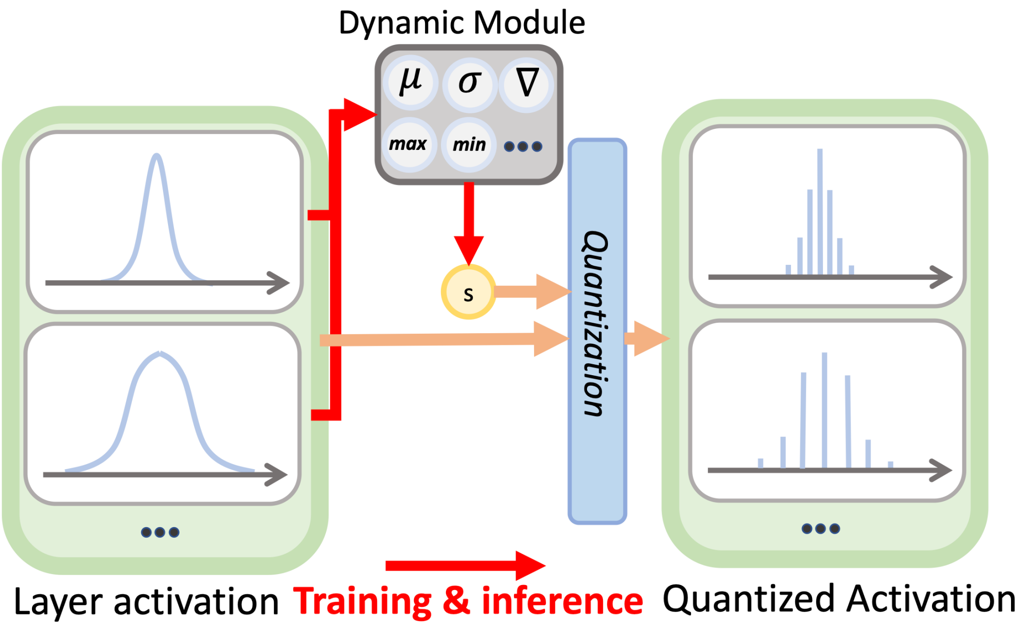
- Solution : Dynamic Quantization ?



- One easy solution is using **Input-dependent dynamic quantization**.

Dynamic Quantization

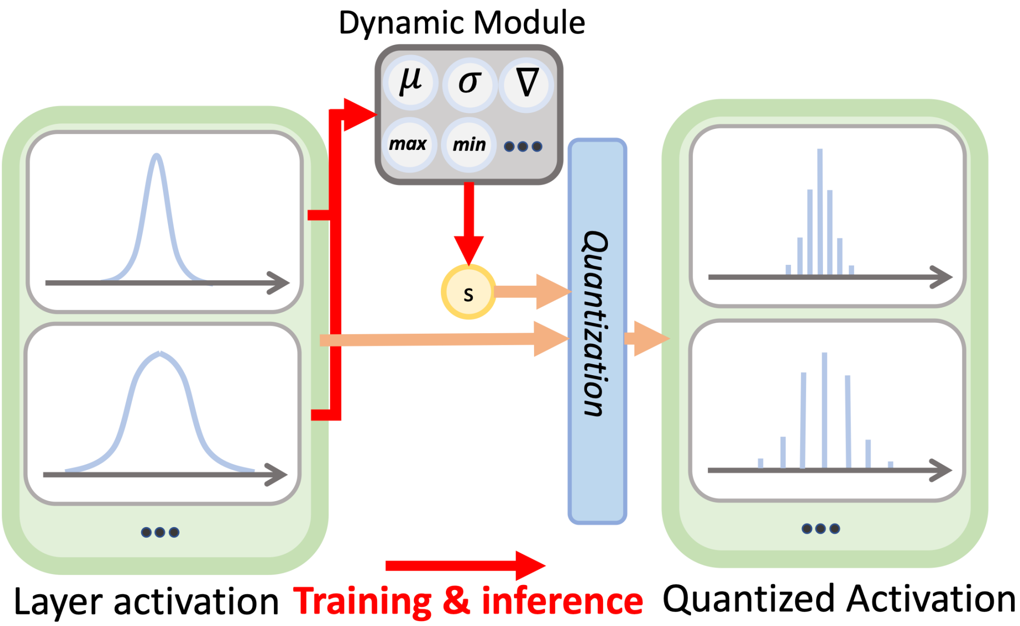
- Solution : Dynamic Quantization ?



- One easy solution is using **Input-dependent dynamic quantization**.
  - It generates quantization interval based on **input statistics**, such as *min,max,var*.

Dynamic Quantization

■ Solution : Dynamic Quantization ?

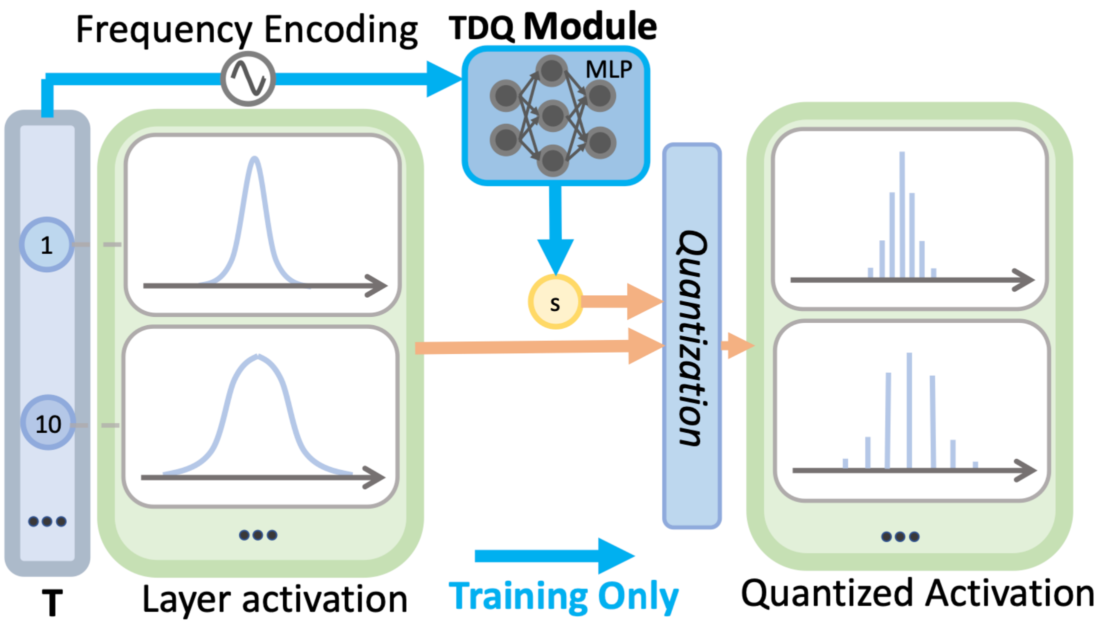


■ One easy solution is using **Input-dependent dynamic quantization**.

- It generates quantization interval based on **input statistics**, such as *min,max,var*.
- However, process of gathering these statistics introduces **significant overhead** in inference.

Ours : Temporal Dynamic Quantization

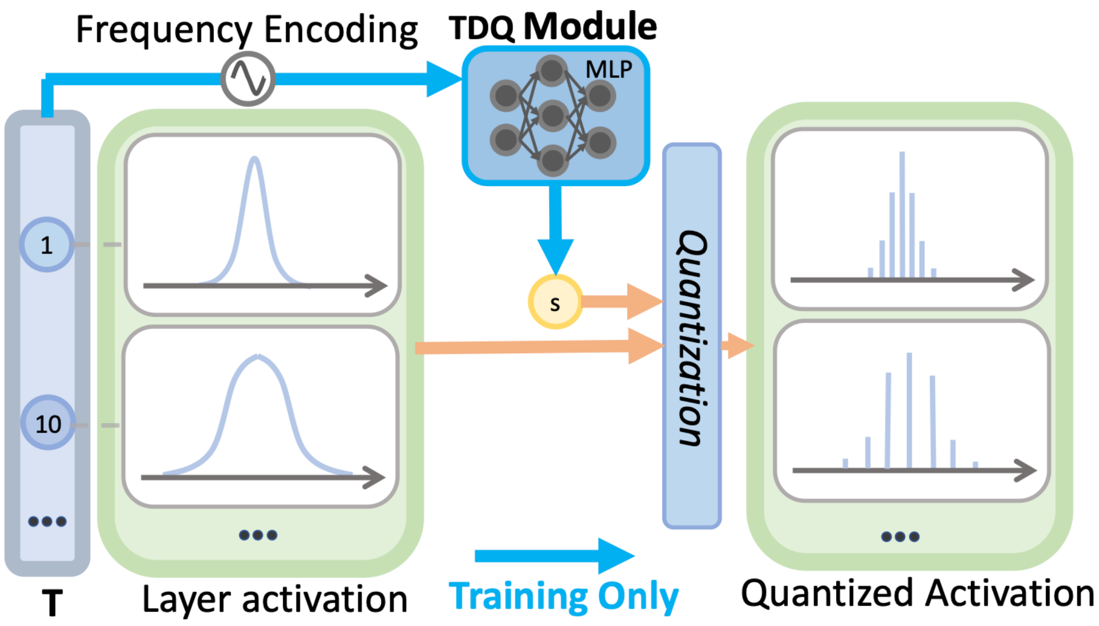
- Our Solution :



- Instead, we propose our method : **Temporal Dynamic Quantization**

Ours : Temporal Dynamic Quantization

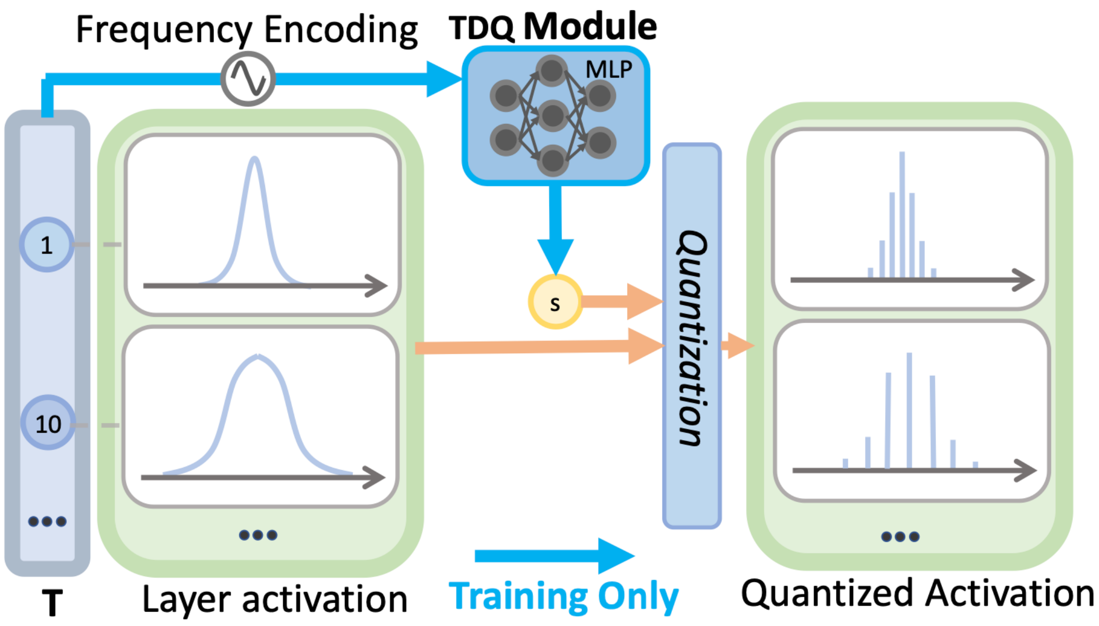
- Our Solution :



- Instead, we propose our method : **Temporal Dynamic Quantization**
- Unlike dynamic quantization, we only use **temporal information** rather than input activation statistics.

Ours : Temporal Dynamic Quantization

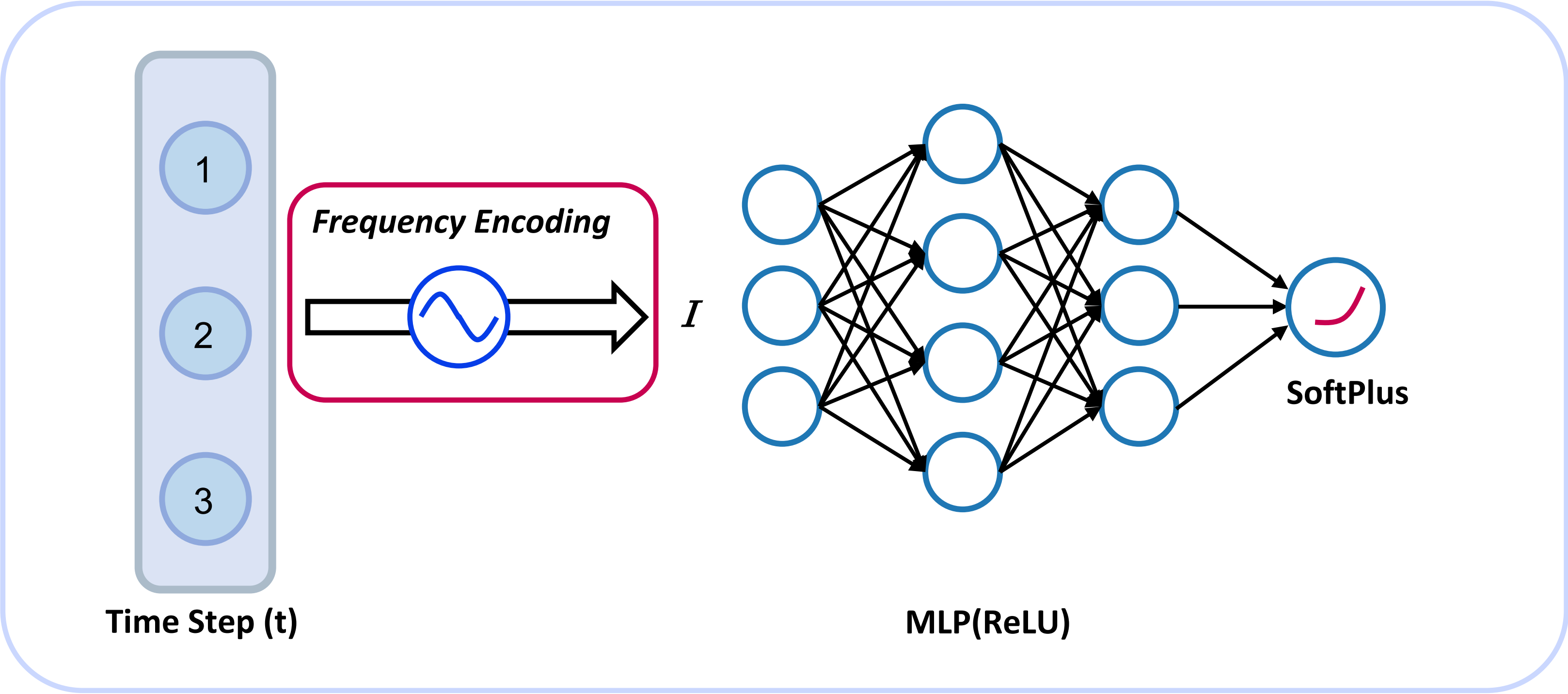
- Our Solution :



- Instead, we propose our method : **Temporal Dynamic Quantization**
- Unlike dynamic quantization, we only use **temporal information** rather than input activation statistics.
  - Since we can pre-compute quantization interval, our method incurs **no overhead**.

Implementation Details

TDQ

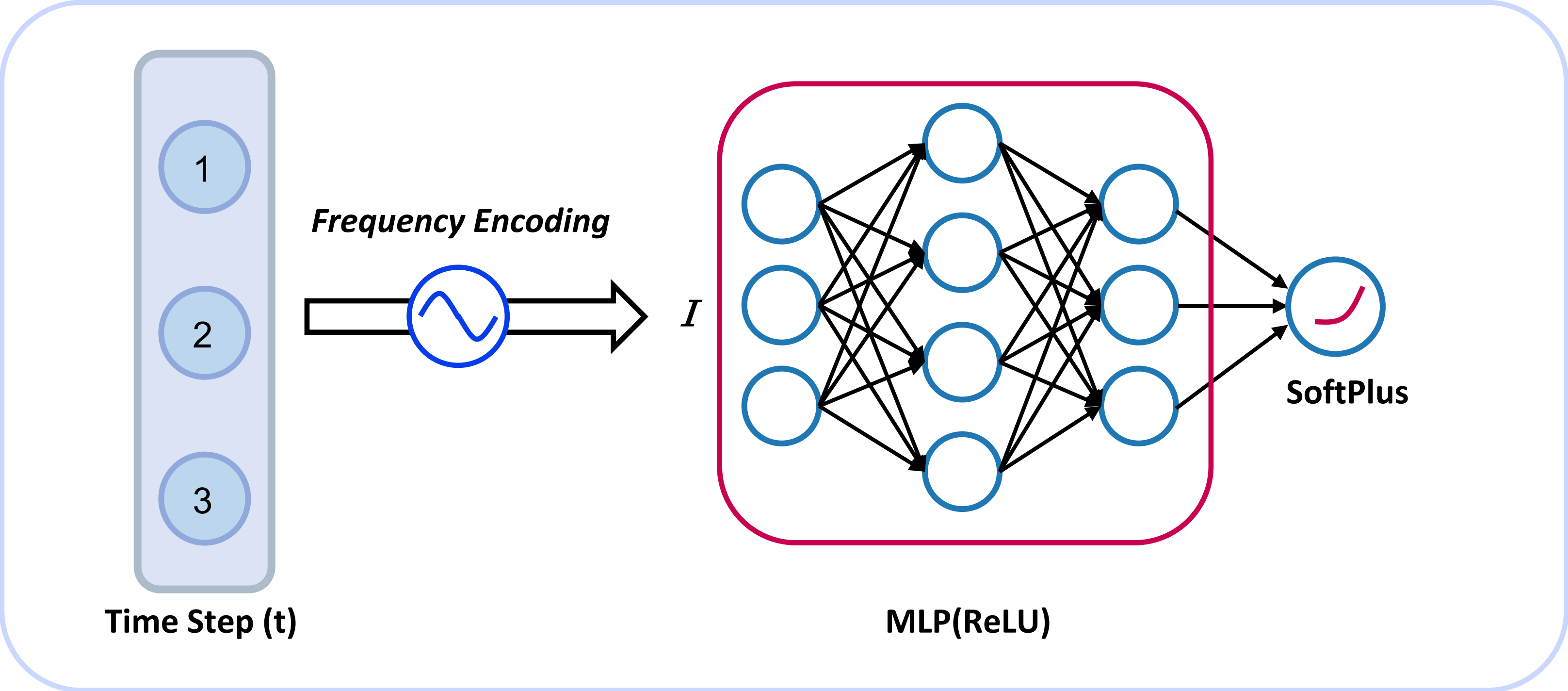


- In standard setting, our TDQ module has 3 components : **Frequency Encoding**
  - **Frequency Encoding** : We use **Geometric Fourier Encoding** to inject high frequency components.

$$I = enc(t) = (sin(\frac{t}{t_{max}^{0/d}}), cos(\frac{t}{t_{max}^{0/d}}), sin(\frac{t}{t_{max}^{2/d}}), cos(\frac{t}{t_{max}^{2/d}}), ..., sin(\frac{t}{t_{max}^{d/d}}), cos(\frac{t}{t_{max}^{d/d}})),$$

Implementation Details

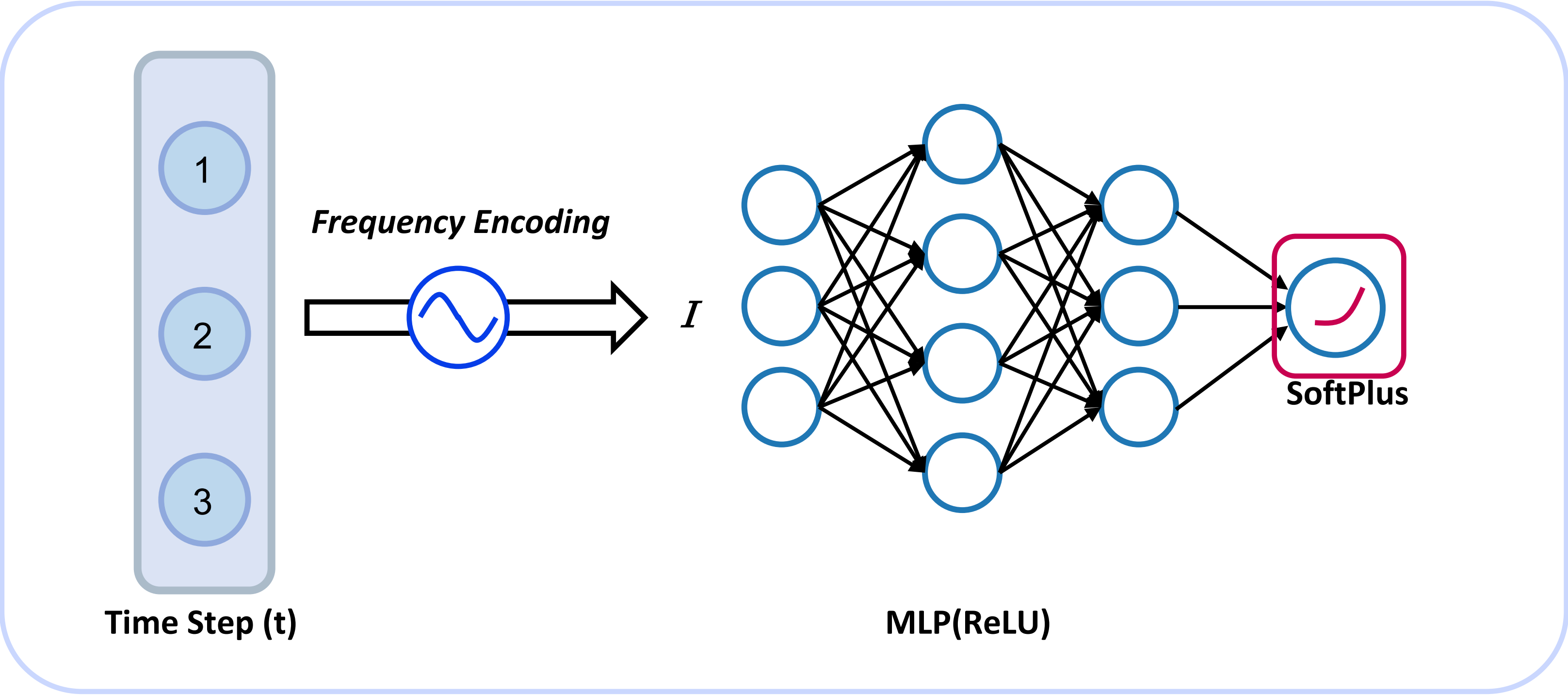
TDQ



- In standard setting, our TDQ module has 3 components : **Frequency Encoding , MLP**
  - **MLP** : MLP is trained to **predict optimal quantization interval** for each time step.
    - TDQ consists of 4 layer MLP with ReLU activation. (hidden dim 64)

Implementation Details

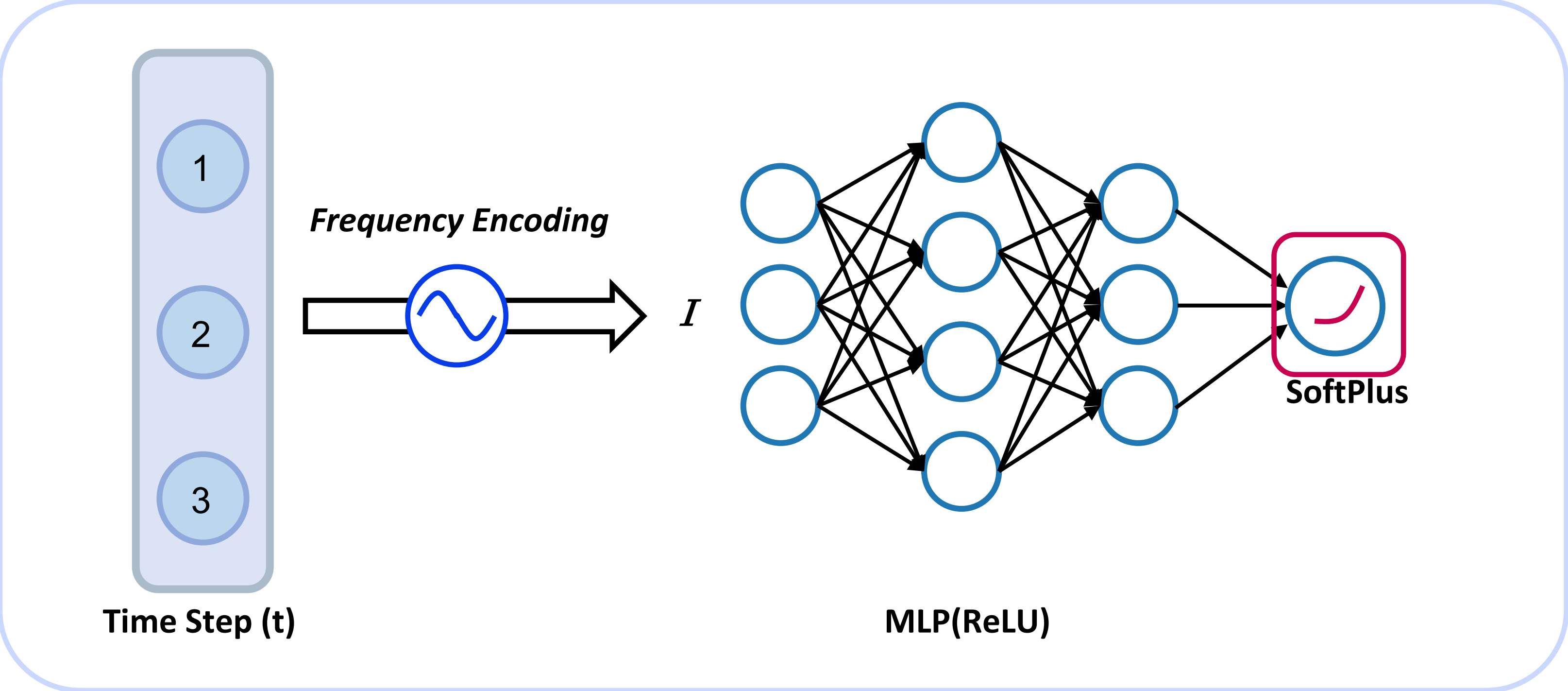
TDQ



- In standard setting, our TDQ module has 3 components : **Frequency Encoding, MLP, Softplus**
  - **SoftPlus** : SoftPlus function constrains data ranges to **non-negative value**.

Implementation Details

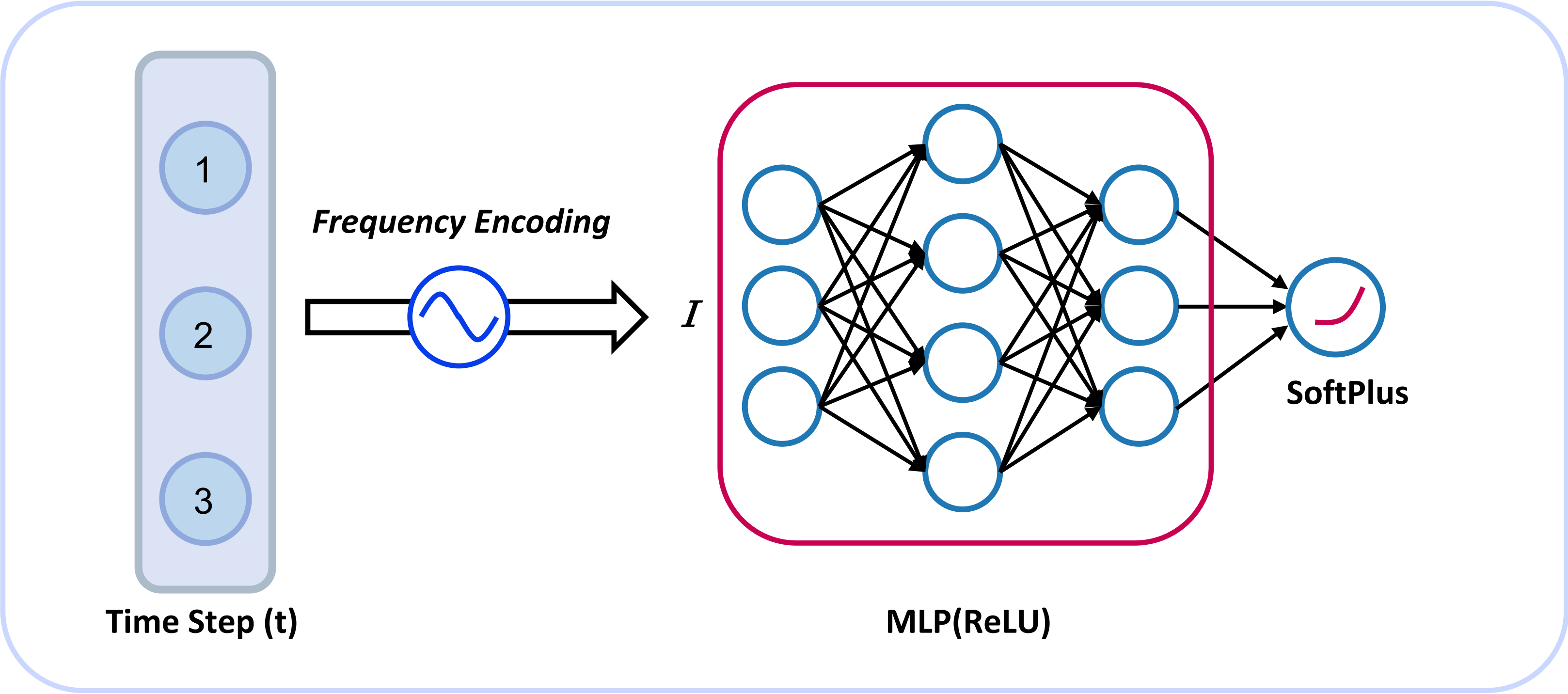
TDQ



- In standard setting, our TDQ module has 3 components : **Frequency Encoding, MLP, Softplus**
- Every part of TDQ module is **differentiable**.
  - TDQ module can be trained to minimize quantization error by using **gradient descent.**

Implementation Details

TDQ

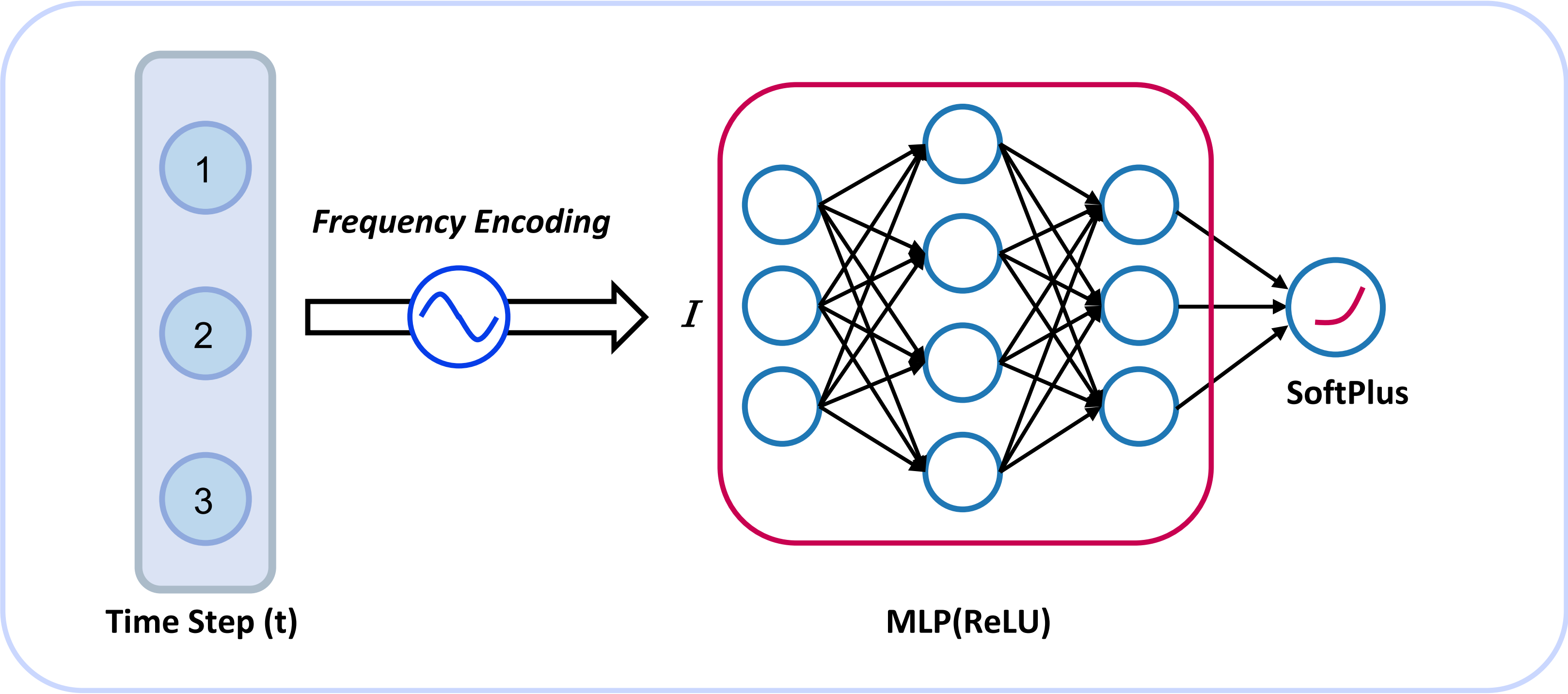


- However, **in PTQ**, standard setting was prone to **overfitting** due to two reason.

  1) **Limited calibration dataset (typically 256 samples)** makes it challenging to train standard TDQ.

Implementation Details

TDQ

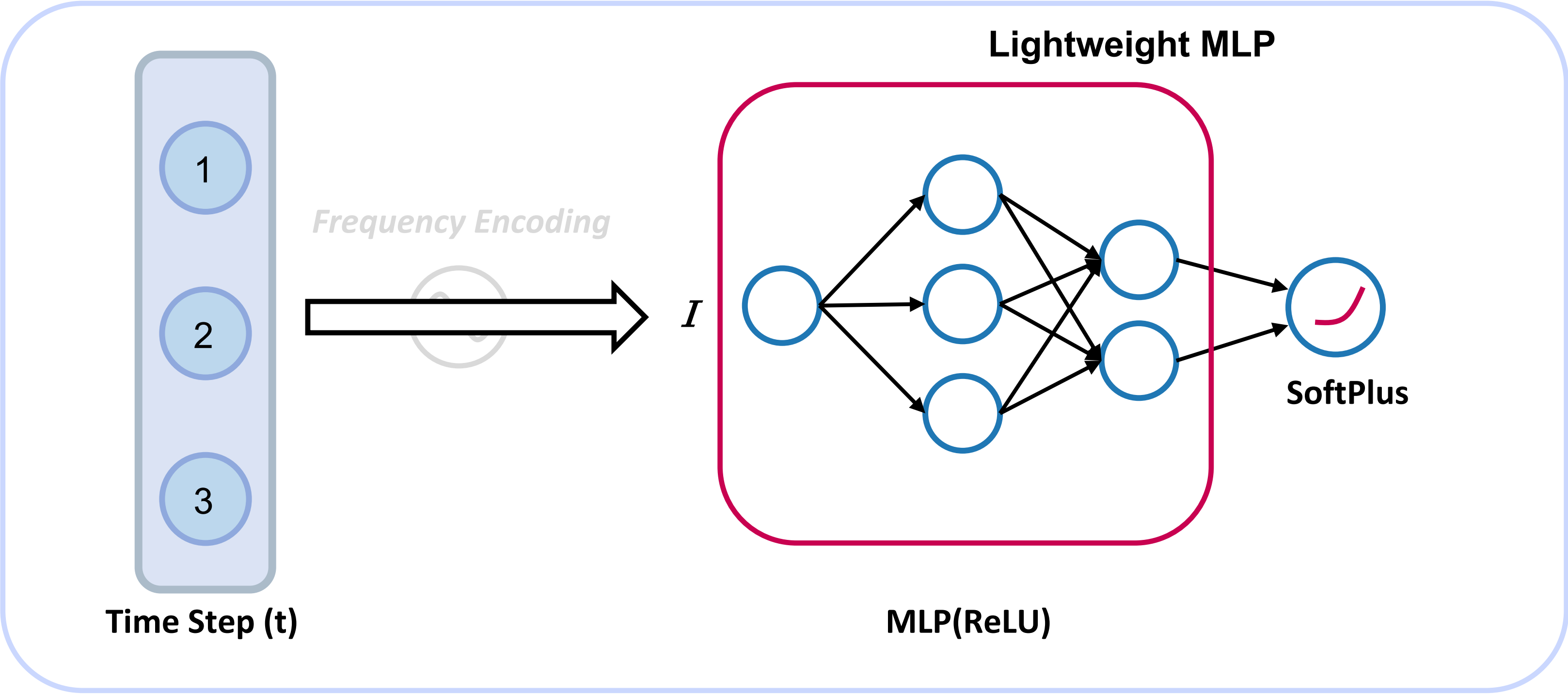


- However, **in PTQ**, standard setting was prone to **overfitting** due to two reason.

  2) The **relatively brief training iteration** make it hard to filter out the high-frequency component.

Implementation Details

TDQ_{thin}



- However, **in PTQ**, standard setting was prone to **overfitting** due to two reason.
  - To mitigate these constraints, we introduced a **streamlined version of TDQ**, referred to as TDQ_{thin}.
  - This refined module uses a 3-layer MLP with a mere 16 hidden dimensions and omits the frequency encoding for time steps.

Experimental Results

- Quantization Aware Training (QAT)

LSUN-Churches

(FID)	W8A8	W4A8	W8A4	W4A4	W3A3
PACT [2]	9.20	9.94	8.59	10.35	12.95
LSQ [3]	-	4.92	5.08	5.06	7.21
Ours	**3.87**	**4.04**	**4.86**	**4.64**	**6.57**

W4A8

Ours

W4A4

Ours

- TDQ gives **substantial quality improvement**, and benefit becomes even larger in lower precision.

[2] Choi, Jungwook, et al. "Pact: Parameterized clipping activation for quantized neural networks."
[3] Esser, Steven K., et al. "Learned step size quantization."

Experimental Results

- Post Training Quantization (PTQ)

Ours

LSQ[3]

(FID)	W8A8	W8A6	W8A5
Min-Max	4.34	103.15	269.05
PTQ4DM [4]	3.97	4.26	7.06
Ours	**3.89**	**4.24**	**4.85**

- TDQ also shows performance improvement in PTQ.
- Our method can be applicable to any quantization pipeline seamlessly.

[4] Shang, Yuzhang, et al. "Post-training quantization on diffusion models."

Experimental Results

- Post Training Quantization (PTQ)

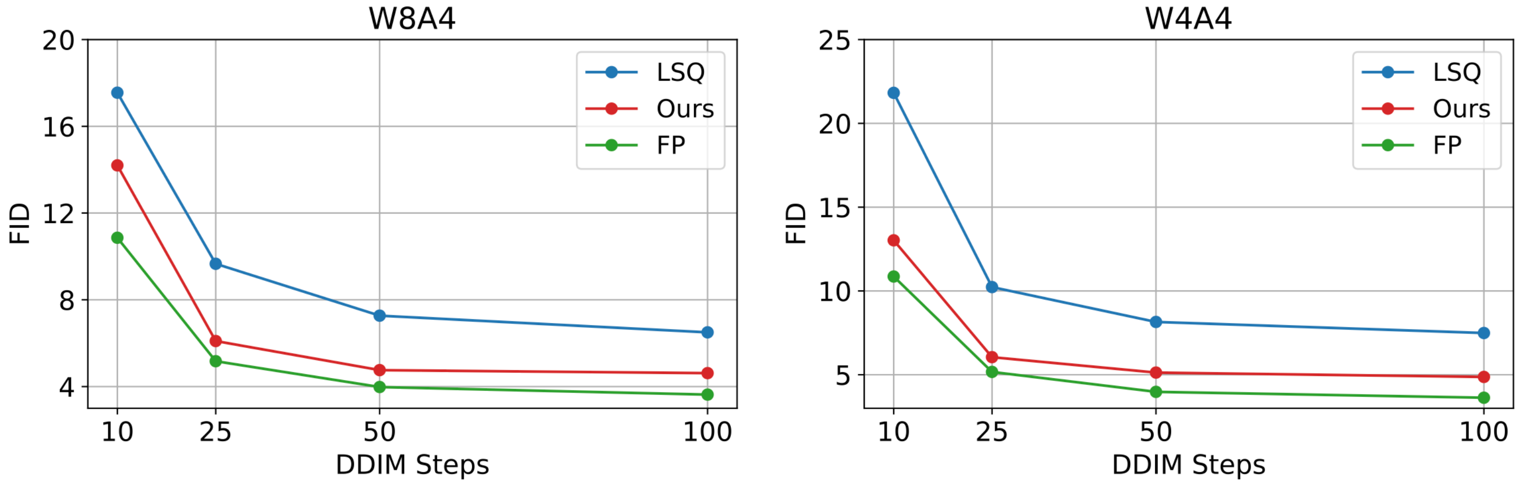
(FID)	Churches W4A8	Churches W4A6	ImageNet W4A6
Baseline [5]	76.36	158.07	47.26
TDQ	44.48	120.53	41.23
$\boldsymbol{TDQ_{thin}}$	**28.74**	**55.27**	**16.96**

- Even at **low precision weights**, TDQ shows performance improvement over baseline.
- Additionally, TDQ_{thin} outperforms all these cases.

POSTECH

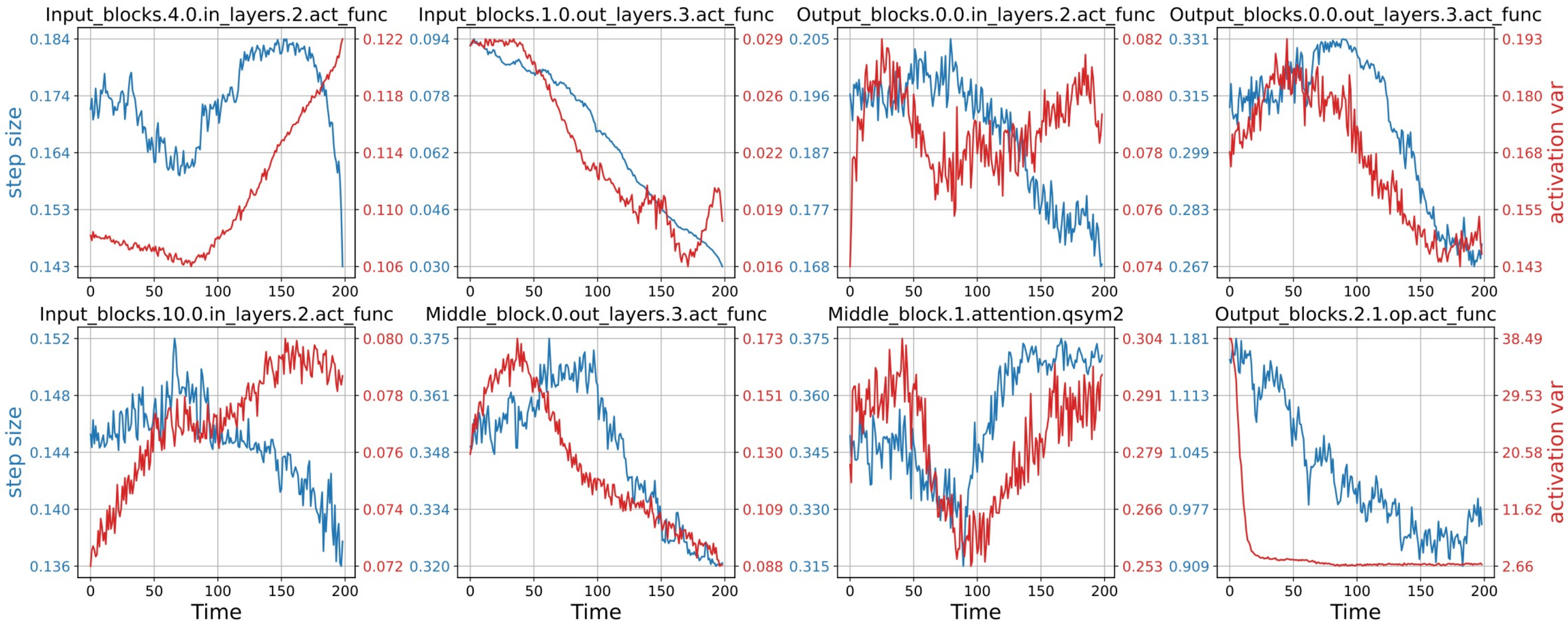
[5] Li, Xiuyu, et al. "Q-diffusion: Quantizing diffusion models."

Generalization Performance



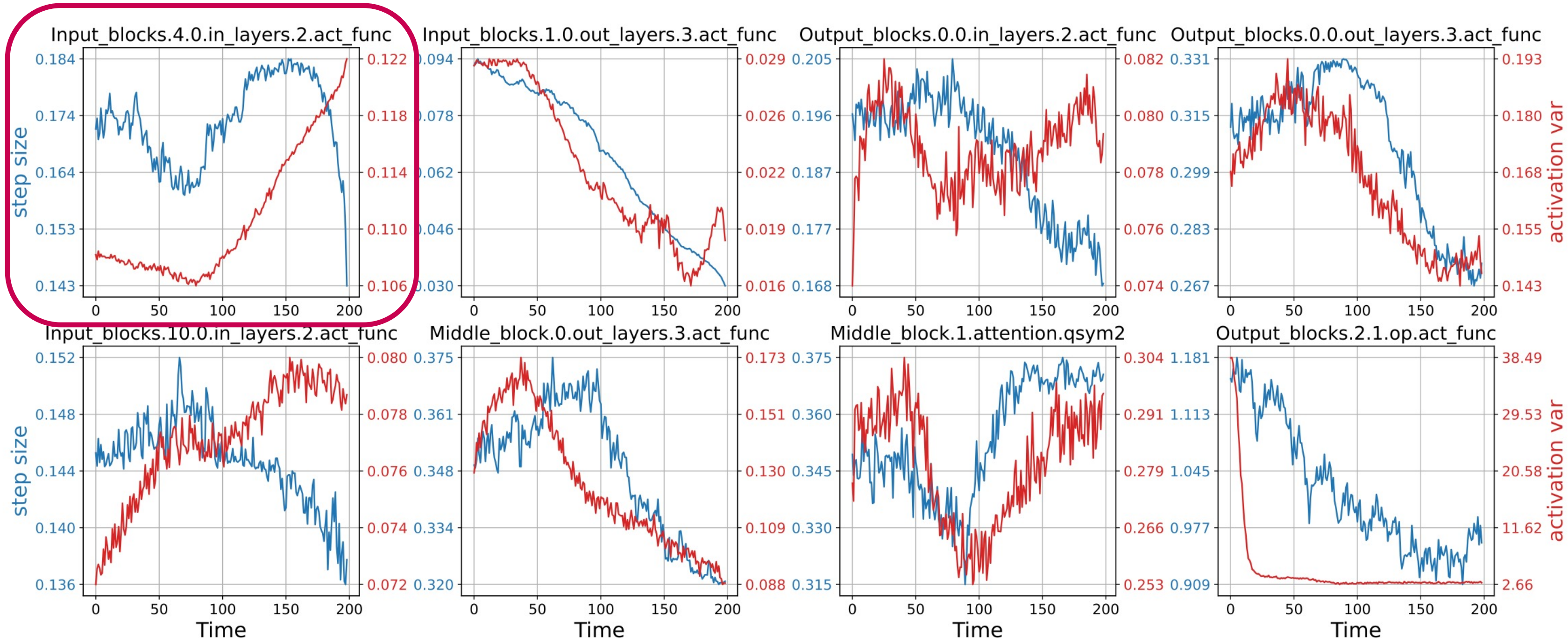
- Sampling process is usually executed in fewer time step (10 ~ 50) than training (1000).
- TDQ's performance declines similarly to the **FP baseline**, while LSQ's performance deteriorates as the number of sampling step decreases.

TDQ Output Dynamics



- **Blue : Predicted Quantization Interval** , **Red : Variance of Activation**
- In most cases, TDQ's output dynamics show alignment with variation.

TDQ Output Dynamics



- **Blue : Predicted Quantization Interval** , **Red : Variance of Activation**
- Few layers show different tendency :
  These indicate that TDQ module is attempting to minimize final task error, not layer quantization error.

Thank You