Large Language Models (Auto-regressive and Diffusion-based)

CS772A: Probabilistic Machine Learning

Piyush Rai

Large Language Models (LLM)

- An LLM defines a probability distribution over sequences of tokens

$$\boldsymbol{x} = \{x_1, x_2, \dots, x_N\}$$

- Autoregressive modeling is a popular way to define this distribution

$$p(\boldsymbol{x}) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots \quad = \prod_{i=1}^{N} p(x_i|\boldsymbol{x}_{<i})$$

- Params $\boldsymbol{\theta}$ of each conditional $p(x_i|\boldsymbol{x}_{<i})$ defined using neural nets (e.g., transformer)

$$p_\theta(x_i|\boldsymbol{x}_{<i}) = \text{softmax}(f_\theta(\boldsymbol{x}_{<i}))$$

Vector of probabilities of all possible values of the next token

A neural net

Training of LLMs and Sequence Generation

- Usually trained using maximum likelihood with log-likelihood defined as

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} \log p_\theta(x_i|\boldsymbol{x}_{<i})$$

- Once trained, generate a sequence of tokens, one at a time. Some popular ways:
  - Greedy (pick the most probable token deterministically): $\hat{x}_i = \mathbf{argmax}\ p_\theta(x_i|\boldsymbol{x}_{<i})$
  - Sampling: $\hat{x}_i \sim\ p_\theta(x_i|\boldsymbol{x}_{<i})$
  - Temperature based sampling: $\hat{x}_i \sim [p_\theta(x_i|\boldsymbol{x}_{<i})]^{1/\tau}$
    - $\tau < 1$ sharpens the distribution (more deterministic sampling)
    - $\tau > 1$ flattens the distribution (more exploratory sampling)
  - Top-k sampling
    - Randomly sample a token from k most probable token
  - Nucleus (top-p) sampling
    - Sample from minimum set of tokens with cumulative probability $\geq p$

Some Limitations of Autoregressive LLMs

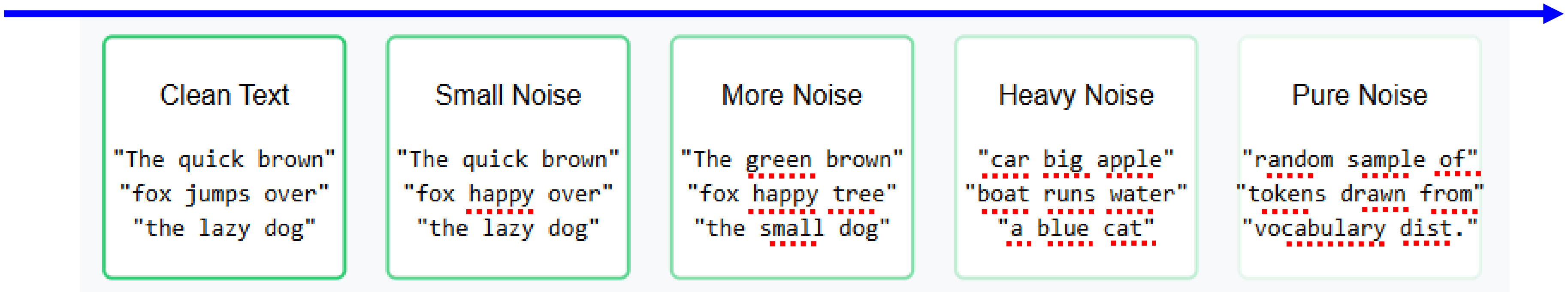
- Sequential Generation: Inherently slow due to token-by-token decoding
- Low Output Diversity: Because of the decoding techniques used
- Locally greedy generation and lacks long-term coherence control.
- Token-Level Objectives: Next-token prediction doesn't align well with task-level goals (e.g., factual consistency).
- Difficulty Handling Edits/Rewrites: Inefficient for tasks requiring partial edits or structured generation.

Diffusion based LLM*

- Autoregressive LLMs generate each token conditioned on earlier tokens

$$p(\boldsymbol{x}) = \prod_{i=1}^{N} p(x_i | x_{<i})$$

- In contrast, diffusion based LLM generate all tokens in parallel
- Diffusion LLM consist of a forward and a reverse process
- Forward process corrupts the token sequence gradually till it becomes pure noise



- Reverse process starts with pure noise and gradually denoises it to generates a token sequence

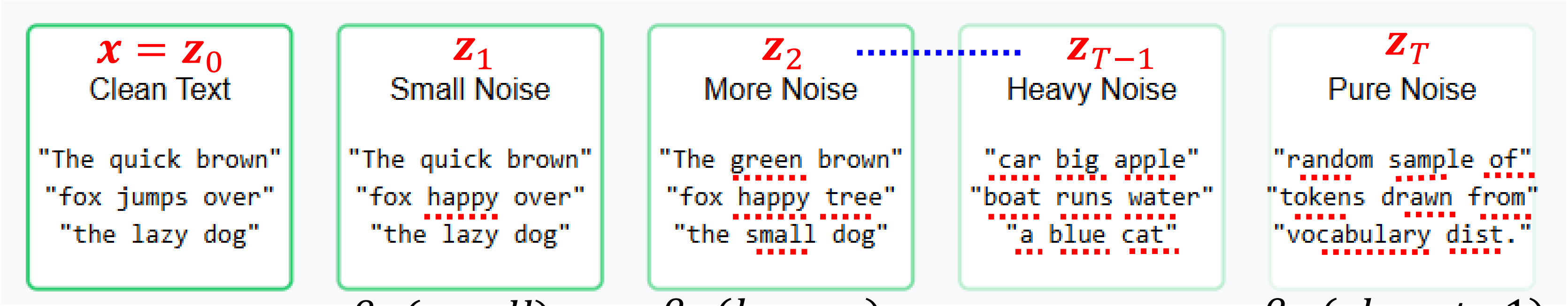
*Structured Denoising Diffusion Models in Discrete State-Spaces (Austin et al, NeurIPS 2021)
*Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions (Hoogeboom et al, NeurIPS 2021)

Forward Process

- Assuming $\boldsymbol{z}_t$ contains N tokens, the forward process in diffusion LLM can be defined as

$$q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \prod_{i=1}^{N} \text{Cat}(z_t^i|\boldsymbol{P}z_{t-1}^i)$$

$\boldsymbol{P}$ is the $V \times V$ transition matrix that defines corruption probabilities, z_t^i are one-hot vectors of size V where V is the vocab size



β_1 (small) $\quad \beta_2$ (larger) $\quad \beta_T$ (close to 1)

- A very simple yet popular form of the above corruption distribution is

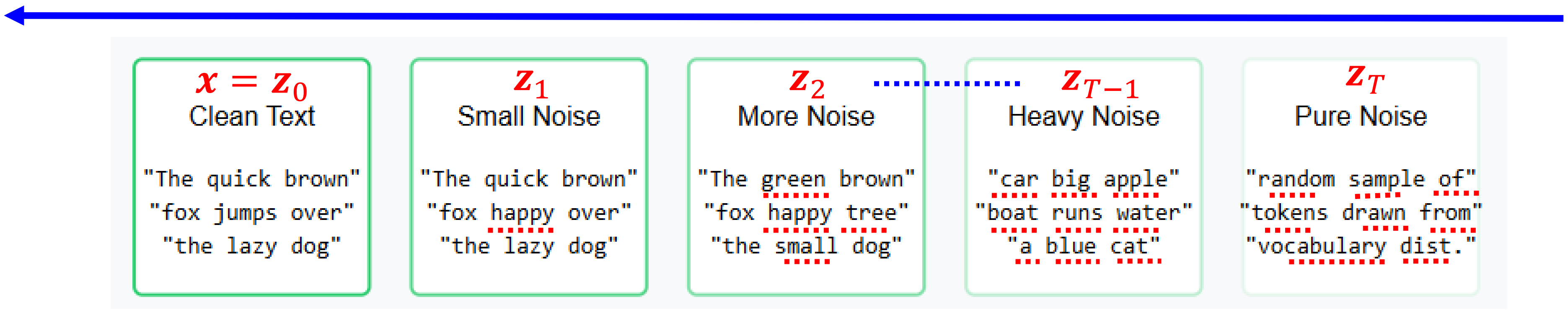
$$q(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = (1-\beta_t)\mathbb{I}[\boldsymbol{z}_t = \boldsymbol{z}_{t-1}] + \beta_t/V$$

- Basically, to get sequence $\boldsymbol{z}_t$ from $\boldsymbol{z}_{t-1}$, it does the following for each token in $\boldsymbol{z}_{t-1}$
  - With probability β_t, replace it by a random token from the vocabulary
  - With probability $1-\beta_t$, keep it unchanged
- Note: Some diffusion LLMs replace tokens by not a random but a "mask" token

Reverse Process

- Takes noisy text and produces less noisy text (basically opposite of forward process)

$$p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) = \prod_{i=1}^{N} \mathrm{Cat}(z_{t-1}^i|p_\theta(z_{t-1}^i|z_t^i))$$



- The training objective is similar to the one used in continuous data LLM
  - Basically we want to match $p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t)$ and $q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{x})$

$$\mathcal{L} = \mathbb{E}_{t,\boldsymbol{x},\boldsymbol{z}_t}[||p_\theta(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t) - q(\boldsymbol{z}_{t-1}|\boldsymbol{z}_t, \boldsymbol{x})||^2]$$

Diffusion LLMs: Some Pros and Cons

- Some pros
  - Parallel Decoding → Faster inference potential via non-sequential generation
  - Better Output Diversity → Naturally handles multi-modal distributions
  - Improved Controllability → Supports classifier-free guidance and conditioning
  - Resilience to Exposure Bias → Trained via denoising, not next-token prediction
  - Flexible Objectives → Enables structured generation, editing, and planning
- Some cons
  - Slower Training → Iterative denoising steps can increase training cost
  - Complex Architecture → Needs noise schedule, denoising network, sampling strategy
  - High Inference Cost (currently) → Requires multiple denoising steps at test time
  - Less Mature → Fewer benchmarks and toolkits compared to autoregressive LLMs
  - Tokenization Challenges → Needs careful handling of discrete text representations