Latent Diffusion (Stable-Diffusion) and Diffusion-LM

Zhiping (Patricia) Xiao

UCLA SCAI Lab Course Reading Group Fall 2022

Outline

Introduction

High-Resolution Image Synthesis with Latent Diffusion Models (CVPR'22)

- Operating on **latent space** of pre-trained auto-encoders, instead of on **pixel space**.
- Code: `https://github.com/CompVis/latent-diffusion` and `https://github.com/CompVis/stable-diffusion`

Diffusion-LM Improves Controllable Text Generation (NeurIPS'22)

- Build a Language Model eligible for fine-grained controllable text generation, by applying diffusion model on **continuous** latent space.
- Code: `https://github.com/XiangLi1999/Diffusion-LM`

Quick Recap: Generative Models

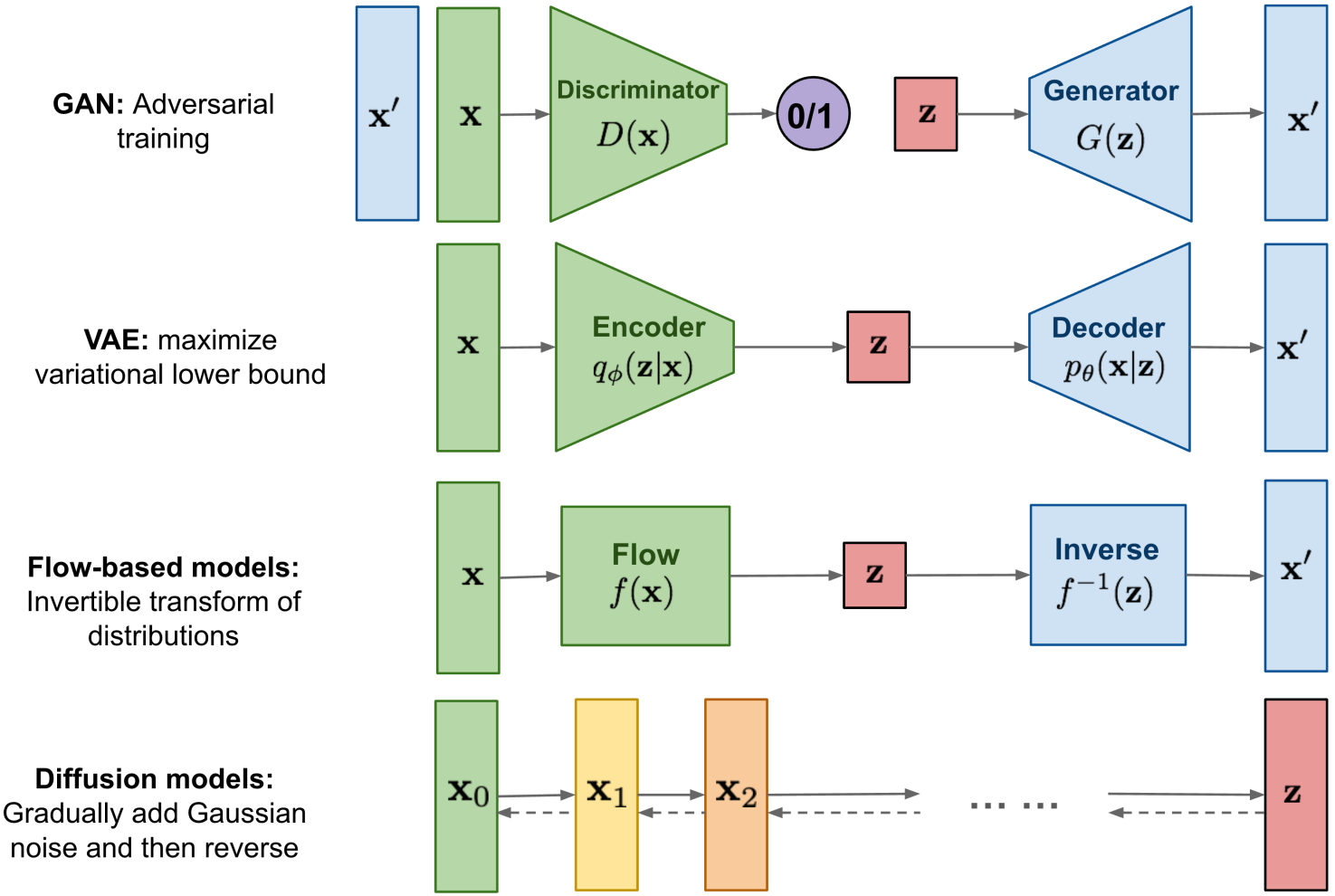

Figure: Overview of different types of generative models.[1]

[1] lilianweng.github.io 2021 post on diffusion models

Table: Advantage & Disadvantage of the Generative Models

Model	**likelihood-based?**	**Advantage**	**Disadvantage**
GAN	NO	efficient sampling; perceptual quality	struggle to optimize; not capture full data distribution
VAE/ Flow-based	YES	well-done density estimation; well-behaved optimize	low sample quality (compare to GANs)
DMs	YES	well-done density estimation; good sample quality	calculate expensive gradients

Quick Recap: Generative Models for LMs

Given a sequence of discrete words $\mathbf{w} = [w_1, w_2, \dots w_n]$, and a trained language model $p_{\text{lm}}(\mathbf{w})$ denoting the probability distribution over sequences of words.

Controllable Text Generation is to compute $p(\mathbf{w}|\mathbf{c})$, aiming at generate $\mathbf{w}$ that satisfies the control target $\mathbf{c}$, where $\mathbf{c}$ is a control variable (e.g. a target syntax tree, a desired sentiment label).

Plug-and-Play Controllable Generation: aims to keep the LM frozen and steer its output using potential functions (e.g., classifiers). In this setting:

- $p_{\text{lm}}(\mathbf{w})$ pre-trained and frozen, encouraging fluent;
- For each task, train $p(\mathbf{c}|\mathbf{w})$ on small amount of data, encourage $\mathbf{w}$ to fulfill the control.
- Posterior $p(\mathbf{w}|\mathbf{c})$ approximated from Bayes rule:

$$p(\mathbf{w}|\mathbf{c}) \propto p(\mathbf{c}|\mathbf{w}) \cdot p_{\text{lm}}(\mathbf{w})$$

Quick Recap: Diffusion Models (DMs)

Belong to the class of likelihood-based probabilistic models.

Designed to learn a data distribution $p(x)$ by gradually denoising a normally-distributed variable, which corresponds to learning the **reverse process** of a fixed Markov Chain of length T.

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right],$$

where t is uniformly sampled from $\{1, 2, \ldots T\}$, and the models can be interpreted as an equally-weighted sequence of denoising autoencoders $\epsilon_\theta(x_t, t)$, trained to predict the denoised version of x_t, namely x. *The above version works well in* ***image*** *settings.*

Diffusion models are capable of modeling conditional distributions in principle. In the form of $p(x|y)$.

Implemented via using a a conditional denoising autoencoder $\epsilon_\theta(x_t, t, y)$.

Typically, when x is an image, y could be text, semantic maps, or other image-to-image translation tasks, etc.

Quick Recap: Diffusion Models (DMs)

Recall that, the case of image:

$$L_{DM} = \mathbb{E}_{x,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(x_t, t)\|_2^2\right].$$

The case of text: to maximize $\mathbb{E}_{\mathbf{x}_0\sim p_{data}}\Big[\log p_\theta(\mathbf{x}_0)\Big]$

View as modeling latent variables of the data $\mathbf{x}_0 \in \mathbb{R}^d$ as a Markov chain $\mathbf{x}_T, \ldots, \mathbf{x}_0 \in \mathbb{R}^d$ where $\mathbf{x}_T$ is a Gaussian.
The initial state $p_\theta(\mathbf{x}_T) \approx \mathcal{N}(0, \mathbf{I})$, and

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)),$$

where μ_θ and Σ_θ may be computed by a U-Net or a Tranformer.

Quick Recap: Diffusion Models (DMs)

To train $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t))$:

- **Forward Process** q: incrementally adds Gaussian noise to data $\mathbf{x}_0$, until at diffusion step T, samples $\mathbf{x}_T$ are approximately Gaussian.

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \,,$$

  where β_t is the amount of noise added at step t. q is pre-defined and contains no trainable parameter, rather, we train a model to reverse the process.

- **Reverse Process** p_θ: reconstruct the data (i.e. $\mathbf{x}_T \to \dots \mathbf{x}_0$).
- The **diffusion model** is trained to maximize the marginal likelihood of the data

$$\mathbb{E}_{\mathbf{x}_0 \sim p_{data}}\Big[\log p_\theta(\mathbf{x}_0)\Big]$$

Quick Recap: Diffusion Models (DMs)

The simplified version of the canonical objective of maximizing

$$\mathbb{E}_{\mathbf{x}_0 \sim p_{data}}[\log p_\theta(\mathbf{x}_0)]$$

is the variational lower bound of $\log p_\theta(\mathbf{x}_0)$, [2] simplified as (NO LONGER a valid lower bound but mathematically proved and empirically more stable):

$$\mathcal{L}_{\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\|\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right],$$

where $\hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)$ is the mean of posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$ which is a closed form Gaussian, and $\mu_\theta(\mathbf{x}_t, t)$ is the predicted mean of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$, computed by a neural network.

Proof in Appendix E of **Diffusion-LM**. In brief: *it uses the closed form solution of the KL divergence of Gaussian*

[2] the canonical objective comes from ICML'15

Latent Diffusion

Latent Diffusion Models (LDMs): Motivation

Learning an LMs, as with any *likelihood-based model*, can be roughly divided into two stages, called **Two-Stage Image Synthesis**:

- **Perceptual Compression** stage: removes high-frequency details, but still learns little semantic variation;
- **Semantic Compression** stage: the generative model learns the semantic and conceptual composition of the data.

Observation: Most bits of a digital image correspond to imperceptible details, but we still need to train and evaluate on all pixels.

Latent Diffusion Models (LDMs)

Challenge: The most powerful DMs are often computationally demanding.

- Costly Training: takes hundreds of GPU days to train, prone to spend excessive amounts of capacity on modeling imperceptible details; [3]
- Costly Evaluation: cost a lot of time and memory, must run the same architecture sequentially for many of steps.
- e.g. Diffusion Models Beat GANs on Image Synthesis (NeurIPS'21) takes 150 - 1000 *V100 days* to train, 25 - 1000 steps to evaluate.

[3] This is because of the Mode-Covering behavior (reference)

Latent Diffusion Models (LDMs)

Challenge: The most powerful DMs are often computationally demanding.

- Costly Training: takes hundreds of GPU days to train, prone to spend excessive amounts of capacity on modeling imperceptible details; [3]
- Costly Evaluation: cost a lot of time and memory, must run the same architecture sequentially for many of steps.
- e.g. Diffusion Models Beat GANs on Image Synthesis (NeurIPS'21) takes 150 - 1000 *V100 days* to train, 25 - 1000 steps to evaluate.

Highlighted Novelty:

- **Latent**: Operating on latent space of powerful pre-trained auto-encoders, instead of directly on pixel space *(compared with standard Diffusion Models / DMs)*.

Benefit: **Less Costly**.

[3] This is because of the Mode-Covering behavior (reference)

Quick Recap: Mode-Covering Behavior

Minimize difference between $Q(x)$ and $P(x)$ at data distribution $P(x) > 0$:

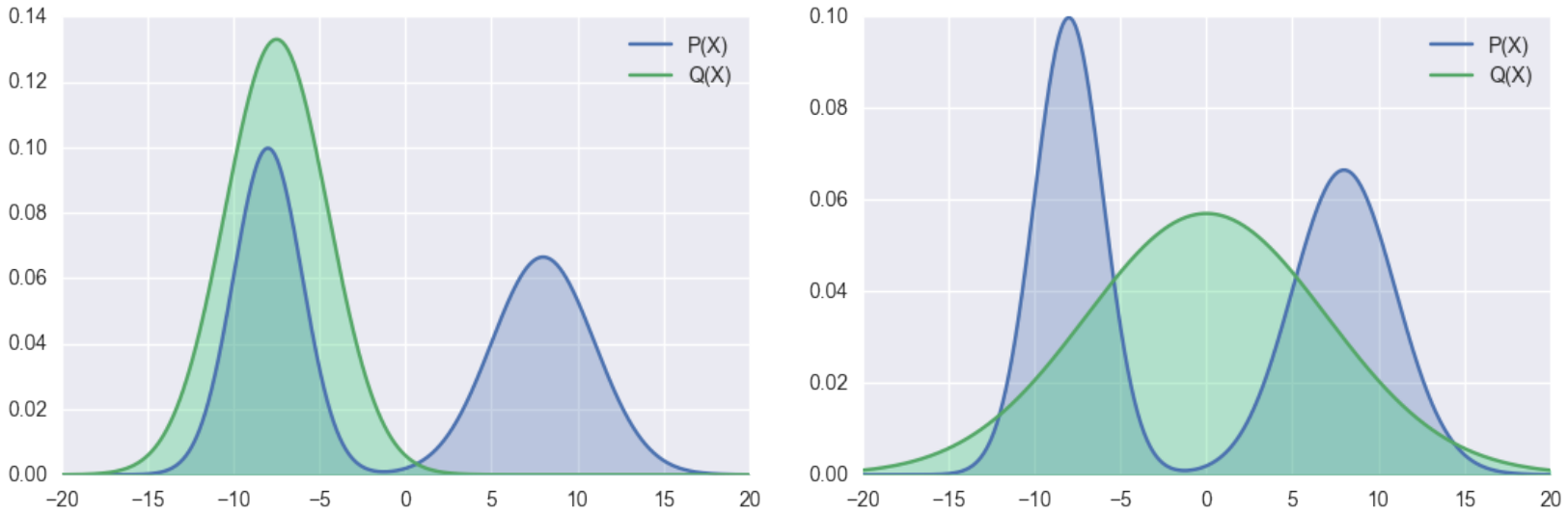

Figure: Bad v.s. Good mode-covering. On the left (bad) example, the right hand side **mode** is not covered by $Q(x)$, but it is the case that $P(x) > 0$. (from agustinus.kristia.de)

This behavior makes the DMs costly.

Latent Diffusion Models (LDMs): Architecture

Two-Stage Image Synthesis:

- Perceptual Image Compression: CLIP
  - Encoder $\mathcal{E}$, Decoder $\mathcal{D}$
  - $z = \mathcal{E}(x)$ where the RGB image $x \in \mathbb{R}^{H \times W \times 3}$ turns into latent representation $z \in \mathbb{R}^{h \times w \times c}$, while $\tilde{x} = \mathcal{D}(z)$ tries to reconstruct x.
  - Some regularization terms applied (e.g. KL-penalty towards a standard normal) to avoid arbitrarily high-variance latent spaces.
- Semantic Image Compression: **Latent Diffusion Models**

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t)\|_2^2\right],$$

where the neural backbone ϵ_θ of LDM is of realized as a time-conditional UNet.

Latent Diffusion Models (LDMs): Architecture

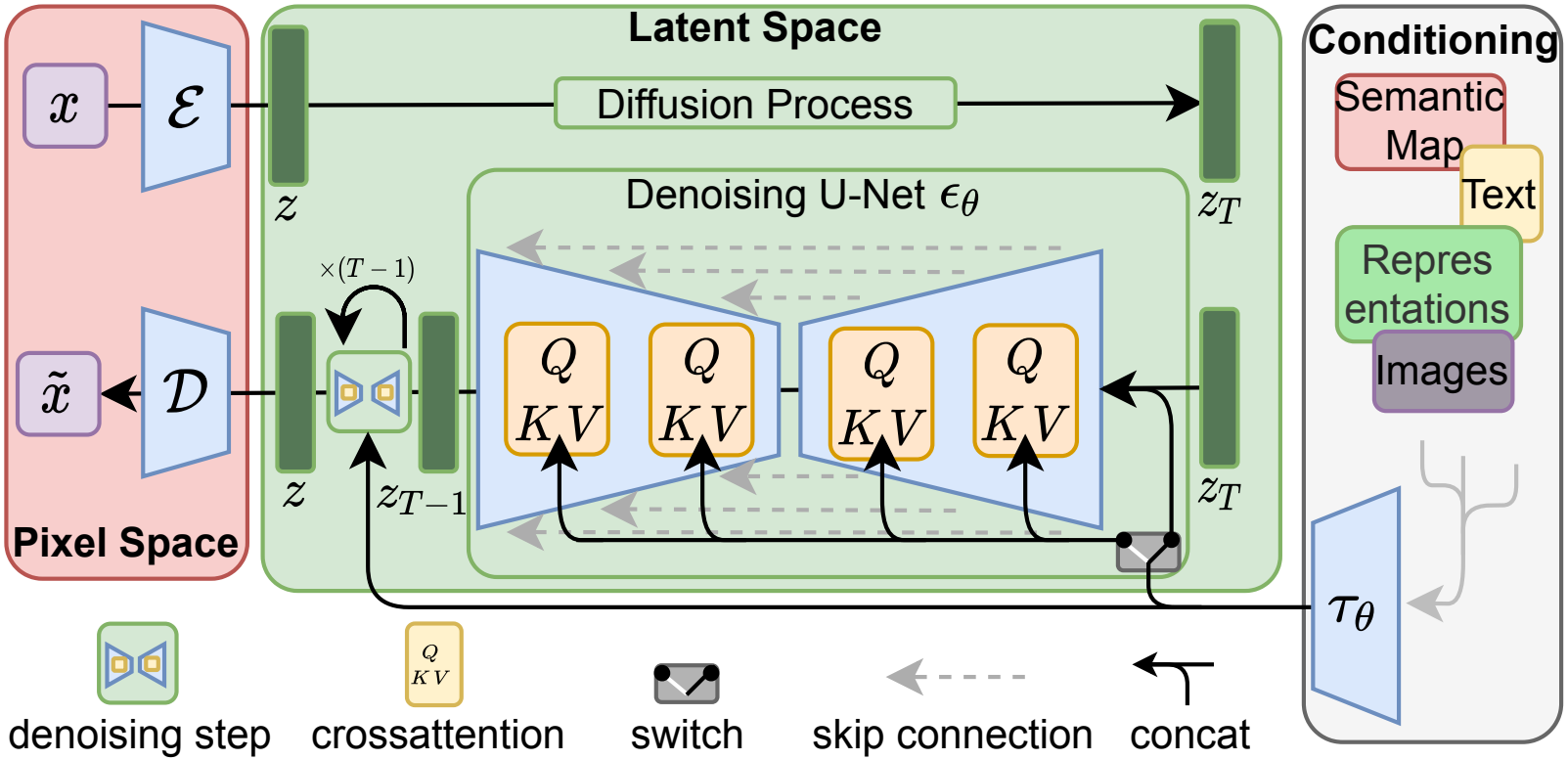

Figure: LDMs are either conditioned via concatenation or by a more general cross-attention mechanism.

CLIP (*)

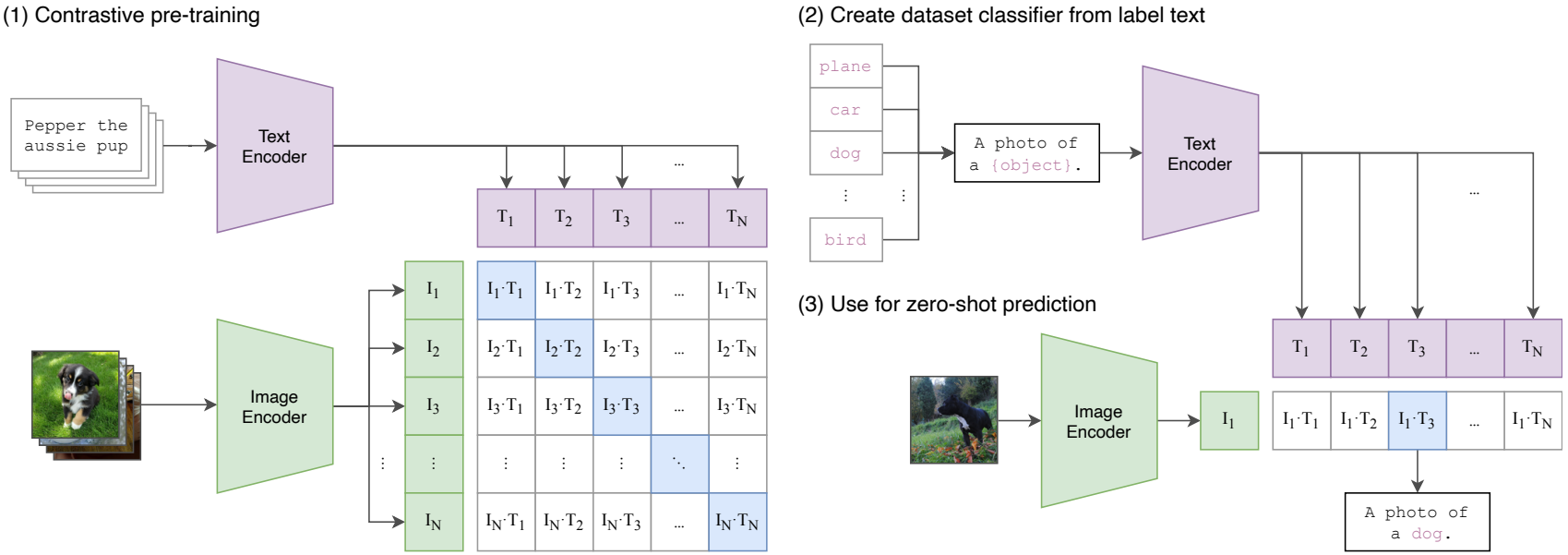

Figure: Standard image models jointly train an image encoder and a linear classifier, whereas CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At inference stage, text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

U-Net (*)

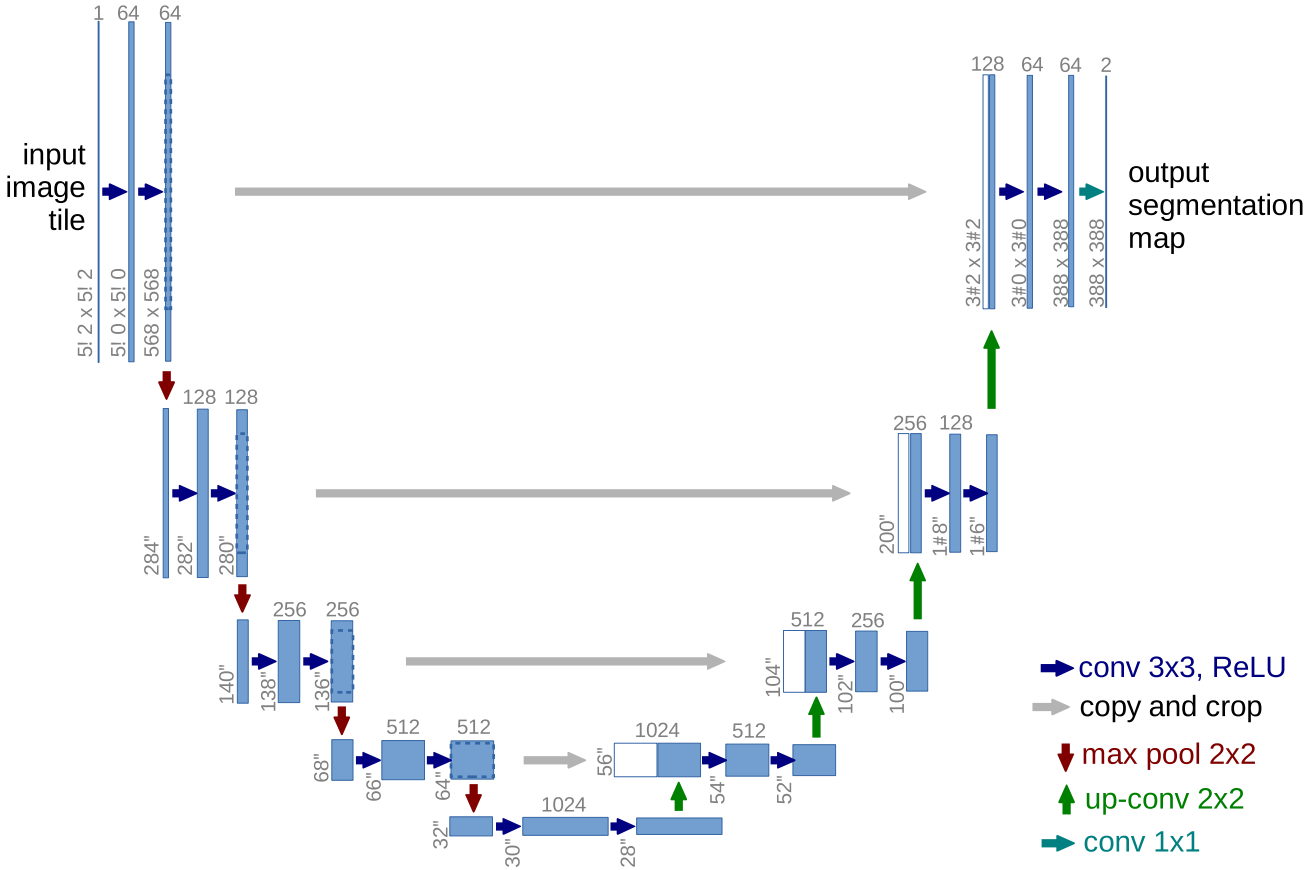

Figure: Each blue is a multi-channel feature map; White boxes copied feature maps. U-Net relies on data augmentation to use the limited available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization.

LDMs: Conditioning Mechanisms

As diffusion model, LDMs should be capable of modeling conditional distribution $p(z|y)$, by using a a conditional denoising autoencoder $\epsilon_\theta(z_t, t, y)$.

In *image synthesis*, combining DMs with y beyond *class-labels* or *blurred variants of the input image*, is under-explored.

LDMs propose to augment the UNet backbone implementing ϵ_θ with the cross-attention mechanism. There, Q, K, V are projection (transformation e.g. $W_Q^{(i)}$ learnable) of $\phi_i(z_t)$, $\tau_\theta(y)$, $\tau_\theta(y)$ respectively. $\phi_i(z_t)$ is the flattened intermediate representation of the UNet implementing ϵ_θ.

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right],$$

where τ_θ is a domain specific encoder used to project y, e.g. τ_θ can be transformers when y are text prompts.

LDMs: Highlighted Result

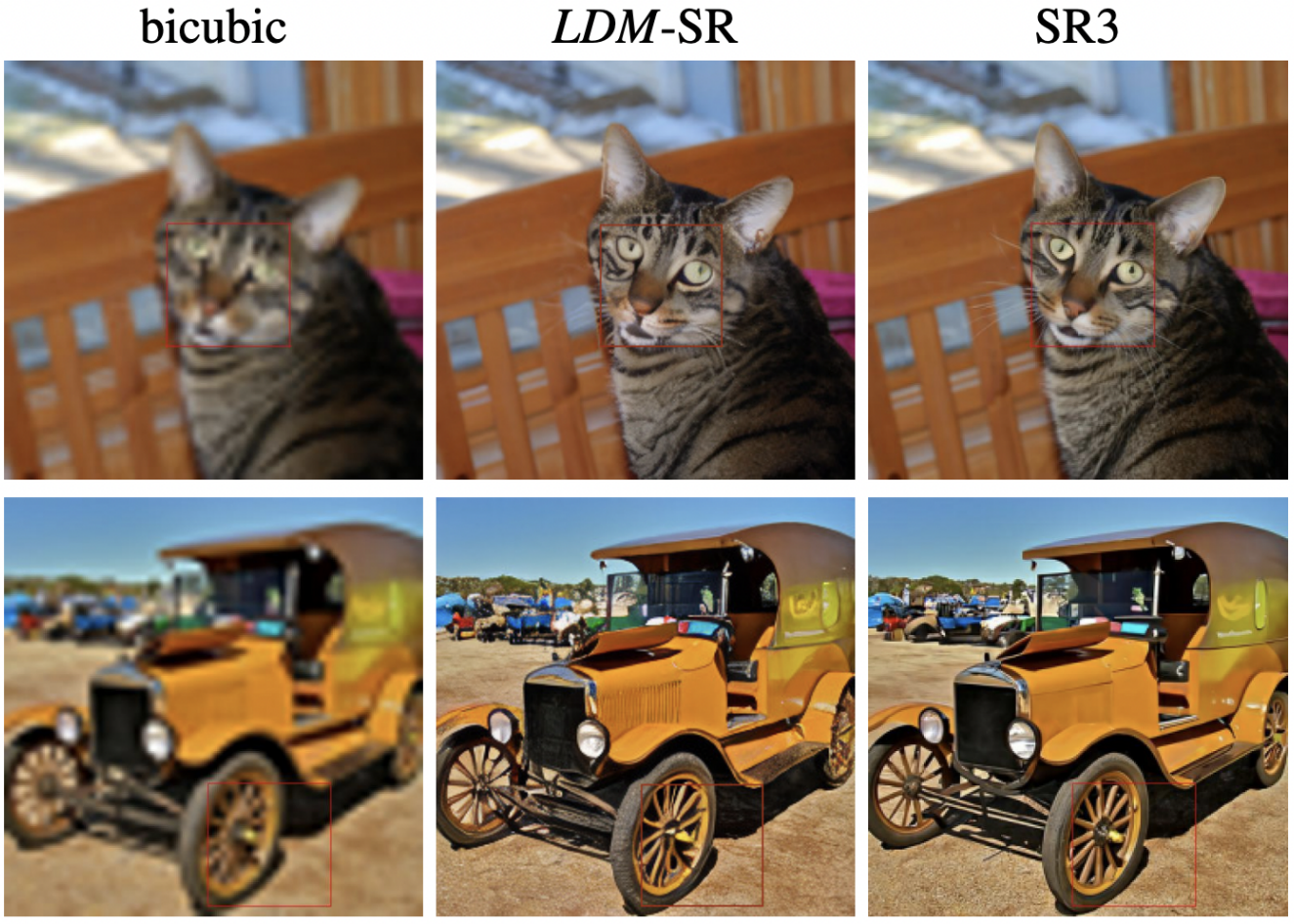

Figure: ImageNet 64 → 256 super-resolution on ImageNet-Val. LDM-SR has advantages at rendering realistic textures but SR3 can synthesize more coherent fine structures.

The first popular AI-painter that gets attention this year (2022), is DALLE (`https://openai.com/dall-e-2/`). Latent diffusion / stable diffusion comes after, but it has a great advantage being **open-sourced**. Therefore, everyone adapted their model and play with images (e.g. `https://novelai.net/`), finally attract audiences from all over the world.

- Note: There has been serious ethics concern on the copyright of the online images.

Some Temporary Resource on Novel AI (*)

`https://pan.baidu.com/s/1_DeroYt9AnlsNpJ0kqcLCQ`
Code: ctkn

- Link will be expired in **30 days** (expire by Jan 6 2023)
- Note Writtrn in Chinese
- Shared by Chenguang Zhang (`https://scholar.google.com/citations?user=lrxyS5wAAAAJ&hl=en`)
- If the link expires but you need these materials, you may try to contact him for help.

Diffusion-LM

Focusing on a language models (LMs) offering **controllable generation** for text without re-training.

Recent works succeed on controlling simple attributes e.g., sentiment, while little progress is made on complex, fine-grained controls (e.g., syntactic structure).

Solution: a new **non**-autoregressive language model based on **continuous diffusions**.

- **continuous** data domains: images, audio, etc. (*enables efficient gradient-based controllable generation*)
- previous text diffusion models: on discrete state spaces, defines a corruption process on **discrete** data (e.g., *each token has some probability to be corrupted to an absorbing or random token*).

Autoregressive LMs: $p_{\text{lm}}(\mathbf{w})$

$$p_{\text{lm}}(\mathbf{w}) = p_{\text{lm}}(w_1) \prod_{i=2}^{n} p_{\text{lm}}(x_i | x_{<i}) \,,$$

where the next-token prediction $p_{\text{lm}}(x_i | x_{<i})$ is often parametrized by Transformer architecture.

Claim: Most large pre-trained LMs are left-to-right autoregressive. (e.g. GPT-3, PaLM)

- Fixed generation order (i.e., left to right) limits the flexibility of models.

Non-autoregressive LMs:

Claim: Most of the existing models in this category are task-specific, such as machine translation and speech-to-text (e.g. CTC and Imputer, NAT))

- It has been shown that these models fail for language modeling in CoMMA paper.

Claim: Diffusion-LM can condition on arbitrary classifiers, which could utilize complex, global properties of the sentence.

Diffusion LM: Embedding & Rounding

Embedding:

- $\text{EMB}(w_i) \in \mathbb{R}^d$ embed discrete word w_i into vector space.
- $\text{EMB}(\mathbf{w}) = [\text{EMB}(w_1), \dots \text{EMB}(w_n)] \in \mathbb{R}^{nd}$ denotes the embedding of a length-n sequence.
- Claim: Fixed pre-trained embedding or random Gaussian embeddings are worse than the embedding trained via an end-to-end framework.

$$q_\phi(\mathbf{x}_0|\mathbf{w}) = \mathcal{N}(\text{EMB}(\mathbf{w}), \sigma_0 I)$$

Rounding: achieved by choosing the most probable word according to argmax

$$p_\theta(\mathbf{w}|\mathbf{x}_0) = \prod_{i=1}^{n} p_\theta(w_i|x_i)\,,$$

where $p_\theta(w_i|x_i)$ is a softmax distribution.

Diffusion LM: Training Objective

Recall that the simplified version of the canonical objective of maximizing $\mathbb{E}_{\mathbf{x}_0 \sim p_{data}}[\log p_\theta(\mathbf{x}_0)]$ is:

$$\mathcal{L}_{\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\|\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right],$$

where $\hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)$ is the mean of $q(\mathbf{x}_{t-1}|\mathbf{x}_0, \mathbf{x}_t)$, and $\mu_\theta(\mathbf{x}_t, t)$ is the predicted mean of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$. The end2end version objective:

$$\begin{aligned}\mathcal{L}^{\text{e2e}}_{\text{simple}}(\mathbf{w}) =& \mathbb{E}_{q_\phi(\mathbf{x}_{0:T}|\mathbf{w})} \Big[\mathcal{L}_{\text{simple}}(\mathbf{x}_0) + \|\text{EMB}(\mathbf{w}) - \mu_\theta(\mathbf{x}_1, 1)\|^2 \\ & - \log p_\theta(\mathbf{x}_0|\mathbf{w}) \Big],\end{aligned}$$

Diffusion LM: Architecture

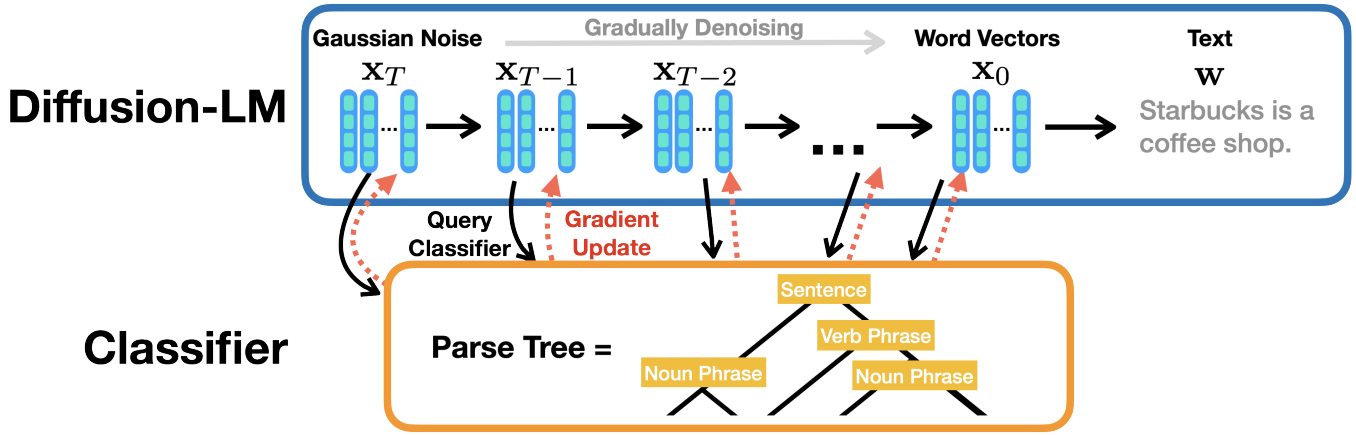

Figure: Iteratively denoises a sequence of Gaussian vectors into word vectors, yielding a intermediate latent variables of decreasing noise level $x_T \dots x_0$. For controllable generation: iteratively perform gradient updates on these continuous latents to optimize for fluency (parametrized by **Diffusion-LM**) and satisfy control requirements (parametrized by a **classifier**).

Diffusion LM: Architecture

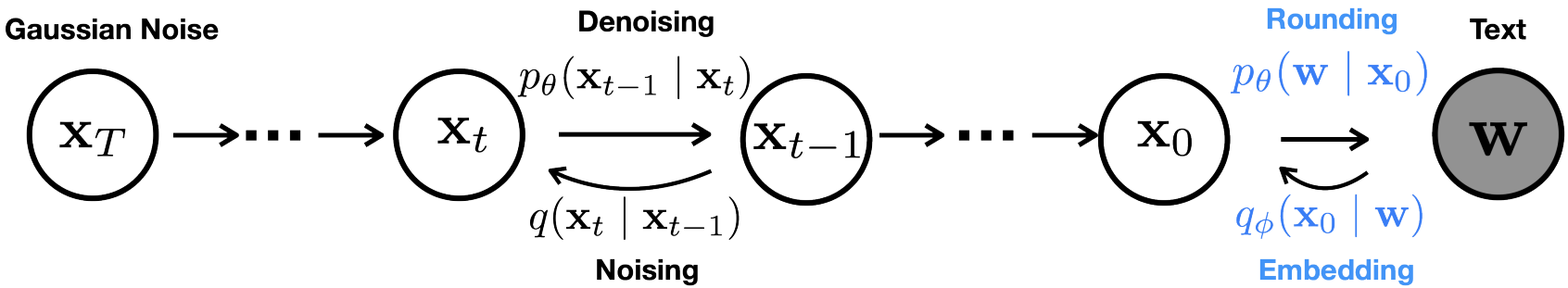

Figure: The forward and reverse diffusion processes. In addition to the original diffusion models, Diffusion-LM add a Markov transition between $\mathbf{x}_0$ and $\mathbf{w}$, defining embedding ($\mathbf{w} \to \mathbf{x}_0$) and rounding ($\mathbf{x}_0 \to \mathbf{w}$).

Recall that rounding is achieved by choosing argmax of

$$p_\theta(\mathbf{w}|\mathbf{x}_0) = \prod_{i=1}^{n} p_\theta(w_i|x_i)\,,$$

where $p_\theta(w_i|x_i)$ is a softmax distribution. But the problem is that, empirically, the model fails to generate $\mathbf{x}_0$ that commits to a single word.

- Explanation: not emphasizing single commit on $\mathbf{x}_0$ enough.

$$\mathcal{L}_{\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\|\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right]$$

- Quick Fix: $\mathbf{x}_0$-parameterized model

$$\mathcal{L}^{\text{e2e}}_{\mathbf{x}_0-\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t} \|f_\theta(\mathbf{x}_t, t) - \mathbf{x}_0\|^2\,.$$

$\mathbf{x}_0$-parameterized model:

$$\mathcal{L}^{\text{e2e}}_{\mathbf{x}_0-\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^{T} \mathbb{E}_{\mathbf{x}_t} \|f_\theta(\mathbf{x}_t, t) - \mathbf{x}_0\|^2 ,$$

where our re-parameterized model $f_\theta(\mathbf{x}_t, t)$ learns $\mathbf{x}_0$ directly.

In the decoding phase, same intuition could be used in **clamping trick**:

- Maps the predicted $f_\theta(\mathbf{x}_t, t)$ to its **nearest** word embedding sequence at every step.
- Forces the predicted vector to commit to a word for intermediate diffusion steps, making predictions more precise and reducing rounding errors.

Controlling $\mathbf{x}_{0:T}$ over $\mathbf{c}$ is equivalent with decoding from the joint inference problem posterior:

$$p(\mathbf{x}_{0:T}|\mathbf{c}) = \prod_{t=1}^{T} p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) \,,$$

which can be decomposed to a sequence of control problems at each diffusion step:

$$p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) \propto p(\mathbf{x}_{t-1}|\mathbf{x}_t) \cdot p(\mathbf{c}|\mathbf{x}_{t-1}, \mathbf{x}_t)$$

And according to a Yang Song et al's paper from ICLR'21, there are *conditional independence assumptions* that we can use to simplify:

$$p(\mathbf{c}|\mathbf{x}_{t-1}, \mathbf{x}_t) = p(\mathbf{c}|\mathbf{x}_{t-1})$$

Therefore, for the t-th step, we run gradient update on $\mathbf{x}_{t-1}$:

$$\nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1}|\mathbf{x}_t) + \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{c}|\mathbf{x}_{t-1}) \,,$$

where the two differentiable terms are parameterized by:

- $\log p(\mathbf{x}_{t-1}|\mathbf{x}_t)$: Diffusion LM;
- $\log p(\mathbf{c}|\mathbf{x}_{t-1})$: an arbitrary neural network classifier;

respectively.

Diffusion LM: Highlighted Result

	Semantic Content		Parts-of-speech		Syntax Tree		Syntax Spans		Length	
	ctrl ↑	lm ↓	ctrl ↑	lm ↓	ctrl ↑	lm ↓	ctrl ↑	lm ↓	ctrl ↑	lm ↓
PPLM	9.9	5.32	-	-	-	-	-	-	-	-
FUDGE	69.9	2.83	27.0	7.96	17.9	**3.39**	54.2	4.03	46.9	3.11
Diffusion-LM	**81.2**	**2.55**	**90.0**	**5.16**	**86.0**	3.71	**93.8**	**2.53**	**99.9**	**2.16**
FT-sample	72.5	2.87	89.5	4.72	64.8	5.72	26.3	2.88	98.1	3.84
FT-search	89.9	1.78	93.0	3.31	76.4	3.24	54.4	2.19	100.0	1.83

Table 2: Diffusion-LM achieves high success rate (ctrl ↑) and good fluency (lm ↓) across all 5 control tasks, outperforming the PPLM and FUDGE baselines. Our method even outperforms the fine-tuning oracle (FT) on controlling syntactic parse trees and spans.

Diffusion LM: Highlighted Result

Syntactic Parse	(S (S (NP *) (VP * (NP (NP * *) (VP * **(NP (ADJP * *) *)**))))) * (S (NP * * *) (VP * (ADJP (ADJP *)))))
FUDGE	Zizzi is a cheap restaurant . **[incomplete]**
Diffusion-LM	Zizzi is a pub providing **family friendly Indian food** Its customer rating is low
FT	Cocum is a Pub serving **moderately priced meals** and the customer rating is high
Syntactic Parse	(S (S (VP * (PP * (NP * *)))) * **(NP * * *)** (VP * (NP (NP * *) (SBAR (WHNP *) (S (VP * (NP * *)))))) *)
FUDGE	In the city near The Portland Arms is a coffee and fast food place named The Cricketers which is not family - friendly with a customer rating of 5 out of 5 .
Diffusion-LM	Located on the riverside , **The Rice Boat** is a restaurant that serves Indian food .
FT	Located near The Sorrento, **The Mill** is a pub that serves Indian cuisine.

Table 3: Qualitative examples from the Syntax Tree control. The syntactic parse tree is linearized by nested brackets representing the constituents, and we use the standard PTB syntactic categories. Tokens within each span are represented as * . We color failing spans red and **bold** the spans of interest that we discuss in §7.1.