Structured Denoising Diffusion Models in Discrete State-Spaces

Written by Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, Rianne van den Berg in Nov 2021

Presentation by: Konrad Breipohl

Agenda

- What is a Diffusion Model
- Introduction to the Paper
- Core Contributions of the Paper
- Technical implementation
- Results of the paper
- Conclusion
- Q&A

What is a Diffusion Model

Basics

- A diffusion model is often used for image generation from prompts
- Two processes at work: a noising algorithm (forward process) and a denoising algorithm (reverse process)

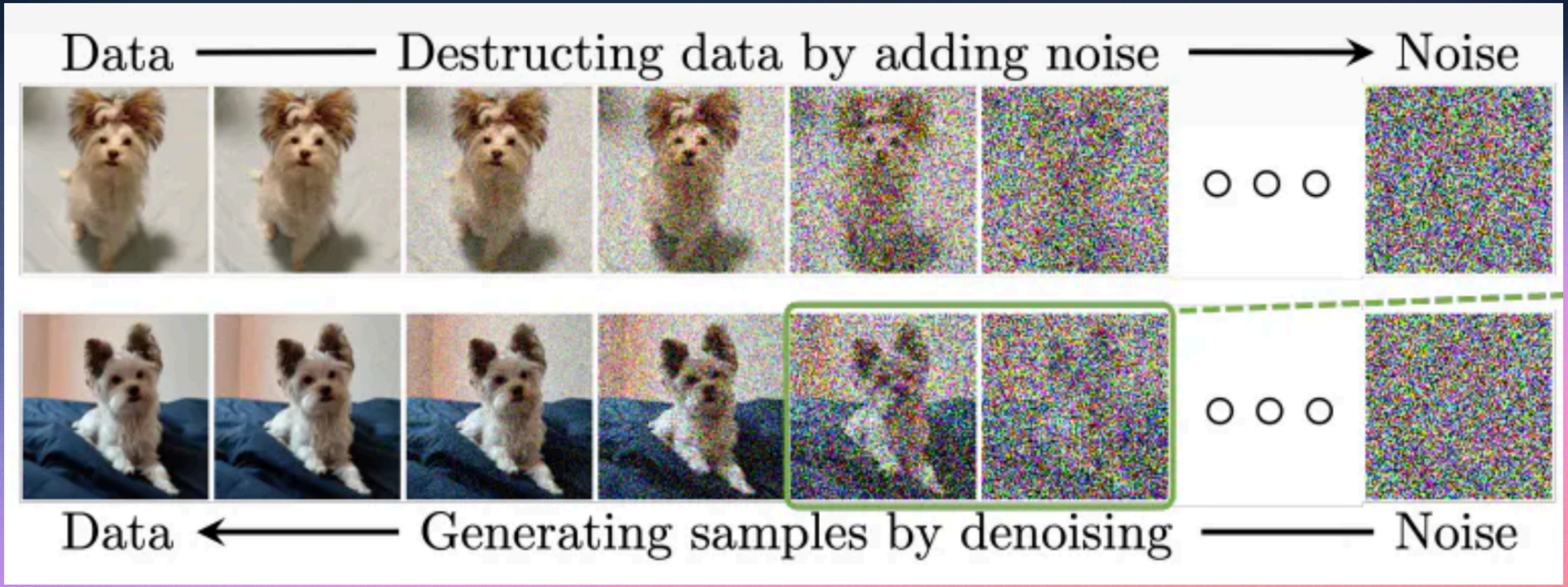


Image source: https://doi.org/10.48550/arXiv.2209.00796

What is a Diffusion Model

Basics

- The noising and denoising is done iteratively
- Both algorithms are run multiple times over
- Only small amount of noise added each step
- Denoising only used to reverse one step of noise
- The noising process is pre-defined, the denoising process is what we train and care about

What is a Diffusion Model

The Noising Process

- The noising process typically uses a Gaussian Distribution
- Only small amount of Gaussian noise is added each step
- Progressively degrades image into pure noise
- Models can use many steps; hundreds to thousands of small noising steps

What is a Diffusion Model

The Denoising Process

- Denoising is done using a Neural Network (NN)
- Iterative; therefore trained on single steps of noising process
  - Tries to reverse the noise created,
  - Denoising performance evaluated using a score function
- Important: Reverse process does not magically create an image out of thin air; it takes a noisy input and removes a small amount of noise and is being applied many times over

What is a Diffusion Model

- Great for image generation
  - Allows us to start with new unseen random noise and then create an image
- Of course can also be used for many other types of data generation

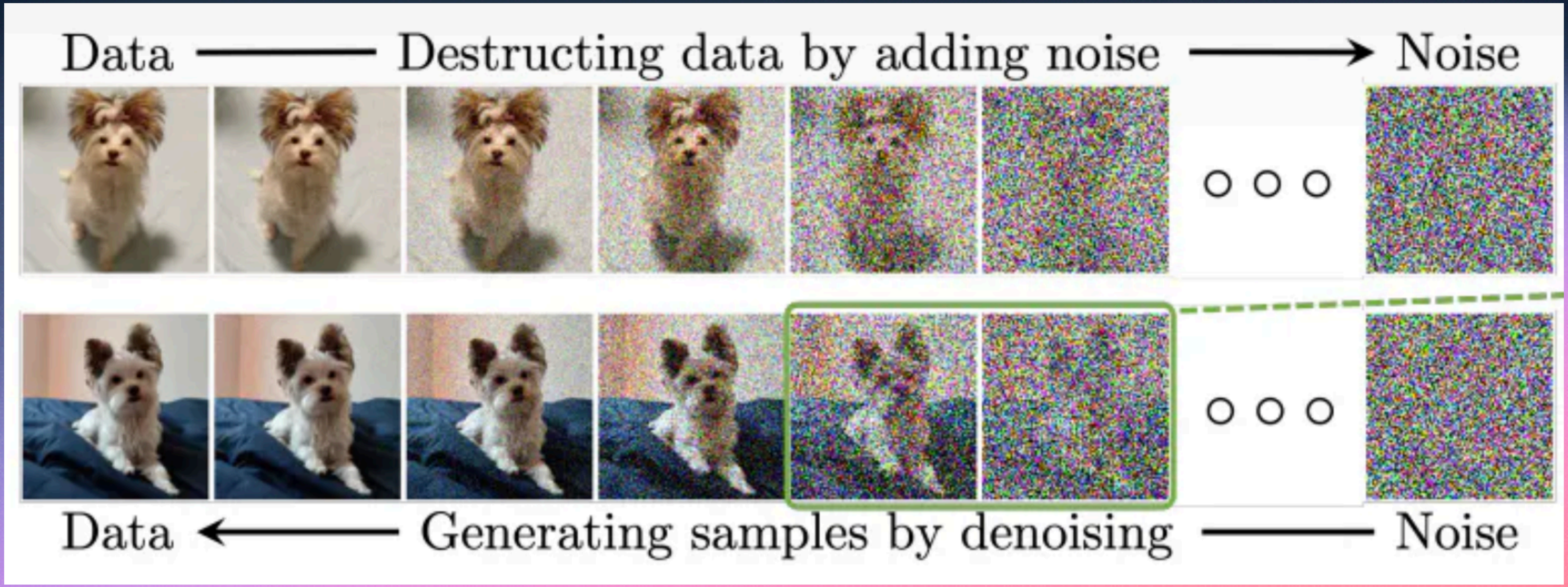


Image source: https://doi.org/10.48550/arXiv.2209.00796

Introduction to the Paper

General idea

- „Structured Denoising Diffusion Models in Discrete State-Spaces“
- Basic Discrete Diffusion Models have already been developed
  - Traditional models only allow continuous data due to Gaussian Noise which can take on any real value, like pixel intensities in images

Introduction to the Paper

General idea

- Instead of Gaussian noise, discrete-appropriate noise is added
  - Could be flipping bits (classification), changing tokens (text) or changing between categorical states (categorization)
- **Paper expands on discrete diffusion models by modelling <u>structured</u> discrete data in the noising process**

Core Contributions

- Reminder: standard approach makes use of <u>uniform</u> or <u>random</u> noise
- Paper introduces Markov transition matrices that mimic the structure of the real data
    - These matrices govern how the data is corrupted/ noised
- Advantage: reverse process is more effectively trained as it is reversing a more plausible corruption process

Core Contributions

Examples of structured noise

- Text: random noise would change a word for a random word. Structured noise would change it with a synonym of the original word (or a similar semantic relationship)
- Images: Pixels have spatial relationships; a pixel surrounded by sky is highly likely to also depict the sky or pixels at edges of objects change in a way that „respects" the boundary rather than completely altering it
- Categorical data: In surveys, responses like „strongly agree", „agree", „neutral", „disagree", „strongly disagree" should not change too much (from strongly agree to strongly disagree). Instead use transition to adjacent categories

Core Contributions

Proposed Markov Transition Matrices

- **Uniform:**
  - Every state has an equal probability of transitioning to any other state.
- **Absorbing State:**
  - Elements change with probability into an absorbing state ([MASK] or gray pixel)
- **Discretized Gaussian:**
  - Transition probabilities favor nearby states, modeled after a Gaussian distribution
- **Token Embedding Distance:**
  - Transition probabilities between tokens are based on the distance in their embedding space, favoring transitions between semantically or syntactically similar tokens

Core Contributions

Noise schedule

- The paper explored multiple approaches
  - Gaussian: linear increase in variance before discretization (leads to non-linear amount of cumulative noise)
  - Uniform: probability of transition is based on cosine function
  - For general transition matrices such approaches may not be applicable
    - Such cases were explored with a linear interpolation of the mutual information of x_t and x_0 to 0

Technical Implementation

Loss function and Parameterizations

- The neural network is optimized using a hybridized loss function combining:
  - **Lower variational bound** (maximizes evidence lower bound for effective approximation of complex posterior distributions.)
  - **Expectation terms** (integrate model performance over initial data distributions and their evolution under noise)
  - **Negative Log-Likelihood**

$$L_\lambda = L_{\mathrm{vb}} + \lambda\, \mathbb{E}_{q(\boldsymbol{x}_0)} \mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)} [-\log \widetilde{p}_\theta(\boldsymbol{x}_0|\boldsymbol{x}_t)].$$

Source: https://doi.org/10.48550/arXiv.2107.03006

Technical Implementation

Loss function and Parameterizations

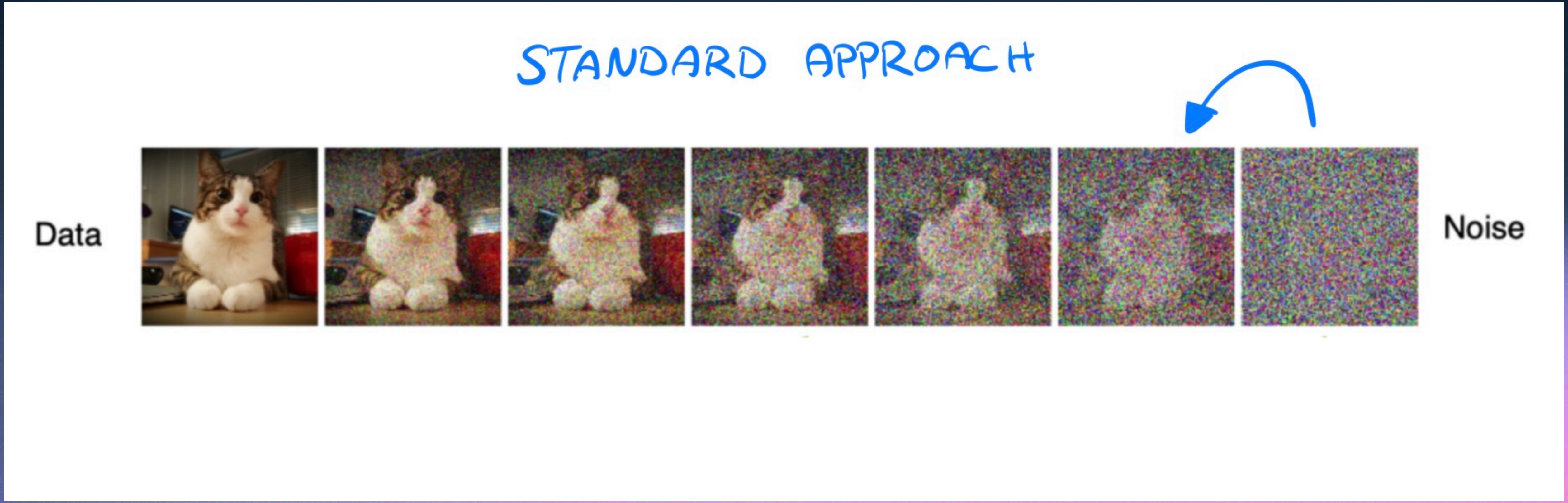
- Typically diffusion models will simply predict the next state x_{t-1} from the current state x_t
- Paper approach: instead of directly predicting x_{t-1} from x_t also utilize predictions about x_0 (the initial state)
    - Leads to more „goal-focussed“ training

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) \propto \sum_{\widetilde{\boldsymbol{x}}_0} q(\boldsymbol{x}_{t-1}, \boldsymbol{x}_t|\widetilde{\boldsymbol{x}}_0)\widetilde{p}_\theta(\widetilde{\boldsymbol{x}}_0|\boldsymbol{x}_t).$$

Source: https://doi.org/10.48550/arXiv.2107.03006

Technical Implementation

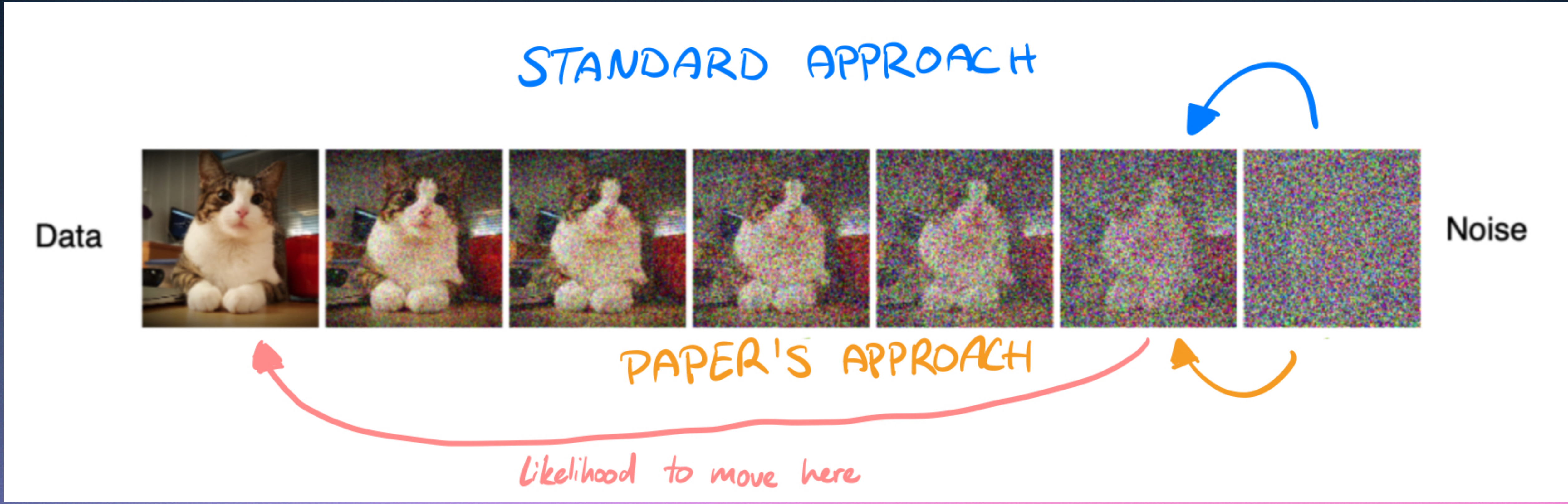
Loss function and Parameterizations



Original image source: https://research.nvidia.com/publication/2022-03_tackling-generative-learning-trilemma-denoising-diffusion-gans-0

Technical Implementation

Loss function and Parameterizations



Original image source: https://research.nvidia.com/publication/2022-03_tackling-generative-learning-trilemma-denoising-diffusion-gans-0

Results

Text Generation Results

Model	Model steps	NLL (bits/char) (↓)	Sample time (s) (↓)
Discrete Flow [49] (8 × 3 layers)	-	1.23	0.16
Argmax Coupling Flow [20]	-	1.80	0.40 ± 0.03
IAF / SCF [57]‡	-	1.88	0.04 ± 0.0004
Multinomial Diffusion (D3PM uniform) [20]	1000	≤ 1.72	26.6 ± 2.2
D3PM uniform [20] (ours)	1000	$\leq 1.61 \pm 0.02$	3.6 ± 0.4
D3PM NN (L_{vb}) (ours)	1000	$\leq 1.59 \pm 0.03$	3.1474 ± 0.0002
D3PM mask ($L_{\lambda=0.01}$) (ours)	1000	$\leq 1.45 \pm 0.02$	3.4 ± 0.3
D3PM uniform [20] (ours)	256	$\leq 1.68 \pm 0.01$	0.5801 ± 0.0001
D3PM NN (L_{vb}) (ours)	256	$\leq 1.64 \pm 0.02$	0.813 ± 0.002
D3PM absorbing ($L_{\lambda=0.01}$) (ours)	256	$\leq 1.47 \pm 0.03$	0.598 ± 0.002
Transformer decoder (ours)	256	1.23	0.3570 ± 0.0002
Transformer decoder [1]	256	1.18	-
Transformer XL [10]†	256	1.08	-
D3PM uniform [20] (ours)	20	$\leq 1.79 \pm 0.03$	0.0771 ± 0.0005
D3PM NN (L_{vb}) (ours)	20	$\leq 1.75 \pm 0.02$	0.1110 ± 0.0001
D3PM absorbing ($L_{\lambda=0.01}$) (ours)	20	$\leq 1.56 \pm 0.04$	0.0785 ± 0.0003

Source: https://doi.org/10.48550/arXiv.2107.03006

Results

Text Generation Results

Model	Model steps	NLL (bits/char) (↓)	Sample time (s) (↓)
Discrete Flow [49] (8×3 layers)	-	1.23	0.16
Argmax Coupling Flow [20]	-	1.80	0.40 ± 0.03
IAF / SCF [57]‡	-	1.88	0.04 ± 0.0004
Multinomial Diffusion (D3PM uniform) [20]	1000	≤ 1.72	26.6 ± 2.2
D3PM uniform [20] (ours)	1000	$\leq 1.61 \pm 0.02$	3.6 ± 0.4
D3PM NN (L_{vb}) (ours)	1000	$\leq 1.59 \pm 0.03$	3.1474 ± 0.0002
D3PM mask ($L_{\lambda=0.01}$) (ours)	1000	$\leq 1.45 \pm 0.02$	3.4 ± 0.3
D3PM uniform [20] (ours)	256	$\leq 1.68 \pm 0.01$	0.5801 ± 0.0001
D3PM NN (L_{vb}) (ours)	256	$\leq 1.64 \pm 0.02$	0.813 ± 0.002
D3PM absorbing ($L_{\lambda=0.01}$) (ours)	256	$\leq 1.47 \pm 0.03$	0.598 ± 0.002
Transformer decoder (ours)	256	1.23	0.3570 ± 0.0002
Transformer decoder [1]	256	1.18	-
Transformer XL [10]†	256	1.08	-
D3PM uniform [20] (ours)	20	$\leq 1.79 \pm 0.03$	0.0771 ± 0.0005
D3PM NN (L_{vb}) (ours)	20	$\leq 1.75 \pm 0.02$	0.1110 ± 0.0001
D3PM absorbing ($L_{\lambda=0.01}$) (ours)	20	$\leq 1.56 \pm 0.04$	0.0785 ± 0.0003

Source: https://doi.org/10.48550/arXiv.2107.03006

Results

Text Generation Results

- On the text8 dataset the absorbing state D3PM performed best in its class
- It was compared to transformer models and the idea introduced in earlier literature
- All transformers outperform the D3PM with absorbing state

Results

Image Generation Results

Model	IS (↑)	FID (↓)	NLL (↓)
Sparse Transformer [9]			2.80
NCSN [45]	8.87 ± 0.12	25.32	
NCSNv2 [46]	8.40 ± 0.07	10.87	
StyleGAN2 + ADA [22]	9.74 ± 0.05	3.26	
Diffusion (original), L_{vb} [43]			≤ 5.40
DDPM L_{vb} [19]	7.67 ± 0.13	13.51	≤ 3.70
DDPM L_{simple} [19]	9.46 ± 0.11	3.17	≤ 3.75
Improved DDPM L_{vb} [30]		11.47	≤ 2.94
Improved DDPM L_{simple} [30]		2.90	≤ 3.37
DDPM++ cont [47]		2.92	2.99
NCSN++ cont. [47]	9.89	2.20	
D3PM uniform L_{vb}	5.99 ± 0.14	51.27 ± 2.15	$\leq 5.08 \pm 0.02$
D3PM absorbing L_{vb}	6.26 ± 0.10	41.28 ± 0.65	$\leq 4.83 \pm 0.02$
D3PM absorbing $L_{\lambda=0.001}$	6.78 ± 0.08	30.97 ± 0.64	$\leq 4.40 \pm 0.02$
D3PM Gauss L_{vb}	7.75 ± 0.13	15.30 ± 0.55	$\leq 3.966 \pm 0.005$
D3PM Gauss $L_{\lambda=0.001}$	8.54 ± 0.12	8.34 ± 0.10	$\leq 3.975 \pm 0.006$
D3PM Gauss + logistic $L_{\lambda=0.001}$	8.56 ± 0.10	7.34 ± 0.19	$\leq 3.435 \pm 0.007$

Source: https://doi.org/10.48550/arXiv.2107.03006

Results

Image Generation Results

Model	IS (↑)	FID (↓)	NLL (↓)
Sparse Transformer [9]			2.80
NCSN [45]	8.87 ± 0.12	25.32	
NCSNv2 [46]	8.40 ± 0.07	10.87	
StyleGAN2 + ADA [22]	9.74 ± 0.05	3.26	
Diffusion (original), L_{vb} [43]			≤ 5.40
DDPM L_{vb} [19]	7.67 ± 0.13	13.51	≤ 3.70
DDPM L_{simple} [19]	9.46 ± 0.11	3.17	≤ 3.75
Improved DDPM L_{vb} [30]		11.47	≤ 2.94
Improved DDPM L_{simple} [30]		2.90	≤ 3.37
DDPM++ cont [47]		2.92	2.99
NCSN++ cont. [47]	9.89	2.20	
D3PM uniform L_{vb}	5.99 ± 0.14	51.27 ± 2.15	$\leq 5.08 \pm 0.02$
D3PM absorbing L_{vb}	6.26 ± 0.10	41.28 ± 0.65	$\leq 4.83 \pm 0.02$
D3PM absorbing $L_{\lambda=0.001}$	6.78 ± 0.08	30.97 ± 0.64	$\leq 4.40 \pm 0.02$
D3PM Gauss L_{vb}	7.75 ± 0.13	15.30 ± 0.55	$\leq 3.966 \pm 0.005$
D3PM Gauss $L_{\lambda=0.001}$	8.54 ± 0.12	8.34 ± 0.10	$\leq 3.975 \pm 0.006$
D3PM Gauss + logistic $L_{\lambda=0.001}$	8.56 ± 0.10	7.34 ± 0.19	$\leq 3.435 \pm 0.007$

Source: https://doi.org/10.48550/arXiv.2107.03006

Results

Image Generation Results

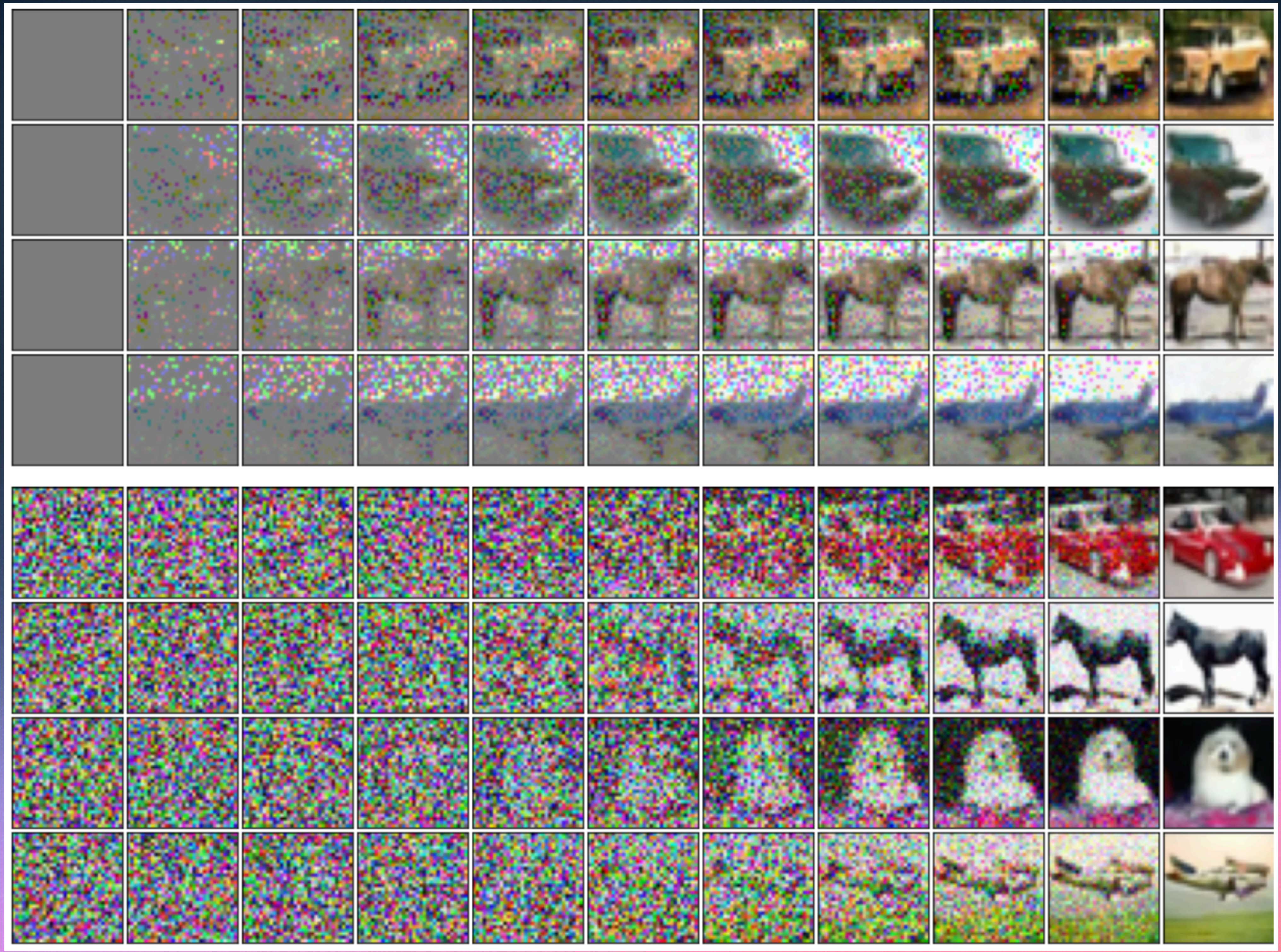
- D3PM outperformed by StyleGAN2 + ADA and NCSN++
- Performs better than original continuous diffusion model (NLL)
- From the D3PM models:
  - Gaussian + logistic parameterization + hybrid loss function performs best
  - All absorbing state and uniform models perform worse than any gaussian model

Results

Example Images

D3PM Absorbing

D3PM Gaussian + Logistic



Source: https://doi.org/10.48550/arXiv.2107.03006

Conclusion

- D3PMs use matrices with structured transition probabilities
- These are used in the noising process and learned by the denoising process
- Overall strong results, especially since it allows wide range of data
  - For text; autoregressive approaches are still better
  - For images; continuous diffusion models are better for image quality
- More work on: noise schedule, loss function or more timesteps might improve results dramatically

Sources

- https://doi.org/10.48550/arXiv.2107.03006
  - (Structured Denoising Diffusion Models in Discrete State-Spaces by Austin et al.)
- https://research.nvidia.com/publication/2022-03_tackling-generative-learning-trilemma-denoising-diffusion-gans-0
  - (Tackling the Generative Learning Trilemma with Denoising Diffusion GANs by Xiao et al.)
- https://doi.org/10.48550/arXiv.2209.00796
  - (Diffusion Models: A Comprehensive Survey of Methods and Applications by Yang et al.)

Q&A