Inverse problems with diffusion models

Valentin De Bortoli

March 12, 2023

Summary of the previous lecture (1/4)

- In the previous lecture we developed some **theory** for **score-based generative modeling**:
  - Continuous **time-reversal**.
  - **Approximation theorem**.
  - Connection with **Normalizing Flows**.
  - **Accelerations** of SGMs.
- Recall the basics of **SGM**:
  - Sample a **forward trajectory**, noising the distribution.

$$\boxed{X_{k+1} = X_k - \gamma X_k + \sqrt{2\gamma} Z_{k+1} \ .}$$

  - Sample a **backward trajectory** via **ancestral sampling**.

$$\boxed{X_k = X_{k+1} + \gamma\{X_{k+1} + \mathbf{s}_\theta(k\gamma, X_{k+1})\} + \sqrt{2\gamma} Z_{k+1} \ .}$$

  - Backward sampling relies on learning the **score** (**score-matching**)

$$\boxed{\mathbf{s}_{\theta^\star}(k\gamma, \cdot) = \arg\min_\theta \{\mathbb{E}[\|\mathbf{s}_\theta(k\gamma, X_k) - \nabla \log p_{k|0}(X_k|X_0)\|^2] \ : \ f \in \mathrm{L}^2(p_k)\} \ .}$$

Summary of the previous lecture (2/4)

Convergence of diffusion models (De Bortoli et al., 2021)

- Assume there exists $M \geq 0$ such that for any $t \in [0, T]$ and $x \in \mathbb{R}^d$

$$||\mathbf{s}_{\theta^\star}(t, x) - \nabla \log p_t(x)|| \leq M \, ,$$

  with $\mathbf{s}_{\theta^\star} \in C([0, T] \times \mathbb{R}^d, \mathbb{R}^d)$ and regularity conditions on the density of π w.r.t. the Lebesgue measure and its gradients.

- Then there exist $B, C, D \geq 0$ s.t. for any $N \in \mathbb{N}$ and $\{\gamma_k\}_{k=1}^N$ the following hold:

$$\boxed{||\mathcal{L}(Y_N) - \pi||_{\mathrm{TV}} \leq B \exp[-T] + C(M + \gamma^{1/2}) \exp[DT] \, .}$$

  where $T = N\gamma$.

- **A few remarks**:
  - The assumption on π is *not* satisfied if π defined on a **manifold** of $\mathbb{R}^d$ with dimension $p < d$.
  - The approximation assumption is strong and could be **relaxed**.
  - The term $\exp[DT]$ can be improved and turned into a **polynomial dependency**.

Summary of the previous lecture (3/4)

- Having a **deterministic** model is useful for:
  - **Likelihood computation**
  - **Interpolation**
  - **Temperature scaling**
- We can explore the **latent structure**.

Figure 1: Interpolation with ODE. Image extracted from Song et al. (2021).

Summary of the previous lecture (4/4)

- For **high-quality** image sampling **vanilla** SGMs are notably **slow**.

A critical drawback of these models is that they require many iterations to produce a high quality sample. For DDPMs, this is because that the generative process (from noise to data) approximates the reverse of the forward *diffusion process* (from data to noise), which could have thousands of steps; iterating over all the steps is required to produce a single sample, which is much slower compared to GANs, which only needs one pass through a network. For example, it takes around 20

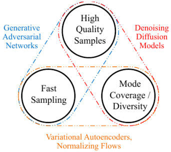
control the generation sample. To obtain high-quality synthesis, a large number of denoising steps is used (i.e. 1000 steps). A notable property of the diffusion process is a closed-form formulation of

network). Although very powerful, score-based models generate data through an undesirably long iterative process; meanwhile, other state-of-the-art methods such as GANs generate data from a single forward pass of a neural network. Increasing the speed of the generative process is thus an active area of research.

denoises the samples under the fixed noise schedule. However, DDPMs often need hundreds-to-thousands of denoising steps (each involving a feedforward pass of a large neural network) to achieve

However, GANs are typically much more efficient than DDPMs at generation time, often requiring a single forward pass through the generator network, whereas DDPMs require hundreds of forward passes through a U-Net model. Instead of learning a generator directly, DDPMs learn to convert

A major downside to score-based generative models is that they require performing expensive MCMC sampling, often with a thousand steps or more. As a result, they can be up to three orders of magnitude slower than GANs, which only require a single network evaluation. To address this issue, Denoising Diffusion Implicit Models, or DDIMs, have been



Outline of the course

- We study **diffusion models** in the setting of **inverse problems**.
- **Goal of the course**:
  - Present techniques to solve **inverse problems** in our framework.
  - Present an end-to-end **text-to-image** model.
- **Outline of the course**
  - **Techniques and tricks** in inverse problem diffusion models.
  - Deep-dive in **Imagen**.

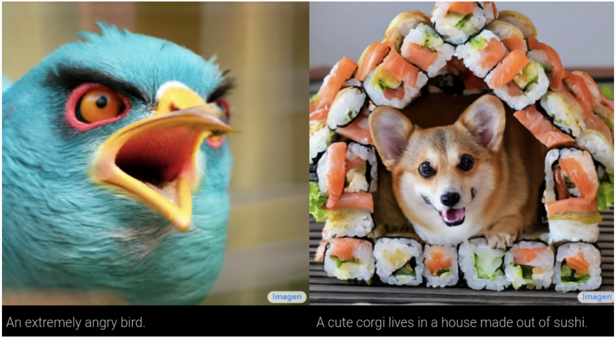


Figure 2: Some outputs of the Imagen model Saharia et al. (2022).

Inverse problems and diffusion models

Illustrative example: astronomical image reconstruction

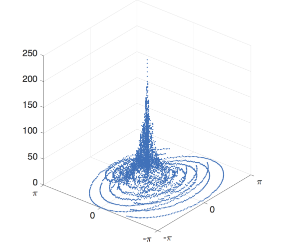
- **Recover** $x \in \mathbb{R}^d$ from low-dimensional degraded observation

$$y = M\mathcal{F}x + w,$$

- $\mathcal{F}$ is the continuous Fourier transform, $M \in \mathbb{C}^{m \times d}$ is a measurement mask operator, and w is Gaussian noise. We use the model

$$\boxed{p(x|y) \propto \exp\left(-\|y - M\mathcal{F}x\|^2/2\sigma^2 - \theta\|\Psi x\|_1\right)\mathbf{1}_{\mathbb{R}^n_+}(x).}$$

- Now, with a **diffusion model** prior!



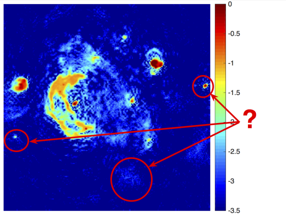


Figure 3: Radio-interferometric image reconstruction of the `W28 supernova`. Credit to Marcelo Pereyra. Left: $\|y\|$, Right: $\hat{x}_{MAP}$.

Diffusion models for inverse problems

- **Question:** how to use denoising diffusion models for inverse problems?
- We present several techniques:
  - **Amortization**
  - **Replacement** (with or without correction)
  - **Conditional guidance**
  - **Denoising Diffusion Restoration Models**
- Main applications:
  - **Inpainting, deblurring**
  - **Class conditional generative modelling**
  - **Text-to-image**

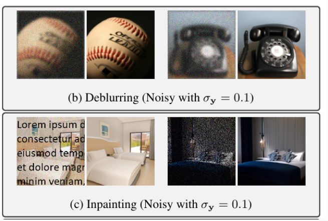


Figure 4: Image extracted from Kawar et al. (2022).

Amortization

- The simplest technique: **amortize** everything.
- **Score matching** techniques: Vincent (2011); Hyvärinen (2005)

$$\nabla \log p_{k+1}(x_{k+1}|y) = \mathbb{E}_{p_{0|k+1,y}}[\nabla \log p_{k+1|0}(x_{k+1}|X_0)].$$

  - **Loss function**: $\boxed{\ell(\mathbf{s}_{k+1}) = \mathbb{E}[\|\mathbf{s}_{k+1}(X_{k+1}, Y) - \nabla \log p_{k+1|0}(X_{k+1}|X_0)\|^2].}$
  - Algorithm: replace $\nabla \log p_{k+1}$ by $\mathbf{s}_{k+1}$.
- Same algorithm as before but instead of sampling X_0 and then noise it, sample (X_0, Y) and then noise it.
- **Advantages:**
  - Straightforward to implement (just another input to the network).
  - Works for generic data.
- **Problems:**
  - What if I only want to train one generative model?
  - What if at inference size y has a different size than the training samples?

The replacement method

- Second technique: **replacement** technique.
- We only train **one** diffusion model.
- Example of inpainting:
  - Train a denoising diffusion model.
  - At inference time, we observe part of the image (y with a mask m)
  - Diffuse y forward in time $Y_{0:N}$
  - Sample $X_N \sim \mathrm{N}(0, \mathrm{Id})$
  - Apply the backward diffusion step:
    $\hat{X}_n = X_{n+1} + \gamma X_{n+1} + 2\gamma \mathbf{s}_\theta(X_n) + \sqrt{2\gamma} Z_n$
  - **Replace** using $X_n = m\hat{X}_n + (1 - m) Y_n$ (pointwise multiplication)
  - Go back to the backward diffusion step and iterate.
- **Advantages:**
  - Only one generative model to train
  - Straightforward to implement
  - Very useful in protein modeling
- **Problems:**
  - Only work on specific problems (mask)
  - No guarantee of convergence

A particle filtering point of view (1/2)

- Our goal is to **sample from $p(x_{0:T}|y_{0:T})$** (where here $y_{0:T}$ is a forward trajectory initialized at y).
- Denote the set of target $\{\pi_t\}_{t=0}^{N}$ such that

$$\boxed{\pi_t = p(x_{t:T}|y_{t:T}).}$$

- The **replacement** procedure:
  - At time T we sample from $p(x_T) \approx p(x_T|y_T)$ (independence).
  - Then, at time t we sample from $p(x_t|x_{t+1}, y_{t+1})$ (sample from $p(x_t, y_t|x_{t+1}, y_{t+1})$ and discard y_t).
  - **But** if we start from $x_{t+1:T} \sim \pi_{t+1}$ then

$$\begin{aligned} x_{t:T} &\sim p(x_t|x_{t+1}, y_{t+1})\pi_{t+1}(x_{t+1:T}) \\ &= p(x_t|x_{t+1}, y_{t+1})p(x_{t+1:T}|y_{t+1:T}) \\ &\neq \pi_t(x_{t:T}) = p(x_{t:T}|y_{t:T}). \end{aligned}$$

  - We have **lost the information** about y_t.

A particle filtering point of view (2/2)

- We have the following **proposal**

$$x_t \sim p(x_t | x_{t+1}, y_{t+1}).$$

- The **extended proposal** is $p(x_t | x_{t+1}, y_{t+1}) \pi_{t+1:T}(x_{t+1:T})$ and we have

$$\boxed{\pi_{t:T} / [p(x_t | x_{t+1}, y_{t+1}) \pi_{t+1:T}(x_{t+1:T})] = p(x_t, y_t | x_{t+1}, y_{t+1}) / p(x_t | x_{t+1}, y_{t+1}).}$$

- This quantifies the **mismatch** in the proposal.
- A simplification

$$p(x_t, y_t | x_{t+1}, y_{t+1}) / p(x_t | x_{t+1}, y_{t+1}) = p(y_t | x_t, y_{t+1}, x_{t+1}).$$

- In a **masked** model we have $x_t \perp y_t$ conditionally to x_{t+1}, y_{t+1} and therefore

$$\boxed{\pi_{t:T} / [p(x_t | x_{t+1}, y_{t+1}) \pi_{t+1:T}(x_{t+1:T})] = p(y_t | x_{t+1}, y_{t+1}).}$$

- Therefore, we need to **reweight** by $p(y_t | x_{t+1}, y_{t+1})$.

SMC-Diff method

- This procedure of **proposal**/**reweighting** is called **Particle Filtering**, Doucet et al. (2009).
  - Many applications in statistics (optimal estimation problems).
  - **Bayesian filtering** methods: **Kalman Filter**, Extended Kalman Filter.
- The complete methodology:
  - **Diffuse y (forward)** and get the trajectory $y_{0:T}$
  - Start with N particles distributed according to $p(x_T)^{\otimes k}$
  - **Update the N particles** according to $p(x_t^k | x_{t+1}^k, y_{t+1})$ for each $k \in \{1, \ldots, N\}$ (independently).
  - **Resample the N particles** with weight proportional to $\{p(y_t | x_{t+1}^k, y_{t+1})\}_{k=1}^N$
- Procedure described in Trippe et al. (2022) (SMC-Diff).
- One potential drawback: **scaling** with the dimension.

Iterative replacement method

Figure extracted from Lugmayr et al. (2022)

- Another trick:
  - Iterating the replacement step Lugmayr et al. (2022) (Repaint)
  - Claim that it increases the **dependency** between the **context** and the generation.

Iterative replacement: algorithm

- At step t and observation y_0
  - Sample from $p(y_{1:T}|y_0)$
  - Sample from $p(x_t|x_{t+1}, y_{t+1})$
  - Sample from $p(x_{t+1}|x_t)$ **(NEW)**
  - Repeat the operation L times **(NEW)**
- The information between x_t and y_t is **mixed** multiple times per time step.

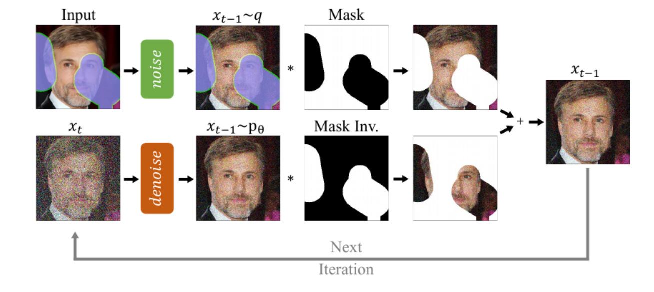


Figure 5: Image extracted from Lugmayr et al. (2022).

Explicit guidance

- Third technique: **conditional guidance**
- Just guide the diffusion with an extra term in the drift

$$\mathbf{s}_\theta(x) \to \mathbf{s}_\theta(x) + \omega \nabla \log p_\phi(y|x)$$

  - ω is the **guidance strength**.
- What is p_ϕ?
  - Classifier in the case of **class conditional sampling** Dhariwal and Nichol (2021).
  - Can be an amortized score model, i.e. (classifier free, Ho and Salimans (2022)) $\nabla \log p_\phi(y|x) \to \mathbf{s}_\theta(x, y) - \mathbf{s}_\theta(x)$
  - Push the samples towards $p(x|y)$ and away from $p(x)$.

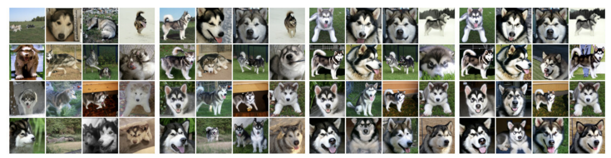


Figure 6: Increasing amount of guidance on the class “malamute” in ImageNet. Image extracted from Ho and Salimans (2022).

Denoising Diffusion Restoration Model

- For **linear models**: **Denoising Diffusion Restoration Models**
  - We assume an observation model of the form $y = \{y_i\}_{i=1}^N$, $y_i = x_0 + \sigma_y^i Z_i$, Z_i i.i.d. Gaussian (we drop the index i for simplicity).
  - Works for more general linear inverse problems using the **SVD** decomposition.
- Take a modified **DDPM approach**.
- Goal: do *not* learn a new model (no need to retrain).

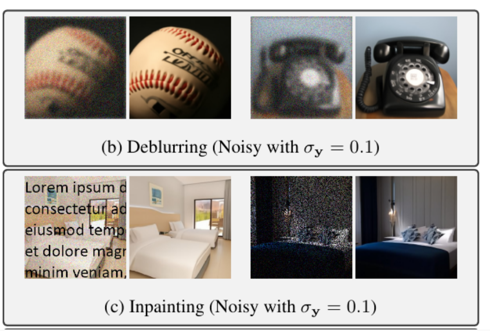


Figure 7: Image extracted from Kawar et al. (2022).

The perturbation model

- In a **DDPM** "like" model [1]

$$q(x_t|x_{t+1}, x_0) = \mathrm{N}(x_t; x_0 + \tfrac{(1-\eta^2)^{1/2}\sigma_t}{\sigma_{t+1}}(x_{t+1} - x_0), \eta\sigma_t),$$

$$q(x_T|x_0) = \mathrm{N}(x_T; x_0, \sigma_T).$$

- Property: for every $t \in \{1, \dots, T\}$, $\boxed{q(x_t|x_0) = \mathrm{N}(x_t; x_0, \sigma_t)}$.
- We consider the following **DDRM** model

$$q(x_t|x_{t+1}, x_0, y) = \begin{cases} \mathrm{N}(x_t, x_0 + \frac{(1-\eta^2)^{1/2}\sigma_t}{\sigma_{t+1}}(x_{t+1} - x_0), \eta\sigma_t) & \text{if } \sigma_y = +\infty \\ \mathrm{N}(x_t, x_0 + \frac{(1-\eta^2)^{1/2}\sigma_t}{\sigma_y}(y - x_0), \eta\sigma_t) & \text{if } \sigma_t \leq \sigma_y \\ \mathrm{N}(x_t, (1-\eta_b)x_0 + \eta_b y, (\sigma_t^2 - \eta_b\sigma_y^2)^{1/2}) & \text{if } \sigma_t \geq \sigma_y \end{cases}$$

$$q(x_T|x_0) = \begin{cases} \mathrm{N}(x_T, x_0, \sigma_T) & \text{if } \sigma_T \leq \sigma_y \\ \mathrm{N}(x_T, y, (\sigma_T^2 - \sigma_y^2)) & \text{if } \sigma_T \geq \sigma_y \end{cases}$$

- **Hyperparameters** (similar to Song et al. (2020)):
  - η, before $\sigma_t \leq \sigma_y$
  - η_b after $\sigma_t \geq \sigma_y$
- Property: for every $t \in \{1, \dots, T\}$, $\boxed{q(x_t|x_0) = \mathrm{N}(x_t; x_0, \sigma_t)}$.

[1] Original DDPM is a discretization of the Ornstein-Uhlenbeck so you won't find these equations in Ho et al. (2020).

Properties of the forward model

- Recall that

$$q(x_t|x_{t+1}, x_0, y) = \begin{cases} \mathrm{N}(x_t, x_0 + \frac{(1-\eta^2)^{1/2}\sigma_t}{\sigma_{t+1}}(x_{t+1} - x_0), \eta\sigma_t) & \text{if } \sigma_y = +\infty \\ \mathrm{N}(x_t, x_0 + \frac{(1-\eta^2)^{1/2}\sigma_t}{\sigma_y}(y - x_0), \eta\sigma_t) & \text{if } \sigma_t \leq \sigma_y \\ \mathrm{N}(x_t, (1-\eta_b)x_0 + \eta_b y, (\sigma_t^2 - \eta_b\sigma_y^2)^{1/2}) & \text{if } \sigma_t \geq \sigma_y \end{cases}$$

$$q(x_T|x_0) = \begin{cases} \mathrm{N}(x_T, x_0, \sigma_T) & \text{if } \sigma_T \leq \sigma_y \\ \mathrm{N}(x_T, y, (\sigma_T^2 - \sigma_y^2)) & \text{if } \sigma_T \geq \sigma_y \end{cases}$$

- **Practical case**: $\eta = 1, \eta_b = 1$
  - When the noise level $\sigma_t \leq \sigma_y$ we rely on x_0.
  - When the noise level $\sigma_t \leq \sigma_y$ we rely on y.
- **DDPM case**: $\eta = 1, \eta_b = 2\sigma_t^2/(\sigma_t^2 + \sigma_y^2)$
  - We recover a DDPM loss.
  - The last equation is only valid if $\eta_b \leq \sigma_t^2/\sigma_y^2$ (in general), with that value of η_b it implies that $\sigma_t \geq \sigma_y$.

The backward model

- As in DDPM:

$$p_\theta(x_t|x_{t+1}, y) = \begin{cases} \mathrm{N}\left(x_t, \hat{x}_0 + \frac{(1-\eta^2)^{1/2}\sigma_t}{\sigma_{t+1}}(x_{t+1} - \hat{x}_0), \eta\sigma_t\right) & \text{if } \sigma_y = +\infty \\ \mathrm{N}\left(x_t, \hat{x}_0 + \frac{(1-\eta^2)^{1/2}\sigma_t}{\sigma_y}(y - \hat{x}_0), \eta\sigma_t\right) & \text{if } \sigma_t \leq \sigma_y \\ \mathrm{N}\left(x_t, (1-\eta_b)\hat{x}_0 + \eta_b y, (\sigma_t^2 - \eta_b\sigma_y^2)^{1/2}\right) & \text{if } \sigma_t \geq \sigma_y \end{cases}$$

$$q(x_T|x_0) = \begin{cases} \mathrm{N}(x_T, 0, \sigma_T) & \text{if } \sigma_T \leq \sigma_y \\ \mathrm{N}(x_T, y, (\sigma_T^2 - \sigma_y^2)) & \text{if } \sigma_T \geq \sigma_y \end{cases}$$

- $q(x_T|x_0) = q(x_T)$ (approximately valid if $\sigma_T \gg 1$).
- $\hat{x}_0$ is the **prediction** of a **generative model** (like DDPM).

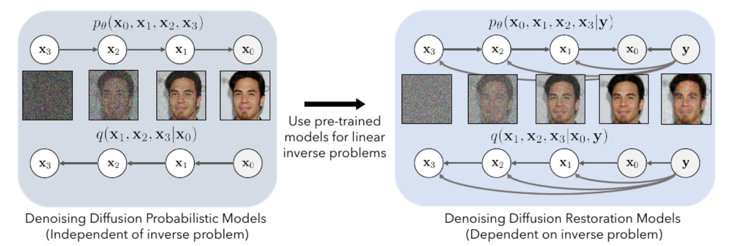


Figure 8: Image extracted from Kawar et al. (2022).

Deep Dive into Imagen

Text-to-image synthesis

Figure 9: Image extracted from Saharia et al. (2022).

A brief history of text-to-image models

- **1.5+ year of progress**:
  - DALLE Ramesh et al. (2021) (24 February 2021)
  - VQ-Diffusion Gu et al. (2022) (29 November 2021)
  - Glide Nichol et al. (2021) (20 December 2021)
  - Stable Diffusion Rombach et al. (2022) (20 December 2021)
  - MidJourney Midjourney (2022) (14 March 2022)
  - DALLE2 Ramesh et al. (2022) (13 April 2022)
  - VQ-GAN Crowson et al. (2022) (18 April 2022)
  - Imagen Saharia et al. (2022) (23 May 2022)
  - E-DIff Balaji et al. (2022) (2 November 2022)
- Earlier work using **GAN approaches** (see references in Gu et al. (2022)).
- A first **comparison** between models Borji (2022).

A black apple and a green backpack.

Figure 10: Image extracted from Saharia et al. (2022).

Overview of the model

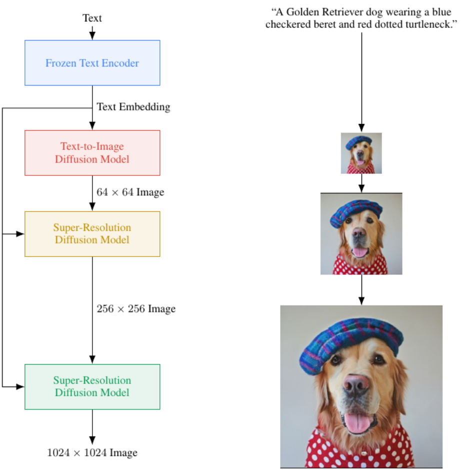


Figure A.4: Visualization of Imagen. Imagen uses a frozen text encoder to encode the input text into text embeddings. A conditional diffusion model maps the text embedding into a 64×64 image. Imagen further utilizes text-conditional super-resolution diffusion models to upsample the image, first $64 \times 64 \rightarrow 256 \times 256$, and then $256 \times 256 \rightarrow 1024 \times 1024$.

Figure 11: Image extracted from Saharia et al. (2022).

Structure of the section

- Presentation of **Imagen** Saharia et al. (2022):
  - **Text-encoder**
  - **Sampler**: conditional guidance and dynamic thresholding
  - **Cascaded Diffusion Models**
  - Architecture (Efficient **U-Net**)
  - Qualitative and quantitative results

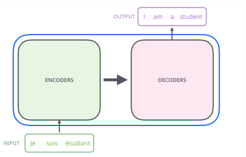
A brain riding a rocketship heading towards the moon. A dragon fruit wearing karate belt in the snow. A strawberry mug filled with white sesame seeds. The mug is floating in a dark chocolate sea.

Figure 12: Image extracted from Saharia et al. (2022).

Text-to-Text Transfer Transformer

- **T5**: Text-to-Text Transfer Transformer Raffel et al. (2020)
  - Based on the **Transformer Architecture** Vaswani et al. (2017)
  - Checkpoint and model available at

`https://huggingface.co/docs/transformers/model_doc/t5`



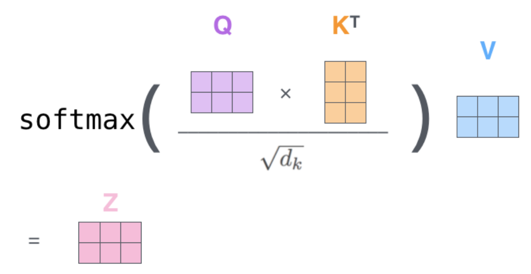
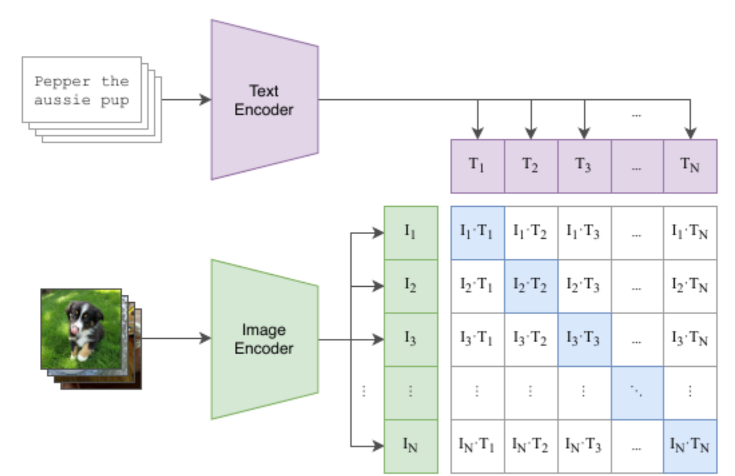


Figure 13: Image extracted from the "Illustrated Transformer" blogpost by Jay Alammar.

- Trained on a **multi-task mixture** (each task is text-to-text). Examples: translation, summarization...

Contrastive Language-Image Pre-Training

- **CLIP**: Contrastive Language-Image Pre-Training Radford et al. (2021)
  - Task: predicting **which caption goes with which image**.
  - Good **text/image representation**



Contrastive learning

- The **training procedure**:
  - We have N (batch) pairs of text/image
  - We get the image embedding $I_e \in \mathbb{R}^{N \times d_e}$ (N is the size of the batch)
  - We get the text embedding $T_e \in \mathbb{R}^{N \times d_e}$ (N is the size of the batch)
  - We compute the $L = T_e I_e^\top \in \mathbb{R}^{N \times N}$
- We compute the **cross-entropy loss** for text and image
  - $\{w^T_{i,j}\}_{i,j=1}^N = \{L_{i,j} / \sum_{k=1}^N L_{i,k}\}_{i,j=1}^N$, $\ell^T = \sum_{i=1}^N \log(w^T_{i,i})$
  - $\{w^I_{i,j}\}_{i,j=1}^N = \{L_{i,j} / \sum_{k=1}^N L_{k,j}\}_{i,j=1}^N$, $\ell^I = \sum_{i=1}^N \log(w^I_{i,i})$
  - Final loss $\ell = \ell^I + \ell^T$

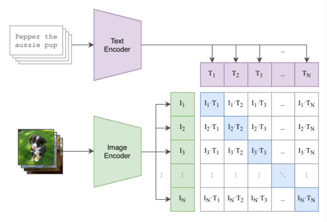


Figure 15: Image extracted from Radford et al. (2021).

- Reminiscent of **contrastive learning** in **unsupervised learning**.

Explicit guidance

- Recall the **conditional guidance** technique
- Just guide the diffusion with an extra term in the drift

$$\mathbf{s}_\theta(x) \to \mathbf{s}_\theta(x) + \omega \nabla \log p_\phi(y|x)$$

  - ω is the **guidance strength**.
- What is p_ϕ?
  - Classifier in the case of **class conditional sampling** Dhariwal and Nichol (2021).
  - Can be an amortized score model, i.e. (classifier free, Ho and Salimans (2022)) $\nabla \log p_\phi(y|x) \to \mathbf{s}_\theta(x, y) - \mathbf{s}_\theta(x)$
  - Push the samples towards $p(x|y)$ and away from $p(x)$.

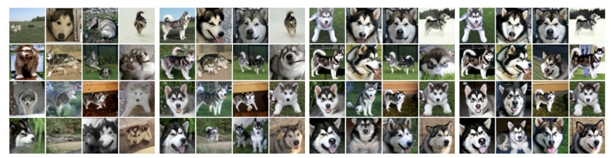


Figure 16: Increasing amount of guidance on the class “malamute” in ImageNet. Image extracted from Ho and Salimans (2022).

Classifier-free guidance

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond}: probability of unconditional training

```
repeat
    (x, c) ~ p(x, c)                          ▷ Sample data with conditioning from the dataset
    c ← ∅ with probability p_uncond           ▷ Randomly discard conditioning to train unconditionally
    λ ~ p(λ)                                  ▷ Sample log SNR value
    ε ~ N(0, I)
    z_λ = α_λ x + σ_λ ε                       ▷ Corrupt data to the sampled log SNR value
    Take gradient step on ∇_θ ‖ε_θ(z_λ, c) − ε‖²   ▷ Optimization of denoising model
until converged
```

Figure 17: Training of the classifier free guidance model. Image extracted from Ho and Salimans (2022)

- p_{uncond} is usually set to 0.2.
  - **Small portion** of training dedicated to unconditional model.
  - Guidance strength: interpolation between **diversity** and **fidelity**.

Figure 18: Image extracted from Ho and Salimans (2022)

Dynamic thresholding

- In Saharia et al. (2022):
  - **Classifier-free guidance**
  - Score is amortized w.r.t. the **text embedding**
- Classifier-free outputs with **strong guidance** are **saturated**. Solution: thresholding (at sampling time)
  - **Static thresholding**: project the update in the range $[-1, 1]$.
  - **Dynamic thresholding**: project the update in the range. Set s to a percentile absolute pixel value.Threshold to the range $[-s, s]$ if $s \geq 1$ and divide by s.

(a) No thresholding. (b) Static thresholding. (c) Dynamic thresholding.

Figure 19: Image extracted from Saharia et al. (2022).

Cascaded Diffusion Models

- **Cascading** for diffusion models was introduced in Ho et al. (2022)

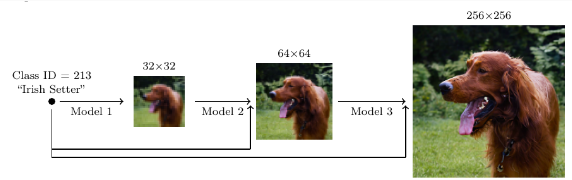


Figure 20: Image extracted from Ho et al. (2022).

- **Useful technique** in generative modeling Menick and Kalchbrenner (2018); Razavi et al. (2019). Below z is an upsampled version of the previous

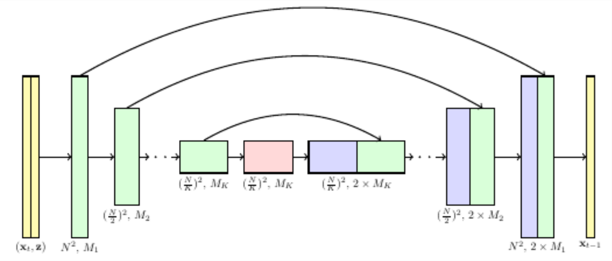


Figure 21: Image extracted from Ho et al. (2022).

Gaussian and truncated conditioning

- It is beneficial to **condition** on a noisy version of the **low-resolution image**.
  - Strategy 1: **Gaussian conditioning**
  - Strategy 2: **Truncated conditioning**

Algorithm 2 Sampling from a two-stage CDM with Gaussian conditioning augmentation

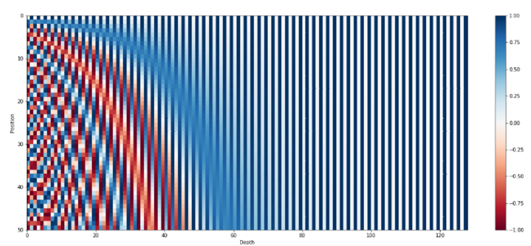
Require: $\mathbf{c}$: class label
Require: s: conditioning augmentation truncation time
$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
if using truncated conditioning augmentation **then**
  for $t = T, \ldots, s+1$ **do**
    $\mathbf{z}_{t-1} \sim p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{c})$
  end for
else
  for $t = T, \ldots, 1$ **do**
    $\mathbf{z}_{t-1} \sim p_\theta(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{c})$
  end for
  $\mathbf{z}_s \sim q(\mathbf{z}_s|\mathbf{z}_0)$ ▷ Overwrite previously sampled value of $\mathbf{z}_s$
end if
$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{z}_s, \mathbf{c})$
end for
return $\mathbf{x}_0$

Figure 22: Image extracted from Ho et al. (2022).

Positional encoding

- The time is **one-dimensional**. We create a **feature vector** associated with it.
- This is like a **continuous** version of the **one-hot encoding**.

```
class TimeEmbedding(hk.Module):
    def __init__(self, dim):
        super().__init__()
        self.dim = dim

    def __call__(self, time):
        half_dim = self.dim // 2
        embeddings = jnp.log(10000) / (half_dim - 1)
        embeddings = jnp.exp(jnp.arange(half_dim) * -embeddings)
        embeddings = time[:, jnp.newaxis] * embeddings[jnp.newaxis, :]
        embeddings = jnp.concatenate(
            (jnp.sin(embeddings), jnp.cos(embeddings)), axis=-1
        )
        return embeddings
```



Resnet block

```python
class Block(hk.Module):
    def __init__(self, dim_out, groups=8):
        super().__init__()
        self.proj = hk.Conv2D(dim_out, kernel_shape=3, padding=(1, 1))
        self.norm = hk.GroupNorm(groups)
        self.act = jax.nn.silu

    def __call__(self, x):
        x = self.proj(x)
        x = self.norm(x)
        x = self.act(x)
        return x
```

```python
class ResnetBlock(hk.Module):
    """https://arxiv.org/abs/1512.03385"""

    def __init__(self, dim_out, groups=8, change_dim=False):
        super().__init__()
        self.mlp = hk.Sequential([jax.nn.silu, hk.Linear(dim_out)])
        self.block1 = Block(dim_out, groups=groups)
        self.block2 = Block(dim_out, groups=groups)
        self.res_conv = (
            hk.Conv2D(dim_out, kernel_shape=1, padding=(0, 0))
            if change_dim
            else lambda x: x
        )

    def __call__(self, x, time_emb):
        h = self.block1(x)

        time_emb = self.mlp(time_emb)
        # We add new axes to the time embedding to for broadcasting.
        h = time_emb[:, jnp.newaxis, jnp.newaxis] + h

        h = self.block2(h)
        return h + self.res_conv(x)
```

Skip connection

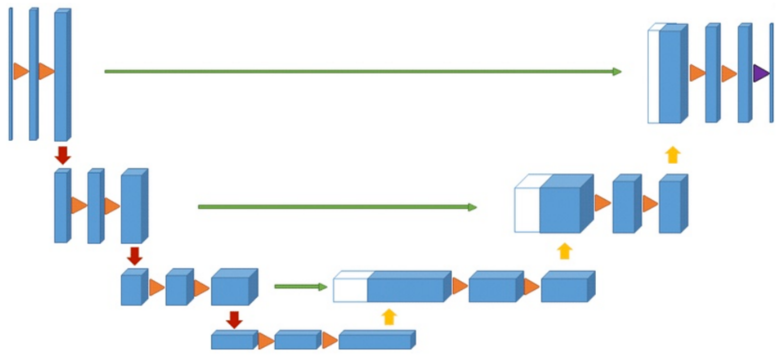
- In the **U-Net** architecture a key component is the **skip connection**.
- Recover **information** lost during the **downsampling**.

```python
def __call__(self, x, time):
    x = self.init_conv(x)
    t = self.time_mlp(time)

    h = []
    # downsample
    for block1, block2, downsample in self.downs:
        x = block1(x, t)
        x = block2(x, t)
        h.append(x)
        x = downsample(x)

    # bottleneck
    x = self.mid_block1(x, t)
    x = self.mid_block2(x, t)

    # upsample
    for block1, block2, upsample in self.ups:
        x = jnp.concatenate((x, h.pop()), axis=-1)
        x = block1(x, t)
        x = block2(x, t)
        x = upsample(x)

    x = self.final_block(x, t)
    return self.final_conv(x)
```

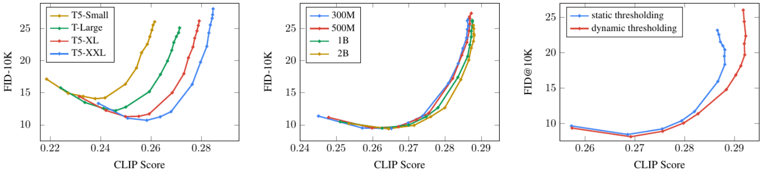
Skip connection

The Unet architecture

- First introduced for **biomedical image segmentation** Ronneberger et al. (2015).
- Used in Ho et al. (2020); Song et al. (2021).
- Putting everything together.
  - **Downsampling** (Resnet block + time embedding)
  - **Upsampling** (Resnet block + time embedding)
  - **Skip connections**



Importance of the text encoder



(a) Impact of encoder size. (b) Impact of U-Net size. (c) Impact of thresholding.

- **Pareto curve** (CLIP score/FID score)
- Sweep over **multiple guidance values** from 1 to 10
- Scaling the **text encoder** is more important than scaling the Unet
- Classifier-free guidance with **large weight**

Some failures

Figure 23: Image extracted from Saharia et al. (2022).

GANs strike back

- Recently: **GigaGAN** Kang et al. (2023)
  - State-of-the-art on the COCO dataset (FID).
  - **Very fast generation** (0.13s for a 512 $\times$ 512 image).

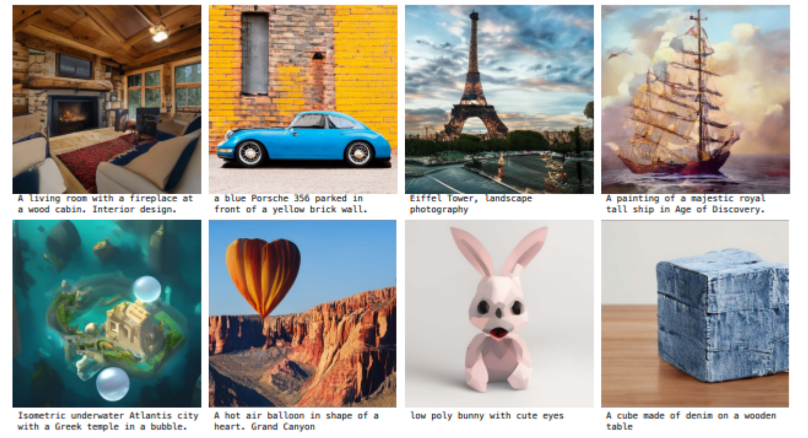


Figure 24: Image extracted from Kang et al. (2023).

References

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, et al. ediffi: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

Ali Borji. Generated faces in the wild: Quantitative comparison of stable diffusion, midjourney and dall-e 2. *arXiv preprint arXiv:2210.00586*, 2022.

Katherine Crowson, Stella Biderman, Daniel Kornis, Dashiell Stander, Eric Hallahan, Louis Castricato, and Edward Raff. Vqgan-clip: Open domain image generation and editing with natural language guidance. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVII*, pages 88–105. Springer, 2022.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, 2021.

Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10696–10706, 2022.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23(47):1–33, 2022.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis, 2023. URL https://arxiv.org/abs/2303.05511.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *arXiv preprint arXiv:2201.11793*, 2022.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022.

Jacob Menick and Nal Kalchbrenner. Generating high fidelity images with subscale pixel networks and multidimensional upscaling. *arXiv preprint arXiv:1812.01608*, 2018.

Midjourney. Midjourney. 2022.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

Brian L Trippe, Jason Yim, Doug Tischer, Tamara Broderick, David Baker, Regina Barzilay, and Tommi Jaakkola. Diffusion probabilistic modeling of protein backbones in 3d for the motif-scaffolding problem. *arXiv preprint arXiv:2206.04119*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.