Structural Causal Models and Diffusion Probabilistic Models

Turmanidze, Lado
turmanidze.lado@kiu.edu.ge

Cao, Zhuo
z.cao@fz-juelich.de

Summer 2025

Abstract

The integration of Structural Causal Models (SCMs) with Diffusion Probabilistic Models (DDPMs) presents a novel approach to enhancing causal inference and representation learning. SCMs offer a comprehensive framework for clarifying the relationships among variables and the mechanisms that drive their interactions. Meanwhile, Diffusion Probabilistic Models have gained prominence for their ability to generate complex data distributions. By combining these two methodologies, new strategies emerge that utilize the strengths of SCMs to refine the generative capabilities of diffusion models and incorporate counterfactual reasoning. This collaboration not only deepens theoretical insights but also paves the way for innovative applications in areas such as molecular dynamics and counterfactual analysis. Preliminary findings illustrate the effectiveness of this integrated approach, and future research directions are proposed to further investigate its potential.

1 Causal Inference

Traditional machine learning models often identify correlations between variables, but they do not inherently provide insights into whether one variable causes changes in another. This limitation can lead to misleading conclusions, especially in complex systems where multiple factors interact. Structural Causal Models (SCMs) allow for the modeling of the causal relationships at play. By representing the variables involved within a directed acyclic graph (DAG), one can identify potential confounders and assess the direct effects of the medication on recovery. SCMs enable the analysis of counterfactual scenarios, which can answer questions of the form "What would have happened if...?".

We will restate important definitions and properties from [1].

1.1 Graphs

Graph

A **graph** is a tuple $\mathcal{G} = (V, E)$ where V is a set of nodes and $E \subseteq V \times V$ is a set of edges connecting the nodes.

Directed Edge Relations

Let $\rightarrow, \leftarrow \in V \times V$ be relations that indicate the existence of directed edges: $u \rightarrow v$ means there is an edge from u to v and $u \leftarrow v$ means there is an edge from v to u.

Directed Graph

Notice that the general graph definition does not impose directionality on edges. A **directed graph** is a graph $\mathcal{G} = (V, E)$ where each edge is directed, represented as an ordered pair. Thus, $u \rightarrow v$ does not imply $v \rightarrow u$ for all $u, v \in V$.

Path

A **path** in a directed graph $\mathcal{G} = (V, E)$ is a sequence of nodes $u_{i_1} \rightarrow u_{i_2} \rightarrow \cdots \rightarrow u_{i_k}$ such that for all $1 \leq i < k$ there exists a directed edge $u_{i_k} \rightarrow u_{i_{k+1}}$, with $u_{i_1}, u_{i_2}, \cdots, u_{i_k} \in V$.

Path Notation

We let $p_{i,j}$ denote a path between nodes u_i and u_j, and $\neg p_{i,j}$ indicates there is no path between u_i and u_j. We denote $up_{i,j}$ and $\neg up_{i,j}$ (paths with undirected edges), and $aup_{i,j}$ and $\neg aup_{i,j}$ (acyclic undirected paths), respectively.

Partially Directed Acyclic Graph (PDAG)

A graph $\mathcal{G} = (V, E)$ is a **partially directed acyclic graph (PDAG)** if it contains no directed cycles; that is, there is no pair (u_i, u_j) such that there are directed paths from u_i to u_j and from u_j to u_i:

$$p_{i,j} \wedge \neg p_{j,i} \vee p_{j,i} \wedge \neg p_{i,j}$$

Directed Acyclic Graph (DAG)

A graph $\mathcal{G} = (V, E)$ is a **directed acyclic graph (DAG)** if it is a directed graph and $\neg p_{i,i}$ for all $u_i \in V$.

Alternatively, a DAG is a special case of a PDAG in which all edges are directed.

Completed Partially Directed Acyclic Graph (CPDAG)

A graph $\mathcal{G} = (V, E)$ is a **completed partially directed acyclic graph (CPDAG)** if:

1. $\mathcal{G}$ is a PDAG that additionally contains undirected edges $(u_i \leftrightarrow u_j)$.
2. The undirected edges can be oriented so the resulting directed graph is acyclic. Specifically, for any undirected edge $(u_i \leftrightarrow u_j)$, it can be oriented as either $u_i \rightarrow u_j$ or $u_j \rightarrow u_i$ without introducing a directed cycle.

In summary, a CPDAG captures the conditional independence relationships among a set of variables: directed edges indicate direct dependencies and undirected edges denote undetermined directionality. For instance, a directed edge from X to Y suggests that all DAGs represented by the CPDAG contain the edge $X \rightarrow Y$. An undirected edge between X and Y indicates that both $X \rightarrow Y$ and $Y \rightarrow X$ are possible in different DAGs in the equivalence class, but X and Y are always adjacent.

Skeleton

A **skeleton** commonly refers to a graph $\mathcal{G} = (V, E)$ that discards edge directions, treating all edges as undirected.

Ancestor, Descendant, and Parent

Given a directed acyclic graph $\mathcal{G} = (V, E)$, a node $u \in V$ is an **ancestor** of $v \in V$ if there exists a directed path from u to v in $\mathcal{G}$ (excluding the trivial path). The set of all parents for node u_i is $\mathbf{PA}_i = \{v \mid (v, u_i) \in E\}$. Conversely, v is a **descendant** of u if there is a directed path from u to v. A node is not considered its own ancestor or descendant.

Adjacency

Two nodes u_i and u_j are **adjacent** if $u_i \rightarrow u_j \in E$ or $u_j \rightarrow u_i \in E$.

d-separation

Let $\mathcal{G} = (V, E)$ be a directed acyclic graph, and let $X, Y, Z \subseteq V$ be pairwise disjoint sets of nodes. A path p between a node in X and a node in Y is **blocked** by Z (which is known) if there exists a node $v \in V$ on p satisfying one of the following:

1. v is a **collider** on p (i.e., the arrows on p "meet" at v: $a \rightarrow v \leftarrow b$ for some $a, b \in V$), and neither v nor any descendant of v is in Z;
2. v is **not a collider** on p (i.e., p passes through v as $a \rightarrow v \rightarrow b$, $a \leftarrow v \rightarrow b$, or $a \leftarrow v \leftarrow b$ for some $a, b \in V$), and $v \in Z$.

The set Z is said to **block** the path p if at least one such v exists.

We say that X and Y are **d-separated** by Z in $\mathcal{G}$ if every path between any node in X and any node in Y is blocked by Z.

If X and Y are d-separated by Z, we write $X \perp\!\!\!\perp Y \mid Z$ in $\mathcal{G}$.

Intuitively, a collider node on a path only allows information to flow if it or its descendants are observed (in Z). If not observed, the collider acts like a closed gate that blocks the path, breaking dependency between variables. Conversely, non-collider nodes block a path by being directly conditioned on. So, even though a collider is not in Z, its presence without conditioning blocks that path, making Z effectively block the connection between variables on that path. This captures how certain structures either transmit or block probabilistic influence depending on observations.

Independence

Let $X, Y, Z \subseteq V$ be (possibly overlapping) sets of nodes in a graph $\mathcal{G}$. We say that X is **independent** of Y given Z, denoted $X \perp\!\!\!\perp Y \mid Z$, if the conditional probability distribution of X given Y and Z is equal to the conditional distribution of X given Z alone; that is, $P(X \mid Y, Z) = P(X \mid Z)$.

This expresses that, once Z is known, knowing Y provides no additional information about X.

Confounder

Let $\mathcal{G} = (V, E)$ be a directed acyclic graph, and let $u_i, u_j \in V$ be two distinct nodes representing variables of interest. A node $u_k \in V \setminus \{u_i, u_j\}$ is called a **confounder** for the relationship between u_i and u_j if the following conditions hold:

1. There exists a path p from u_i to u_j in $\mathcal{G}$ such that u_k is a common ancestor of both u_i and u_j (i.e., there are directed edges leading from u_k to both u_i and u_j).

2. The path p is not blocked by the empty set (i.e., it is open when no nodes are conditioned on).

3. The path p is blocked when conditioning on u_k (i.e., u_k is included in the set of conditioned nodes, and p becomes blocked by conditioning on u_k).

Intuitively, a confounder is a hidden common cause that influences both the treatment and the outcome, creating a spurious association between them. It can make it seem like the treatment affects the outcome when, in fact, the confounder causes changes in both. Identifying and adjusting for confounders is crucial to uncover the true causal effect and avoid misleading conclusions in analysis.

1.2 Cause and Mechanism are independent

The cause C and the mechanism M by which the cause brings the effect E are independent of each other. It does not matter which knife a boy uses to cut his finger; he now knows that all knives can cut his fingers.($C \mapsto$ The boy using a knife, $M \mapsto$ The property of knives being able to cut, $E \mapsto$ The boy cutting his finger). Since M and C are independent, we can perform any kind of intervention on C and deduce that M stays the same. Given a dataset consisting of information about the cause C and the effect E, we have $P(c, e) = P(e \mid c) * P(c)$ with $P(c)$ being the distribution of the cause and $P(e \mid c)$ being the mechanism. This generalizes to Bayesian Network Factorization: $P(u_1, \ldots, u_n) = \prod_i P(u_i \mid \mathbf{PA}_i)$.

Understanding that cause and mechanism are independent is crucial because it allows us to predict the effect under new interventions on the cause without needing to know how the mechanism might change. This stability enables reliable causal inference and generalization beyond observed data, ensuring that altering the cause alone is sufficient to study effects without re-estimating the underlying process.

1.3 Reichenbach's Common Cause Principle

This is a key idea in causal inference that states if X and Y are correlated and the data is unbiased, then one of the following is true:

1. X is the cause of Y.
2. Y is the cause of X.
3. There is a confounder Z that is the cause of both X and Y.

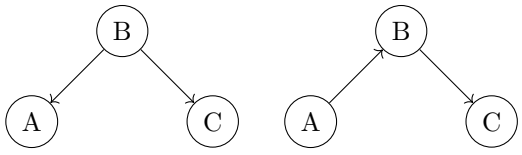
Causal Markov Assumption and Faithfulness

Causal Markov assumption states: "a node is independent of all its non-descendants given the set of all its parents":

$$P(u_i \mid \mathbf{PA}_i) = P(u_i \mid \mathbf{PA}_i, \{\mathbf{PA}_x \mid u_x \in \mathbf{PA}_i\})$$

Markov Equivalence Classes

It is essential to recognize that the same distribution can be Markovian for different causal graphs, whereas multiple data distributions can fulfill the Causal Markov condition concerning $\mathcal{G}$.



For both graphs above:

- A is independent of C conditioned on B
- C is independent of A conditioned on B
- None of the other variables is independent of B

If graphs $\mathcal{G}_0, \mathcal{G}_1$ contain the same independencies, they are in the same Markov equivalence class.

A probability distribution P is said to be **faithful** to a directed acyclic graph $\mathcal{G}$ if every conditional independence that holds in P is implied by the d-separation criterion in $\mathcal{G}$. That is, for all disjoint sets $X, Y, Z \subseteq V$,

$$X \perp\!\!\!\perp Y \mid Z \text{ in } P \implies X \text{ and } Y \text{ are d-separated by } Z \text{ in } \mathcal{G}.$$

Faithfulness ensures that the graph structure captures all and only the conditional independencies present in the distribution.

Finally, we restate **Causal Sufficiency**: all confounders of the relevant variables are observed in the given dataset.

Corollary: edges in the DAG imply causal relationships.

1.4 Causal Assumptions

Causal assumptions are necessary to ensure that the conclusions drawn from data analysis are valid and meaningful. They help to clarify the relationships between variables, address potential biases, and provide a framework for causal inference. Without these assumptions, we risk making erroneous conclusions based on observational data alone. Common assumptions are:

1. Faithfulness
2. Acyclic data generating mechanisms (no feedback loops).
3. No conditioning on unobserved colliders.
4. No unobserved confounding (can be relaxed as in FCI [6]).

Key Idea: Estimate the CPDAG from data, as causal relationships produce conditional independencies that can be utilized to reconstruct aspects of the causal model, specifically its Markov equivalence class.

1.5 Causal Structure Learning Algorithms

Causal structure learning algorithms are crucial for uncovering the causal relationships among variables from observational data. The PC algorithm (Peter & Clark) is a prominent method that identifies these relationships by detecting conditional independencies, constructing a DAG to represent the causal structure [4]. Additionally, the Greedy Equivalence Search (GES) algorithm [5] optimizes the search for the best scoring DAG, while the Fast Causal Inference (FCI) algorithm [6] extends the capabilities of the PC algorithm to account for latent variables and unobserved confounding. Here, we will only recall the PC algorithm from [4]:

PC Algorithm

For $a, b \in V$, let $A_{a,b}^{\mathcal{G}} = \{u \mid u \notin \{a,b\} \wedge (u \leftrightarrow a \in E \vee u \leftrightarrow b \in E)\}$ and $U_{a,b}^{\mathcal{G}} = \{u \mid u \notin \{a,b\} \wedge \exists aup_{a,b} \in G\}$. $A_{a,b}^{\mathcal{G}}$ and $U_{a,b}^{\mathcal{G}}$ are constantly changing as the algorithm progresses, as $\mathcal{G} = (V, E)$ is continually updated.

Algorithm 1 PC Algorithm

Input: Set of vertices V
Output: all graphs (which are actually CPDAGs) with these orientations
Part A:
Form the complete undirected graph $\mathcal{G}$ on V:
  $E := \{\{u_i, u_j\} \mid u_i, u_j \in V\}$
Part B:
$n := 0$
repeat
  for all $(a,b) \in E$ **do**
    if $|A_{a,b}^{\mathcal{G}} \cap U_{a,b}^{\mathcal{G}}| \geq n$ **and** $a \perp\!\!\!\perp b \mid \{S \mid S \subseteq A_{a,b}^{\mathcal{G}} \cap U_{a,b}^{\mathcal{G}} \wedge |S| = n\}$ **then**
      delete (a,b) from E
    end if
  end for
  $n := n + 1$
until $\forall (a,b) \in E$, $|A_{a,b}^{\mathcal{G}} \cap U_{a,b}^{\mathcal{G}}| < n$
Call the resulting undirected graph F
Part C:
for all $\{a, b, c \in V \mid (a,b), (b,c) \in F \wedge (a,c) \notin F\}$ **do**
  if $\forall S \subseteq A_{a,b}^{\mathcal{G}} \cap U_{a,b}^{\mathcal{G}}$, $b \in S : a \not\perp\!\!\!\perp c \mid S$ **then**
    Orient $a \leftrightarrow b \leftrightarrow c$ as $a \rightarrow b \leftarrow c$
  end if
end for

Additionally, one would implement Meek's orientation rules [8] for correctness and completeness: no incorrect orientations occur, and no further orientations can be made (See Figure 1).

Intuitively, one could interpret Part B of the PC algorithm as eliminating as many edges as possible using conditional independence tests. In Part C, the algorithm establishes causal directions for each remaining edge by considering colliders, the assumption that there are no cycles, and any additional arbitrary assumptions.

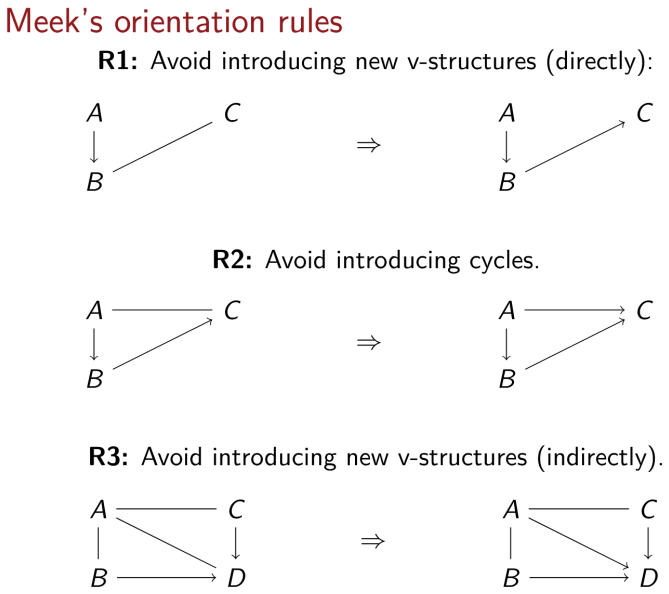


Figure 1: Meek's Orientation Rules from [7]

Using PC on empirical data requires either a significance level to use in the tests or a conditional independence test. For the former, we do not have a principled approach for choosing the appropriate significance level; for the latter, there does not exist a generally correct test of conditional independence that does not rely on some distributional assumptions [9].

If the data is entirely categorical, we can directly test **conditional independence** by use of. a χ^2 test of independence on the multi-way cross tabulation over X, Y, Z.

For mix of binary and numerical variables, one might test for non-association using GLMs with spline-expansions [10].

For the special case when the data are jointly normally distributed, we have $X \perp\!\!\!\perp Y \mid Z \Leftrightarrow \text{cor}(X, Y \mid Z) = 0$ which is equivalent with testing null hypothesis $H_0 : \beta = 0$ in the linear regression model $Y_i = \alpha + \beta \cdot \mathbf{X}_i + \gamma \cdot Z_i + \epsilon_i$.

The **significance level** used for individual tests in the PC algorithm is not a proper significance level for the globally estimated graph, as it does not describe the overall risk of a Type I error. Furthermore, if many tests are conducted and the result of one test informs which test should be conducted next, this

implies a complicated multiple testing issue without obvious solutions. As with everything in statistics, we typically default to a significance level of $\alpha = 0.05$.

1.6 Formal Definition of SCM

Causal graphs are DAGs that depict causal relationships in a binary manner. There is either a cause-effect relationship between two variables (indicated by an edge between them) or there is not. It would be more informative to have a model that includes more details about the causal relationship, such as the ability to quantify the effect. Therefore, we need to formulate a set of equations that describe all causal relationships in our model.

Random noise represents the variability in data that cannot be explained by the model, arising from measurement errors, environmental factors, or inherent randomness. Including random noise (with a specific distribution) in our model can enhance its applicability to real-world scenarios.

In [1], SCMs are defined in following way:

A **structural causal model (SCM)** $\mathfrak{C} := (\mathbf{S}, P_{\mathbf{N}})$ consists of:

1. a collection $\mathbf{S}$ of d structural assignments

$$\mathbf{X}_j := f_j(\mathbf{PA}_j, N_j), \quad j \in \{1, \dots, d\},$$

   where each $\mathbf{PA}_j \subseteq \{\mathbf{X}_1, \dots, \mathbf{X}_d\} \setminus \{\mathbf{X}_j\}$ is the set of parents of $\mathbf{X}_j$;

2. a joint distribution $P_{\mathbf{N}} = P_{N_1,\dots,N_d}$ over the noise variables, which are assumed to be mutually independent, i.e.,

$$P_{\mathbf{N}} = \prod_{j=1}^{d} P_{N_j}.$$

Notice the assignment operator := in the definition. This is similar to assignment in programming, where we can change the value of a (random) variable at any time. This will be useful for interventions.

Structural Causal Models (SCMs) are inherently probabilistic due to the presence of random noise terms, which we assume to be independent. We posit that our model adequately captures all systematic dependencies within the system, leaving no unobserved confounding factors. Should unobserved confounding exist, it would necessitate the inclusion of additional unobserved nodes or a generalization of the SCM framework.

Structural Minimality

Definition of SCMs currently distinguishes between:

$$S_1 : X := N_x, Y = 0 \cdot X + N_y, \quad S_2 : X := N_x, Y := N_Y$$

As this contradicts our intuition, we add the requirement that the functions f_j depend on all of their input arguments: For $k \in \{1, \ldots, d\}$, a function g and a minimal (with respect to set size) subset $\mathbf{PA}_k^-$ such that:

$$f_k(\mathbf{PA}_k, N_k) = g(\mathbf{PA}_k^-, N_k), \quad \forall \mathbf{PA}_k, N_k \text{ with } P(N_k) > 0 \text{ and } \mathbf{PA}_k^- \subsetneq \mathbf{PA}_k$$

In this case, we always choose the latter representation; thus, with this definition of structural minimality, one would choose S_2 over S_1. We assume that structural minimality always holds.

1.7 From Causal Graph to SCM

If a causal graph is constructed by following the aforementioned steps, the quantification step can be achieved through various methodologies from classical statistics and machine learning. Each function f_i may be learned using techniques such as artificial neural networks (ANNs), ordinary least squares, decision trees, or neural networks, which are recognized as the state-of-the-art methods for general structural causal model estimation.

Nevertheless, it is necessary to evaluate the assumptions that can be made (for instance, linearity in the case of ordinary least squares), substantiate these assumptions with theoretical foundations, and implement suitable models accordingly.

1.8 Interventions

Interventional questions are central to causal inference, inquiring about the effects of setting one or more variables to specific values. These questions describe interventions, which involve altering a system by forcing variables to adopt designated values.

This process of intervening is fundamentally different from conditioning, which involves selecting observations based on certain values while ignoring others. Intervening requires all instances to adopt a specified value. As a result, the expression $P(Y \mid \boldsymbol{do}(\mathbf{X}_j := k))$, representing the probability distribution of Y when $\mathbf{X}_j$ is set to a specific value, does not generally equal the conditional probability $P(Y \mid \mathbf{X}_j = k)$.

Because conditioning and intervening are not equivalent, interventional questions cannot be addressed simply by comparing means. SCMs are particularly valuable in this context, as they facilitate the modeling of intervention distributions when proper causal discovery has been conducted or when expert knowledge of the qualitative graph is available. After intervening on a variable, we assert that the new SCM entails a new distribution known as the interventional distribution.

Formal Definition:

Given a SCM $\mathfrak{C}$ with endogenous variables $\mathbf{X}_1, \ldots, \mathbf{X}_n$, an intervention $\mathfrak{C}^{\boldsymbol{do}(\mathbf{X}_i := k)}$ denotes an external manipulation that sets $\mathbf{X}_i$ to the value k, ir-

respective of its natural causes.

Mechanics:
Applying the intervention $\boldsymbol{do}(\mathbf{X}_i := k)$ to the SCM corresponds to:

- Replacing the structural equation for $\mathbf{X}_i$ with a constant assignment: $\mathbf{X}_i := k$.
- Removing all incoming edges into $\mathbf{X}_i$ in the associated causal graph.

To address challenges such as the uncertainty regarding the identification of causal variables, please refer to [11].

2 Diffusion Probabilistic Models

From [12]: "The astonishing growth of generative tools in recent years has empowered many exciting applications in text-to-image and text-to-video generation. The underlying principle behind these generative tools is the concept of *diffusion*, a particular sampling mechanism that has overcome some longstanding shortcomings in previous approaches".

Sohl-Dickstein et al. [14] proposed constructing a chain of conversions instead of a one-step process of variational auto-encoder (VAE) [15]. We define $\mathbf{x}_{0:T} = \{\mathbf{x}_0, \ldots, \mathbf{x}_T\}$ as collection of all state variables from $t = 0$ to $t = T$. Analogous to the encoder and decoder architecture of VAE, they defined forward and reverse processes. In both cases, they consider a sequence of variables $\mathbf{x}_0, \ldots, \mathbf{x}_T$ whose joint distribution is denoted as $q_\phi(\mathbf{x}_{0:T})$ and $p_\theta(\mathbf{x}_{0:T})$ respectively for the forward and reverse processes. Again, analogous to the VAE, the true transition distribution $p(\mathbf{x}_t \mid \mathbf{x}_{t+1})$ (as well as $p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$) is not accessible, so we approximate it with $q_\phi(\mathbf{x}_{0:T})$ and $p_\theta(\mathbf{x}_{0:T})$. To ensure computational tractability, the Markov property is imposed on both processes:

$$\text{Forward}(\mathbf{x}_0 \to \mathbf{x}_T):\ q_\phi(\mathbf{x}_{0:T}) = q(\mathbf{x}_0) \prod_{t=1}^{T} q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1})$$

$$\text{Reverse}(\mathbf{x}_T \to \mathbf{x}_0):\ p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$$

Since transition distributions depend only on their immediate previous state, the overall generation process is broken down into many smaller tasks. The same neural network will be used for T times. This "Divide and Conquer" strategy allows us to use simpler distributions at each step. The properties of a Gaussian ensure that the posterior will remain a Gaussian if both the likelihood and the prior are Gaussian. Since a Gaussian depends only on two hyperparameters, namely mean and variance, it is highly tractable.

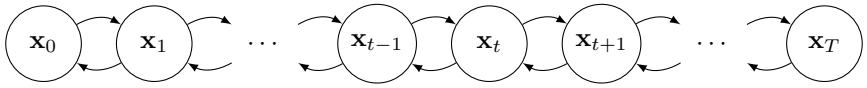


Figure 2: Variational Diffusion Model

2.1 Variational Auto Encoder

VAE-inspired diffusion model is, unsurprisingly, called the **variational diffusion model** (see Figure 2). In this model, $\mathbf{x}_0$ corresponds to the input $\mathbf{x}$ in the VAE, $\mathbf{x}_T$ corresponds to the latent variable z, and $\mathbf{x}_i$, for all $i \in \{1, \ldots, T-1\}$, are (latent) intermediate states.

In a variational diffusion model (and also DDPM which we will discuss later), the (Gaussian) transition distribution is defined as:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t \mid \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbb{I})$$

The mean is $\sqrt{\alpha_t}\mathbf{x}_{t-1}$ and the variance is $(1-\alpha_t)$. The choice of the scaling factor $\sqrt{\alpha_t}$ is to make sure that the variance magnitude is preserved so that it will not explode and vanish after many iterations.

Conditional distribution $q_\phi(\mathbf{x}_t \mid \mathbf{x}_0)$ is given by:

$$q_\phi(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t \mid \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\sqrt{\bar{\alpha}_t})\mathbb{I}) \text{ with } \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

A common point of confusion is why we need to derive transition distributions such as $q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ and $q_\phi(\mathbf{x}_t \mid \mathbf{x}_0)$ if the apparent goal is simply to map an input $p(\mathbf{x}_0)$ to white noise $p(\mathbf{x}_T)$. This is because these distributions define the forward diffusion process, where the data is iteratively corrupted by Gaussian noise in a Markov chain, leading eventually to an standard normal distribution as T becomes large. Importantly, the forward process is constructed to be analytically tractable, allowing us to express transitions like $q_\phi(\mathbf{x}_t \mid \mathbf{x}_0)$ in closed form. The reverse process, which is the focus of training and generation, relies on these closed-form expressions and Markov structure to define and couple each denoising step to its corresponding forward step. Thus, $q_\phi(\mathbf{x}_t \mid \mathbf{x}_0)$ provides a convenient and formally grounded way to describe and implement the forward process, serving as the foundation for learning to reverse the transformation from noise back to meaningful data.

ELBO

Recall the loss function for VAE, evidence lower bound (ELBO) from [15]. For variational diffusion model, ELBO is defined as (with $\mathbf{x}_0 = \mathbf{x}$, $\mathbf{x}_1 \sim q_\phi(\mathbf{x}_1 \mid \mathbf{x}_0)$ and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbb{I})$):

$$\begin{aligned}\mathcal{L}_{\phi,\theta}(\mathbf{x}) = &\mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)\right] \\ &- \mathbb{E}_{q_\phi(\mathbf{x}_{T-1}|\mathbf{x}_0)}\left[\mathbb{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{x}_T \mid \mathbf{x}_{T-1}) \,\|\, p(\mathbf{x}_T)\right)\right] \\ &- \sum_{t=1}^{T-1} \mathbb{E}_{q_\phi(\mathbf{x}_{t-1},\mathbf{x}_{t+1}|\mathbf{x}_0)}\left[\mathbb{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \,\|\, p_\theta(\mathbf{x}_t \mid \mathbf{x}_{t+1})\right)\right]\end{aligned}$$

The first expected value measures how good the initial block is, the second expected value measures how good the final block is, and the sum of the remaining expected values measures how good the intermediate blocks are.

Why expectation? $p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)$ is a function of $\mathbf{x}_1$, so for different values of $\mathbf{x}_1$, the value of $p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)$ will differ. Taking the expectation removes the dependency on $\mathbf{x}_1$.

In prior matching, the KL divergence measures the difference between $q_\phi(\mathbf{x}_T \mid \mathbf{x}_{t-1})$ and $p(\mathbf{x}_T)$. We use the conditional distribution $q_\phi(\mathbf{x}_{T-1} \mid \mathbf{x}_0)$ because $\mathbf{x}_{T-1}$ depends on $\mathbf{x}_0$. The expectation over $q_\phi(\mathbf{x}_{T-1} \mid \mathbf{x}_0)$ provides the KL divergence value for each $\mathbf{x}_{T-1}$ sampled from $q_\phi(\mathbf{x}_{T-1} \mid \mathbf{x}_0)$ and also eliminates the dependency. The same reasoning applies to the intermediate transition blocks.

The challenge with the intermediate transition block of the loss function is that we need to sample $(\mathbf{x}_{t-1}, \mathbf{x}_{t+1})$ from the joint distribution $q_\phi(\mathbf{x}_{t-1}, \mathbf{x}_{t+1} \mid \mathbf{x}_0)$, which is unknown. Even though it is Gaussian by design, we cannot sample future values of $\mathbf{x}_{t+1}$.

By conditioning on $\mathbf{x}_0$, and applying Bayes' theorem, we get:

$$q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)\, q_\phi(\mathbf{x}_t \mid \mathbf{x}_0)}{q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}.$$

Without conditioning on $\mathbf{x}_0$, we cannot sample from $q(\mathbf{x}_{t-1})$, since $\mathbf{x}_{t-1}$ depends on the initial input $\mathbf{x}_0$. Now, we can switch from $q_\phi(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{x}_0)$ to $q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$. This avoids handling two opposing directions and still allows us to check for consistency across transition blocks. The direction $q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ aligns with $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$.

$$\begin{aligned}\mathcal{L}_{\phi,\theta}(\mathbf{x}) = &\mathbb{E}_{q_\phi(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)\right] \\ &- \mathbb{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{x}_T \mid \mathbf{x}_0) \,\|\, p(\mathbf{x}_T)\right) \\ &- \sum_{t=2}^{T} \mathbb{E}_{q_\phi(\mathbf{x}_t|\mathbf{x}_0)}\left[\mathbb{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right)\right]\end{aligned}$$

The initial block remains unchanged. Prior matching has been simplified to the KL divergence between $q_\phi(\mathbf{x}_T \mid \mathbf{x}_0)$ and $p(\mathbf{x}_T)$, since we now condition on $\mathbf{x}_0$ and there is no need to sample from $q_\phi(\mathbf{x}_{T-1} \mid \mathbf{x}_0)$ or take an expectation over it.

The new consistency term differs from the previous one in two key ways. First, the running index t starts at $t = 2$ and ends at $t = T$. Second, the distribution matching is now performed between $q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. Thus, instead of requiring the forward transition to match the reverse transition, we use q_ϕ to construct a reverse transition and match it with p_θ.

Distribution of $q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ takes the form of $q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \boldsymbol{\Sigma}_q(t))$ where:

$$\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0$$

$$\boldsymbol{\Sigma}_q(t) = \frac{(1-\alpha_t)(1-\sqrt{\bar{\alpha}_{t-1}})}{1-\bar{\alpha}_t}\mathbb{I} =: \sigma_q^2(t)\mathbb{I}$$

where $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. For the proof, see page 30 of [12].

Note that $q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ is fully determined by $\mathbf{x}_t$ and $\mathbf{x}_0$, and is not learned through a neural network, unlike in a VAE. There is no statistical learning involved if the hyperparameter α_t is provided.

We now turn our attention to $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$. Since $q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ is Gaussian, we choose $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ to also be Gaussian, allowing for efficient computation of the KL divergence. Specifically, we define

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_\theta(\mathbf{x}_t), \sigma_q^2(t)\,\mathbb{I}),$$

where $\boldsymbol{\mu}_\theta(\mathbf{x}_t)$ is a neural network.

The KL divergence is simplified to:

$$\begin{aligned}
&\mathbb{D}_{\mathrm{KL}}\left(q_\phi(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)\right) \\
&\quad = \mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}\left(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_q^2(t)\,\mathbb{I}\right) \,\|\, \mathcal{N}\left(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}(\mathbf{x}_t), \sigma_q^2(t)\,\mathbb{I}\right)\right) \\
&\quad = \frac{1}{2\sigma_q^2(t)}\left\|\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}(\mathbf{x}_t)\right\|^2
\end{aligned}$$

For the ELBO, the dependency on ϕ is dropped because q_ϕ depends only on $\mathbf{x}_t$ and $\mathbf{x}_0$, which are not learned:

$$\begin{aligned}
\mathcal{L}_\theta(\mathbf{x}) = \mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}\left[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)\right] \\
- \mathbb{D}_{\mathrm{KL}}\left(q(\mathbf{x}_T \mid \mathbf{x}_0) \,\|\, p(\mathbf{x}_T)\right) \\
- \sum_{t=2}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\|\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}(\mathbf{x}_t)\|^2\right]
\end{aligned}$$

with $\mathbf{x} = \mathbf{x}_0$ and $\mathbf{x}_T \sim \mathcal{N}(0, \mathbb{I})$.

The reconstruction term (initial block) can be further simplified:

$$\begin{aligned}\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1) &= \log \mathcal{N}(\mathbf{x}_0 \mid \boldsymbol{\mu}_\theta(\mathbf{x}_1), \sigma_q^2(1)\mathbb{I}) \\ &= \log \frac{1}{\sqrt[d]{2\pi\sigma_q^2(1)}} \exp\left\{-\frac{\|\mathbf{x}_0 - \boldsymbol{\mu}_\theta(\mathbf{x}_1)\|^2}{2\sigma_q^2(1)}\right\} \\ &= -\frac{\|\mathbf{x}_0 - \boldsymbol{\mu}_\theta(\mathbf{x}_1)\|^2}{2\sigma_q^2(1)} - \frac{d}{2}\log(2\pi\sigma_q^2(1))\end{aligned}$$

2.2 Denoising Diffusion Probabilistic Model

Recall $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\mathbf{x}_0$. Defining $\boldsymbol{\mu}_\theta(\mathbf{x}_t) := \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t)$, where $\hat{\mathbf{x}}_\theta(\mathbf{x}_t)$ is another network, we can rewrite

$$\frac{1}{2\sigma_q^2(t)}\|\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) - \boldsymbol{\mu}(\mathbf{x}_t)\|^2$$

as

$$\frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|^2$$

Using the above, and incorporating $\mathbb{E}_{q(\mathbf{x}_1|\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0 \mid \mathbf{x}_1)]$ into the summation, we obtain the ELBO for denoising diffusion probabilistic model (with $\bar{\alpha}_t = \prod_{i=1}^t \alpha_i$) as:

$$\mathcal{L}_\theta^{\text{DDPM}} = -\sum_{t=1}^T \frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)}\left[\|\hat{\mathbf{x}}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|^2\right]$$

For the proof, see page 34 of [12].

Using Monte Carlo approximation of expectation, we can rewrite the optimization problem: $\arg\max_\theta \sum_{\mathbf{x}_0 \in \mathcal{X}} \mathcal{L}_\theta^{\text{DDPM}}$ as

$$\arg\min_\theta \sum_{\mathbf{x}_0 \in \mathcal{X}} \sum_{t=1}^T \frac{1}{M}\frac{1}{2\sigma_q^2(t)}\frac{(1-\alpha_t)^2\bar{\alpha}_{t-1}}{(1-\bar{\alpha}_t)^2}\left\|\hat{\mathbf{x}}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}_t^{(m)}\right) - \mathbf{x}_0\right\|^2$$

where $\boldsymbol{\epsilon}_t^{(m)} \sim \mathcal{N}(0, \mathbb{I})$ and $\mathcal{X}$ is the training set. This loss function has $\hat{\mathbf{x}}_0(\cdot)$ which is a denoiser, so this model is called the denoising diffusion probabilistic model (DDPM).

Training DDPM

The forward diffusion in DDPM generates intermediate variables $\mathbf{x}_{1:T-1}$ by sampling

$$\mathbf{x}_t \sim q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t \mid \sqrt{\bar{\alpha}_t}\mathbf{x}_0,\ (1-\bar{\alpha}_t)\mathbb{I}\right) \Rightarrow \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I}).$$

The forward diffusion process requires no training. Given an input $\mathbf{x}_0$, it produces the entire sequence $\mathbf{x}_{1:T}$. We train a single denoiser network to handle all noise levels, since each $\mathbf{x}_t$ corresponds to a noise level with a different variance $1 - \bar{\alpha}_t$. Training separate denoisers for each time step is computationally infeasible due to the high dimensionality of T

Algorithm 2 Training Algorithm for DDPM

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ $(\mathbf{x}_0 \in \mathcal{X})$
  $t \sim$ Uniform$[1, T]$ (time stamp)
  $\boldsymbol{\epsilon}_t^{(m)} \sim \mathcal{N}(0, \mathbb{I})$
  $\mathbf{x}_t^{(m)} = \bar{\alpha}_t \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}_t^{(m)}$
  Take gradient step on $\nabla_\theta \left\{ \frac{1}{M} \sum_{m=1}^{M} \|\hat{\mathbf{x}}_\theta(\mathbf{x}_t^{(m)}) - \mathbf{x}_0\|^2 \right\}$
until convergence

Inference of DDPM

Once the denoiser $\hat{\mathbf{x}}_\theta$ is trained, we can do the inference. We sample from $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ over the sequence of states $\mathbf{x}_T, \mathbf{x}_{T-1}, \ldots, \mathbf{x}_1$.

$$\mathbf{x}_{t-1} \sim p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_\theta(\mathbf{x}_t), \sigma_q^2(t)\mathbb{I})$$

By reparameterization, we have (for $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I})$):

$$\mathbf{x}_{t-1} = \boldsymbol{\mu}_\theta(\mathbf{x}_t) + \sigma_q(t)\boldsymbol{\epsilon} = \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t}\mathbf{x}_t + \frac{(1 - \alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t) + \sigma_q(t)\boldsymbol{\epsilon}$$

Using the reparameterization above, we run the denoiser T times, starting from the white noise vector $\mathbf{x}_T$ and iteratively denoising back to an estimate of the original input $\bar{\mathbf{x}}_0$.

Algorithm 3 Inference Algorithm for DDPM

Given: $\mathbf{x}_T \sim \mathcal{N}(0, \mathbb{I})$
for $t = T, T-1, \ldots, 1$ **do**
  $\boldsymbol{\epsilon}_t^{(m)} \sim \mathcal{N}(0, \mathbb{I})$
  $\mathbf{x}_{t-1} = \frac{(1 - \bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1 - \bar{\alpha}_t}\mathbf{x}_t + \frac{(1 - \alpha_t)\sqrt{\bar{\alpha}_{t-1}}}{1 - \bar{\alpha}_t}\hat{\mathbf{x}}_\theta(\mathbf{x}_t) + \sigma_q(t)\boldsymbol{\epsilon}$
end for

Useful insights from [13]

We will skip the derivation (algebraic) steps, which can be found in [13], but we highlight a few useful insights below:

We can learn to predict noise

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0 \implies \mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0}{\sqrt{\bar{\alpha}_t}}$$

From this, we obtain the following equivalent expressions:

$$\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha}_t}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\boldsymbol{\epsilon}_0$$

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t) = \frac{1}{\sqrt{\alpha}_t}\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}\sqrt{\alpha_t}}\hat{\boldsymbol{\epsilon}}_0(\mathbf{x}_t)$$

This is useful, as we have expressed $\boldsymbol{\mu}_q(\mathbf{x}_t, \mathbf{x}_0)$ as a function of $\boldsymbol{\epsilon}_0$ rather than $\mathbf{x}_0$. Substituting this into the new ELBO:

$$\mathcal{L}_\theta^{\text{DDPM}}(\mathbf{x}_0, \boldsymbol{\epsilon}_0) = -\sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{x}_t|\mathbf{x}_0)} \left[\frac{1}{2\sigma_q^2(t)} \frac{(1-\alpha_t)^2}{(1-\bar{\alpha}_t)^2\alpha_t} \|\hat{\boldsymbol{\epsilon}}_0\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0\right) - \boldsymbol{\epsilon}_0\|^2 \right]$$

As this ELBO is a function of $\mathbf{x}_0$ and $\boldsymbol{\epsilon}_0$, we need to solve the following optimization problem:

$$\arg\min_\theta \sum_{\mathbf{x}_0 \in \mathcal{X}} \frac{1}{M} \sum_{m=1}^{M} \mathcal{L}_\theta^{\text{DDPM}}(\mathbf{x}_0, \boldsymbol{\epsilon}_0^{(m)})$$

Algorithm 4 Training Algorithm for DDPM (using $\hat{\epsilon}_\theta(\mathbf{x}_t)$)

repeat
  $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ ($\mathbf{x}_0 \in \mathcal{X}$)
  $t \sim$ Uniform$[1, T]$ (time stamp)
  $\boldsymbol{\epsilon}_t^{(m)} \sim \mathcal{N}(0, \mathbb{I})$
  $\mathbf{x}_t^{(m)} = \bar{\alpha}_t\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t^{(m)}$
  Take gradient step on $\nabla_\theta \left\{\|\hat{\epsilon}_\theta(\mathbf{x}_t) - \mathbf{x}_0\|^2\right\}$
until convergence

Algorithm 5 Inference Algorithm for DDPM (using $\hat{\epsilon}_\theta(\mathbf{x}_t)$)

Given: $\mathbf{x}_T \sim \mathcal{N}(0, \mathbb{I})$
for $t = T, T-1, \ldots, 1$ **do**
  $\boldsymbol{z} \sim \mathcal{N}(0, \mathbb{I})$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha}_t}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\boldsymbol{\epsilon}}_0(\mathbf{x}_t)\right) + \sigma_q(t)\boldsymbol{z}$
end for

Denoising Diffusion Implicit Model (DDIM)

A key drawback of DDPMs is their reliance on many iterations to generate high-fidelity samples. As noted by Song et al [16], a DDPM would take more than 1,000 hours to generate 50,000 256×256 images on a standard GPU. This inefficiency arises because the reverse diffusion process requires iterative denoising; if the process intrinsically needs many steps to converge, generation will inherently be slow. Therefore, to accelerate computation, it is necessary to reduce the number of iterations. DDIM was introduced to address this limitation and provide faster image generation.

Probability Distributions in DDIM

We introduced the Markov property for $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ to make computation tractable; however, a key drawback is that a Markov chain often requires many steps to converge. DDIM addresses this issue by departing from the Markovian structure and adopting a non-Markovian formulation:

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}\left(\mathbf{x}_t \mid \sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\mathbf{x}_{t-1}, \left(1 - \frac{\alpha_t}{\alpha_{t-1}}\right)\mathbb{I}\right)$$

Now, $\bar{\alpha}_t := \prod_{i=1}^{t} \frac{\alpha_i}{\alpha_{i-1}} = \alpha_t$ assuming $\alpha_0 = 1$.

This implies: $q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\mathbf{x}_t \mid \sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbb{I}\right)$. By means of reparameterization, we get $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I})$.

The key idea is to rewrite the expression for $\boldsymbol{\epsilon}$ so that $\mathbf{x}_{t-1}$ is no longer directly dependent on $\mathbf{x}_0$, which is perturbed by white noise. Starting from the standard forward diffusion equation:

$$\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}$$

we can rearrange it to isolate $\boldsymbol{\epsilon}$:

$$\boldsymbol{\epsilon} = \frac{\mathbf{x}_t - \sqrt{\alpha_t}\,\mathbf{x}_0}{\sqrt{1-\alpha_t}}$$

Using the standard forward diffusion equation for $\mathbf{x}_{t-1}$ and substituting the expression for $\boldsymbol{\epsilon}$ derived above, we obtain:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1-\alpha_{t-1}}\frac{\mathbf{x}_t - \sqrt{\alpha_t}\,\mathbf{x}_0}{\sqrt{1-\alpha_t}}$$

While $\boldsymbol{\epsilon}$ simplifies DDPMs, the Gaussian noise also slows the reverse diffusion process.

Likely the most important argument in DDIM is that we desire the marginal distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)$ to have the same form as $q(\mathbf{x}_t \mid \mathbf{x}_0)$. The motivation for aiming for this distribution is that, ultimately, we care about the marginal distribution $q(\mathbf{x}_t \mid \mathbf{x}_0)$, which we want to become pure white noise when $t = T$,

and the original image when $t = 0$. Therefore, while we can have millions of different choices for the transition distribution $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$, only a few specialized transition probabilities can ensure that $q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)$ takes a desirable form.

Using equation 2.115 from [17], we get the **DDIM Transition Distribution**:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0 + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\left(\frac{\mathbf{x}_t - \sqrt{\alpha_t}\,\mathbf{x}_0}{\sqrt{1-\alpha_t}}\right), \sigma_t^2\mathbb{I}\right)$$

If $q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_t}\mathbf{x}_0, (1-\alpha_t)\mathbb{I}\right)$, then it follows from our derivation that $q(\mathbf{x}_{t-1} \mid \mathbf{x}_0) = \mathcal{N}\left(\sqrt{\alpha_{t-1}}\mathbf{x}_0, (1-\alpha_{t-1})\mathbb{I}\right)$.

Inference for DDIM

The inference for DDIM is derived based on the transition distribution. Starting with the forward process, if we want to perform the reverse, we need to solve for $\mathbf{x}_0$ from the following equation:

$$\mathbf{x}_t = \sqrt{\alpha_t}\,\mathbf{x}_0 + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}$$

where $\mathbf{x}_t$ is given, $\sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}$ is estimated by the network, and our goal is to find $\sqrt{\alpha_t}\,\mathbf{x}_0$. By estimating the noise $\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t)$, which replaces $\boldsymbol{\epsilon}$. We also define the estimate of the true signal $\mathbf{x}_0$ as:

$$f_\theta^{(t)}(\mathbf{x}_t) := \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t)\right)$$

We recall that this distribution is denoted by $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$. If we don't have access of $\mathbf{x}_0$, we replace it with $f_\theta^{(t)}(\mathbf{x}_t)$. Skipping derivations, we get for DDIM:

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}}\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t)}{\sqrt{\alpha_t}}\right) + \sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\boldsymbol{\epsilon}_\theta^{(t)}(\mathbf{x}_t) + \sigma_t\cdot\boldsymbol{\epsilon}_t,\ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I})$$

The main difference between DDPM and DDIM is subtle. While they both use $\mathbf{x}_t$ and $\boldsymbol{\epsilon}^{(t)}(\mathbf{x}_t)$ in their updates, the specific update formula makes a different convergence speed. In fact, later in the differential equation literature where people connect DDIM and DDPM with stochastic differential equations, it was observed that DDIM employed some special accelerated first-order numerical schemes when solving the differential equation.

3 Score-Matching Langevin Dynamics & Langevin and Fokker-Planck Equations

In this section, we briefly introduce the high-level intuition behind Score-Matching Langevin Dynamics as well as the Langevin and Fokker-Planck equations, which

together form an important foundation for understanding the theory and practice of diffusion-based generative models. We will not go into detailed mathematical derivations or technical proofs for either topic; our focus is on building intuition for their roles and contributions. For readers interested in rigorous exposition and further technical details, we recommend [18] and [19] for Score-Matching Langevin Dynamics (SMLD), and [20] for formal analysis of the Langevin and Fokker-Planck equations.

3.1 SMLD: Principles and Significance

Score-Matching Langevin Dynamics (SMLD) offers a geometric and probabilistic approach to generative modeling that stands in contrast to classic methods such as Variational Autoencoders (VAE) or step-wise denoising methods like DDPM and DDIM. Whereas VAEs rely on explicitly mapping data to and from a latent space, and DDPMs/DDIMs define a fixed sequence of noise addition and removal steps, SMLD focuses on learning a vector field (the score function) which points in the direction of increasing data probability.

Once the score is learned, new samples are synthesized by simulating a stochastic process: starting from noise, we iteratively nudge each sample towards higher probability regions according to the score, while also injecting randomness at each iteration. This process, rooted in Langevin Dynamics (which comes from physics), forces samples to follow the geometry of the data distribution. Unlike the scheduled or latent-variable approaches of VAE and DDPM/DDIM, SMLD provides a more direct, geometry-driven, and potentially more flexible path for generative modeling.

3.2 Langevin and Fokker-Planck Equations in Diffusion

The intuition behind Score-Matching Langevin Dynamics is mathematically grounded in the Langevin and Fokker-Planck equations, originally developed to describe the behavior of particles under thermal fluctuations. The Langevin equation characterizes the stochastic evolution of individual samples influenced by both deterministic drift (guided by the score) and random noise. Its companion, the Fokker-Planck equation, describes how the overall probability distribution of a collection of such samples evolves over time.

From a generative modeling perspective, these equations make clear that synthesizing data can be viewed as a stochastic process: as we apply the learned score field and add noise, initially random samples are gradually transformed so that their distribution converges towards the data distribution. Unlike the explicit latent representations of VAEs or the fixed noise-reversal schedules of DDPM/DDIM, this approach leverages the continuous transformation of distributions under stochastic dynamics, interpreted through the lens of statistical physics.

3.3 Further Reading

The Score-Matching Langevin framework, together with the Langevin and Fokker-Planck formalism, extends generative modeling beyond the boundaries of scheduled, latent-variable, or decoder-based architectures. It provides both practical flexibility and deeper theoretical insight, highlighting generation as a guided random walk driven by the data probability landscape.

For a comprehensive and mathematical account of Score-Matching Langevin Dynamics, readers are referred to [18] and [19]. For the foundational theory of Langevin and Fokker-Planck equations, [20] serves as a canonical reference.

4 Stochastic Differential Equations

Stochastic Differential Equations (SDEs) describe the evolution of systems over time by incorporating random perturbations that simulate diffusion processes. By formulating the denoising process as an SDE, we can leverage techniques from stochastic calculus to derive efficient noise reduction algorithms. SDEs enable modeling of both the forward diffusion process, which gradually adds noise to data, and the reverse process, which aims to recover the original signal. This reverse process can be optimized using methods like Langevin dynamics, facilitating the generation of high-quality samples from the learned distribution.

Wiener Process

A collection $\{\mathbf{X}(t) \mid t \geq 0\}$ of random variables is called a **stochastic process**. For each $\omega \in \Omega$, the mapping $t \mapsto \mathbf{X}(t, \omega)$ represents the corresponding sample path. For fixed t and ω, the value of $\mathbf{X}(t, \omega)$ remains constant. However, since ω is typically unknown, the process exhibits nondeterministic behavior. In computer science, a parameter like ω that converts a nondeterministic function into a deterministic one is known as an oracle input.

From [21]: A real-valued stochastic process $W(\cdot)$ is called a **Wiener process** (or **Brownian motion**) if:

1. $W(0) = 0$ almost surely.
2. $W(t) - W(s) \sim \mathcal{N}(0, t - s)$ for all t, s with $t \geq s \geq 0$.
3. Independent increments: for all $0 < t_1 < t_2 < \cdots < t_n$, the random variables $W(t_1), W(t_2) - W(t_1), \ldots, W(t_n) - W(t_{n-1})$ are independent.

Note in particular that for all $t \geq 0$, $\mathbb{E}[W(t)] = 0$ and $\mathbb{E}[W^2(t)] = t$. It can be easily shown that for all $t, s \geq 0$, $\mathbb{E}[W(t)W(s)] = \min\{s, t\}$.

Heuristics

The expression $\frac{dW(t)}{dt} = \xi(t)$ denotes one-dimensional white noise (a Gaussian process with zero mean and delta autocorrelation). For almost every ω, the

sample path $t \mapsto W(t, \omega)$ is nowhere differentiable for any $t \geq 0$. Thus, $\frac{dW(t)}{dt} = \xi(t)$ does not exist in the classical sense, but with δ_0 (the Dirac delta function centered at 0), we have the following heuristic formula from [21]:

$$\mathbb{E}[\xi(t)\xi(s)] = \delta_0(s - t).$$

Stochastic Integral

For a simple process:

$$H(t) = \sum_{i=0}^{n-1} H_i \cdot \mathbf{1}_{[t_i, t_{i+1})}(t)$$

where each H_i is $\mathcal{F}_i = \sigma\{W_s \mid s \leq t_i\}$-measurable and $\{t_i\}$ forms a partition of $[0, T]$, the Itô integral is defined as:

$$\int_0^T H(t)\, dW_t = \lim_{n \to \infty} \sum_{i=0}^{n-1} H(t_i)(W_{t_{i+1}} - W_{t_i})$$

For general $H \in L^2([0, T] \times \Omega)$, the integral $\int_0^T H(t) dW_t$ is defined as the L^2 limit of such simple processes. Crucially, the integrand is evaluated at $H(t_i)$ rather than at intermediate points (the latter approach would yield the Stratonovich integral).

SDE

In an SDE, in addition to a deterministic function $f(\mathbf{x}, t)$, we consider a stochastic perturbation. For example, the stochastic perturbation can take the following form:

$$\frac{dx(t)}{dt} = f(\mathbf{x}, t) + g(\mathbf{x}, t)\xi(t), \quad \text{where} \quad \xi(t) \sim \mathcal{N}(0, \mathbb{I})$$

where $\xi(t)$ is a noise function, e.g., white noise. We can define $dW_t = \xi(t)dt$, where dw is often known as the differential form of the Wiener process. Then, the differential form of this SDE can be written as

$$dx = f(\mathbf{x}, t)\, dt + g(\mathbf{x}, t)\, dW_t$$

Because $\xi(t)$ is random, the solution to this differential equation is also random. To be explicit about the randomness of the solution, we should interpret the differential form via the integral equation

$$x(t, \omega) = x_0 + \int_0^t f(\mathbf{x}(s, \omega), s)\, ds + \int_0^t g(\mathbf{x}(s, \omega), s)\, dW(s, \omega),$$

where ω denotes the index of the state of x. Therefore, as we pick a particular state of the random process $w(s, \omega)$, we solve a differential equation corresponding to this particular ω.

The Bridge to DDPM

The forward diffusion process can be written as:

$$\frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}, t) + g(t)\xi(t), \quad \xi(t) \sim \mathcal{N}(0, \mathbb{I}).$$

This process is driven by a Wiener process, where $\xi(t)$ is assumed to be i.i.d. Gaussian noise for all t. The autocorrelation function is given by the Dirac delta:

$$\mathbb{E}[\xi(t)\xi(s)] = \delta_0(t - s), \quad t \neq s.$$

Using the notation $dW_t = \xi(t)dt$, where $f(\mathbf{x}, t)$ is the **drift term** and $g(t)$ the **diffusion term**, we express the forward diffusion in standard SDE form:

$$d\mathbf{x} = f(\mathbf{x}, t)\, dt + g(t)\, dW_t.$$

- The **drift coefficient** $f(\mathbf{x}, t)$ is a vector-valued function that determines the system's deterministic dynamics, specifying how the state $\mathbf{x}$ would evolve in the absence of noise.
- The **diffusion coefficient** $g(t)$ is a scalar function that scales the system's response to random perturbations, controlling the magnitude of noise affecting the trajectory.

From [22], the reverse-time SDE is:

$$d\mathbf{x} = \left[f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + g(t) d\overline{W}_t,$$

where:

- $f(\mathbf{x}, t)$ is the drift term.
- $p_t(\mathbf{x})$ is the probability density of $\mathbf{x}$ at time t.
- $\overline{W}_t$ is a Wiener process under time reversal.
- $g(t)d\overline{W}_t$ represents the reverse-time diffusion term.

SDE for DDPM

From [12] (page 59): The forward sampling equation of DDPM can be written as an SDE:

$$d\mathbf{x} = -\frac{\beta(t)}{2}\mathbf{x}\, dt + \sqrt{\beta(t)}\, dW_t$$

This representation means DDPM estimates can be determined by solving the SDE. For an appropriately defined SDE solver, we could numerically solve this equation. While the DDPM iteration itself provides a solution (being a first-order SDE solver), alternative solvers may offer improved accuracy.

The reverse sampling equation of DDPM is given by:

$$d\mathbf{x} = -\beta(t)\left[\frac{\mathbf{x}}{2} + \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] dt + \sqrt{\beta(t)}\, d\overline{W}_t$$

Predictor-Corrector Algorithm

Since many ODEs and SDEs cannot be solved analytically, we must rely on numerical solutions. The most commonly used numerical solvers are the Euler method and Runge-Kutta method, though we will also cover the Predictor-Corrector algorithm. Different numerical solvers exhibit varying approximation errors, meaning that using standard off-the-shelf solvers will yield solutions with different levels of accuracy.

Below is the prediction-correction algorithm for DDPM from [23], where $\boldsymbol{s}_\theta(\mathbf{x}) := \nabla_x \log p_\theta(\mathbf{x})$ is Stein's score function from SMLD.

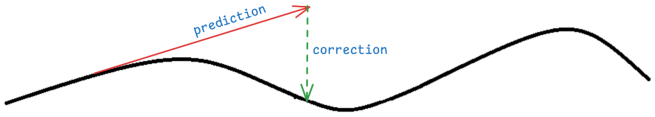


Figure 3: Prediction-Correction visualized

Algorithm 6 Prediction Correction Algorithm for DDPM [23]

$\mathbf{x}_N \sim \mathcal{N}(0, \mathbb{I})$
for $i = N, N-1, \ldots, 0$ **do**
  $\boldsymbol{z}_i \sim \mathcal{N}(0, \mathbb{I})$
  Prediction: $\mathbf{x}_{i-1} = \frac{1}{\sqrt{1-\beta_i}}\left[\mathbf{x}_i + \frac{\beta_i}{2}\boldsymbol{s}_\theta(\mathbf{x}_i, i)\right] + \sqrt{\beta_i}\boldsymbol{z}_i$
  for $m = 1, \ldots, N$ **do**
    Correction: $\mathbf{x}_{i-1} = \mathbf{x}_i + \epsilon_i \boldsymbol{s}_\theta(\mathbf{x}_i, i) + \sqrt{2\epsilon_i}\boldsymbol{z}_i$ with $\boldsymbol{z}_i, \epsilon_i \sim \mathcal{N}(0, \mathbb{I})$
  end for
end for

5 Causal Diffusion Autoencoders

This section is based on the work of Komanduri et al. [2]. The authors aim to model causal relations among semantic latent codes in order to learn causal representations and enable counterfactual generation in diffusion probabilistic models (DPMs). Causal reasoning about hypothetical scenarios provides insights into the interactions between causal variables in complex systems. To this end, the authors propose **CausalDiffAE**, a framework for causal representation learning and controllable counterfactual generation in DPMs.

The main idea is to learn a causal representation using a stochastic encoder, and to model the relationships among latent variables by parameterizing causal mechanisms with neural networks. A variational objective with a label alignment prior is formulated to encourage disentanglement of the learned factors. Decoding and stochastic variation modeling are achieved using a conditional DDIM. This approach yields a compact, causally-relevant latent representation

for reverse diffusion image synthesis. By explicitly modeling causal relations in the latent space, counterfactual samples can be generated via interventions $\boldsymbol{do}(\cdot)$ on the learned causal variables. The authors further propose a DDIM variant for counterfactual generation under such interventions. The architecture is shown in Figure 4.

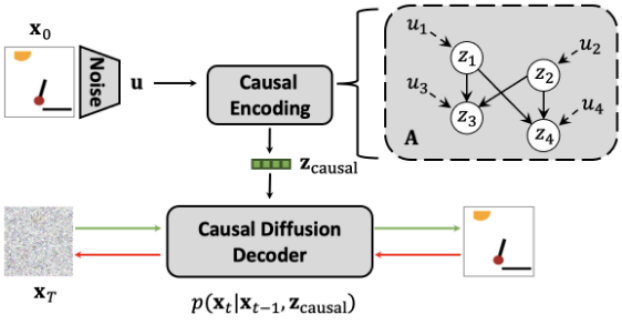


Figure 4: CausalDiffAE Architecture

Causal Encoding

Given an observed image $\mathbf{x}_0 \in \mathbb{R}^d$, forward diffusion is applied to create noisy samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$. It is assumed that $n \in \mathbb{N}$ abstract causal variables summarize high-level semantics. The encoder maps $\mathbf{x}_0$ to exogenous noise $\mathbf{u} \in \mathbb{R}^n$, which is then transformed into causal latents $\mathbf{z}_{\text{causal}} \in \mathbb{R}^n$. Each u_i acts as noise for z_i in an SCM $\mathfrak{C}$, with dependencies structured by an adjacency matrix A:

$$z_i = f_i(z_{\mathbf{PA}_i}, u_i)$$

where $z_{\mathbf{PA}_i}$ are the parents of z_i. Typically, post-nonlinear additive mechanisms are used, e.g. $z_i = f_i(A_i \circ z; \nu_i) + u_i$ with $z \in z_{\mathbf{PA}_i}$ and $\nu_i \sim \mathcal{N}(0, \mathbb{I})$, ensuring that latent variables reflect causal semantics.

Generative Model

Forward diffusion is defined by:

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

using a noise schedule β_t. The generative process decomposes as:

$$p(\mathbf{x}_{0:T}, \mathbf{u}, \mathbf{z}_{\text{causal}} \mid y) = p_\theta(\mathbf{x}_{0:T} \mid \mathbf{u}, \mathbf{z}_{\text{causal}})\, p(\mathbf{u}, \mathbf{z}_{\text{causal}} \mid y)$$

with $p(\mathbf{u}) = \mathcal{N}(0, \mathbb{I})$ and $p(\mathbf{z}_{\text{causal}} \mid y)$ serving to align latent variables to labels. θ are the parameters of the reverse process of the conditional diffusion model. Maximizing model evidence involves:

$$\log p(\mathbf{x}_0, y) = \log \int p(\mathbf{x}_{0:T}, \mathbf{u}, \mathbf{z}_{\text{causal}}, y)\, d\mathbf{x}_{1:T}\, d\mathbf{u}\, d\mathbf{z}_{\text{causal}},$$

where an intractable posterior is approximated by:

$$q(\mathbf{x}_{1:T}, \mathbf{u}, \mathbf{z}_{\text{causal}} \mid \mathbf{x}_0, y) = q_\phi(\mathbf{z}_{\text{causal}}, \mathbf{u} \mid \mathbf{x}_0, y)\, q(\mathbf{x}_{1:T} \mid \mathbf{u}, \mathbf{z}_{\text{causal}}, \mathbf{x}_0)$$

with variational encoder parameters ϕ.

Causal Diffusion Decoder

The conditional DDIM decoder reverses diffusion from $(\mathbf{z}_{\text{causal}}, \mathbf{x}_T)$, with the goal of approximating $q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0)$ by $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{z}_{\text{causal}})$:

$$p_\theta(\mathbf{x}_{0:T} \mid \mathbf{z}_{\text{causal}}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{z}_{\text{causal}})$$

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{z}_{\text{causal}}) = \mathcal{N}(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \epsilon_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{causal}})),$$

where ϵ_θ is a noise prediction UNet. Training minimizes:

$$\mathcal{L}_{\text{simple}} = \sum_{t=1}^{T} \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \|\epsilon_\theta(\mathbf{x}_t, t, \mathbf{z}_{\text{causal}}) - \epsilon_t\|_2^2$$

where $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\,\epsilon_t$, and $\epsilon_t \sim \mathcal{N}(0, \mathbb{I})$.

Learning Objective

Disentanglement and semantic alignment are encouraged by the following loss:

$$\begin{aligned}\mathcal{L}_{\text{CausalDiffAE}} = \mathcal{L}_{\text{simple}} + \gamma \Big[& \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{z}_{\text{causal}} \mid \mathbf{x}_0, y) \,\|\, p(\mathbf{z}_{\text{causal}} \mid y)) \\ & + \mathbb{D}_{\text{KL}}(q_\phi(\mathbf{u} \mid \mathbf{x}_0) \,\|\, \mathcal{N}(0, \mathbb{I})) \Big]\end{aligned}$$

where γ is a regularization hyperparameter similar to the bottleneck parameter in β-VAEs and with a prior:

$$p(\mathbf{z}_{\text{causal}} \mid y) = \prod_{i=1}^{n} \mathcal{N}(z_i; \mu_\nu(y_i), \sigma_\nu^2(y_i)\mathbb{I}),$$

where μ_ν and σ_ν^2 are the learned mean and variance of the Gaussian, ensuring alignment between the latent variables and the corresponding label factors.

Since **CausalDiffAE** is formulated as a variational model with a stochastic encoder, representations can be sampled directly from the defined prior. This eliminates the need to train a separate diffusion model in the latent space, allowing the following algorithm to be applied:

Algorithm 7 CausalDiffAE Training

Input: (image, label) pairs $(\mathbf{x}_0, y)$
Output: Trained causal diffusion autoencoder ϵ_θ
repeat
  Sample image $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
  Noise Encoding: sample $\mathbf{u} \sim q_\phi(\mathbf{u} \mid \mathbf{x}_0)$
  Causal Encoding: $\mathbf{z}_{\text{causal}} = \{f_i(u_i, z_{\mathbf{PA}_i}; \nu_i)\}_{i=1}^n$
  Sample timestep: $t \sim \text{Uniform}[1, T]$
  Sample noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{I})$
  Corrupt data to sampled time: $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}$
  Compute and take gradient step on $\nabla_\theta \mathcal{L}_{\text{CausalDiffAE}}$
until convergence

Algorithm 8 CausalDiffAE Counterfactual Generation

Input: Factual sample $\mathbf{x}_0$; intervention target set $\mathcal{I}$ with intervention values c; pre-trained causal diffusion autoencoder ϵ_θ
Output: Counterfactual sample $\mathbf{x}_0^{\text{CF}}$
Sample exogenous noise: $\mathbf{u} \sim q_\phi(\mathbf{u} \mid \mathbf{x}_0)$
for $i = 1$ to n **(in topological order) do**
  if $i \in \mathcal{I}$ **then**
    $z_i := c_i$
  else
    $z_i := f_i(u_i, z_{\mathbf{PA}_i})$
  end if
end for
Set intervened latent: $\bar{\mathbf{z}}_{\text{causal}} = \{z_1, \ldots, z_n\}$
Sample $\mathbf{x}_T \sim \mathcal{N}(\sqrt{\alpha_T}\mathbf{x}_0, (1-\alpha_T)\mathbb{I})$
$\mathbf{x}_T^{\text{CF}} := \mathbf{x}_T$
for $t = T, \ldots, 1$ **do**

$$\mathbf{x}_{t-1}^{\text{CF}} := \sqrt{\bar{\alpha}_{t-1}}\left(\frac{\mathbf{x}_t^{\text{CF}} - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(\mathbf{x}_t^{\text{CF}}, t, \mathbf{z}_{\text{causal}})}{\sqrt{\bar{\alpha}_t}}\right)$$
$$= \quad + \sqrt{1-\bar{\alpha}_{t-1}}\,\epsilon_\theta(\mathbf{x}_t^{\text{CF}}, t, \bar{\mathbf{z}}_{\text{causal}})$$

end for
return $\mathbf{x}_0^{\text{CF}}$

Counterfactual Generation

Interventions on any z_i via $\boldsymbol{do}(\cdot)$ can be propagated through the neural SCM, allowing counterfactual samples to be generated by the DDIM decoder. The authors utilize the DDIM sampling algorithm to ensure the stochastic noise x_T is a deterministic encoding to enable semantic manipulations. Changes in

$\mathbf{z}_{\text{causal}}$ yield controllable counterfactuals reflecting the model's causal semantics. To generate counterfactuals from a trained **CausalDiffAE**, where $\mathbf{x}_0$ refers to the factual observation and $\mathbf{x}_0^{\text{CF}}$ refers to the generated counterfactual sample, use the algorithm 8.

6 Experiments

The code referenced in [2] is accessible at the GitHub repository: `https://github.com/Akomand/CausalDiffAE.git`. However, this code contained numerous bugs, and the execution of the examples provided in the README.md file was unclear. Consequently, the author of this paper forked the repository, rectified the bugs, and updated the README.md file to include not only instructions on how to run the code but also a comprehensive step-by-step guide for transforming the current version of the code (commit 4e9f669) into functional code. It is important to note that this corrected code has been tested and confirmed to work for the MorphoMNIST example; however, it has not been tested and is not expected to function with the pendulum or CausalCircuit examples. The new fork can be found at [25].

6.1 Metrics

The DCI framework is a quantitative approach to assess the quality of learned representations in terms of disentanglement and completeness.

Disentanglement

The disentanglement score measures how well each latent variable captures an individual generative factor.

Let $\mathbf{R}_{ij}$ denote the importance of latent variable z_i for predicting generative factor y_j. The importance can be estimated using supervised predictors such as gradient boosting regressors. The probability that z_i is a strong predictor of y_j is

$$P_{ij} = \frac{\mathbf{R}_{ij}}{\sum_{k=1}^{K} \mathbf{R}_{ik}}$$

where K denotes the number of generative factors. The disentanglement score for latent code z_i is:

$$D_i = 1 + \sum_{k=1}^{K} P_{ik} \log_K P_{ik}$$

and the total disentanglement score is a weighted average:

$$D = \sum_{i=1}^{d} \rho_i D_i$$

where $\rho_i = \frac{\sum_j \mathbf{R}_{ij}}{\sum_{i,j} \mathbf{R}_{ij}}$ and d is the dimension of the latent representation.

Completeness

Completeness quantifies to what extent each generative factor is predominantly explained by a single latent variable.

The completeness score per generative factor y_j is given by

$$C_j = 1 + \sum_{k=1}^{d} Q_{kj} \log_d Q_{kj} \text{ where } Q_{kj} = \frac{\mathbf{R}_{kj}}{\sum_{i=1}^{d} \mathbf{R}_{ij}}$$

and the total completeness score is the weighted sum:

$$C = \sum_{j=1}^{K} \phi_j C_j \text{ with } \phi_j = \frac{\sum_i \mathbf{R}_{ij}}{\sum_{i,j} \mathbf{R}_{ij}}$$

6.2 Interventions

Interventions will be performed in the following way: let $x \in \mathbb{R}^{C \times H \times W}$ be an input image, and let the encoder E of the trained model produce Gaussian parameters $(\mu, \sigma^2) = E(x)$. The latent representation μ is partitioned as $\mu = \left[\mu^{(1)}, \mu^{(2)}\right], \mu^{(i)} \in \mathbb{R}^d$, ($d = 256$ in this case), where $\mu^{(1)}$ and $\mu^{(2)}$ correspond to disentangled generative factors (thickness, intensity).

For thickness intervention, the code sets $\mu^{(1)} := 0.2 \cdot \mathbf{1}$, leaving $\mu^{(2)}$ unchanged.

For intensity intervention, the code instead modifies the post-masking vector z_{post} by setting $z^{(2)}_{\text{post}} := 0.2 \cdot \mathbf{1}$.

Both operations employ the causal masking $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ (feature 0 represents thickness, while feature 1 represents intensity. We expect changes in thickness to affect intensity, but not the other way around) to produce:

$$z_{\text{pre}} = A^{\intercal}\mu, z_{\text{post}} = f_{\text{NL}}(z_{\text{pre}}) + \mu$$

where f_{NL} is a learned non-linearity. After transpose, for thickness T and intensity I, we get:

$$z^{(1)}_{\text{pre}} = 0 \cdot T + 0 \cdot I = 0 \text{ (no parents for thickness)}$$

$$z^{(2)}_{\text{pre}} = 1 \cdot T + 0 \cdot I = T \text{ (thickness as parent for intensity)}$$

The modified latent variables z are obtained by reparameterization:

$$z = z_{\text{post}} + \sigma \cdot \epsilon, \ \epsilon \sim \mathcal{N}(0, \mathbb{I})$$

To generate the intervened image, the forward diffusion process produces a noisy state $\mathbf{x}_t$ at timestep $t = 249$, $\mathbf{x}_t = q_t(\mathbf{x}, \epsilon), \epsilon \sim \mathcal{N}(0, \mathbb{I})$, and the reverse process uses a sampling operator $\mathcal{S} * w$ (either $p * \theta$ or DDIM):

$$\hat{\mathbf{x}} * 0 = \mathcal{S} * w(\mathbf{x}_t \mid z, y)$$

where the scalar parameter w controls classifier-free guidance by forming a convex combination of unconditional and conditional model predictions:

$$\epsilon_{\text{guided}} = w \cdot \epsilon_{\text{cond}} + (1 - w) \cdot \epsilon_{\text{uncond}}$$

Thus, $w > 0$ amplifies the influence of the conditional signal (the intervened latent factor), sharpening the causal effect in the output image.

Checking Causal Structure

1. **Norm of Thickness z_{pre}**: The norm of thickness z_{pre} should be close to 0, as thickness has no parents in the causal structure.
2. **Similarity Between Intensity and Thickness**: There should be a high similarity between the intensity z_{pre} and the original thickness, since intensity receives thickness as input.
3. **Effects of Interventions**: In interventions, changes to thickness should visibly affect intensity in the generated images, while changes to intensity should not affect thickness.

6.3 Results

We anticipate that the intervention on thickness T will influence intensity I, but not vice versa:

$$T = N_T$$
$$I = f(T) + N_I$$

In this formulation, N_T and N_I represent noise variables, while f denotes a learned function that relates thickness to intensity.

Note: Results below are from the model trained for 210,400 steps!

DCI Scores

The model achieved a disentanglement score of ≈ 0.99 and a completeness score of ≈ 0.53.

The disentanglement score of ≈ 0.99 shows that the model has almost perfectly separated the latent variables so that each one corresponds to a single generative factor. This means the representation is highly interpretable, with minimal mixing between factors of variation.

The completeness score of ≈ 0.53, however, reveals that while each latent cleanly maps to one factor, the reverse is less true - information about a single factor is distributed across several latents. In our case, the dependency from thickness to intensity likely contributes to this, as correlated factors are harder to isolate into individual latents.

Overall, the model is excellent at factor separation but only moderately effective at compactly capturing each factor, leaving room to improve completeness despite the very high disentanglement.

Examples

The model has been run for three values of $w \in \{0, \frac{1}{2}, 1\}$. Due to the sampling technique, the original images on which we intervene differ for all three values of w:

References

[1] Jonas Peters, Dominik Janzing, Bernhard Schölkopf. *Elements of Causal Inference*. The MIT Press, 2017.

[2] Aneesh Komanduri, Chen Zhao, Feng Chen, and Xintao Wu. *Causal Diffusion Autoencoders: Toward Counterfactual Generation via Diffusion Probabilistic Models*. In Proceedings of the 27th European Conference on Artificial Intelligence, 2024. Available at: `https://github.com/Akomand/CausalDiffAE.git`.

[3] Qi Liu, Yuanqi Du, Fan Feng, Qiwei Ye, Jie Fu. *Structural Causal Model for Molecular Dynamics Simulation*. City University of Hong Kong, Cornell University, Beijing Academy of Artificial Intelligence, 2023.

[4] Peter Spirtes, Clark Glymour. *An Algorithm for Fast Recovery of Sparse Causal Graphs*. Social Science Computer Review, Volume 9, Issue 1, 1991.

[5] Bryon Aragam. *Greedy Equivalence Search for Nonparametric Graphical Models*. 2024.

[6] Peter Spirtes. *An Anytime Algorithm for Causal Inference*. 2001.

[7] Anne Helby Petersen. *Introduction to causal discovery: CPDAGs and the PC algorithm*. University of Copenhagen, Section of Biostatistics, EuroCIM 2024.

[8] Christopher Meek. *Causal Inference and Causal Explanation with Background Knowledge*. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, UAI'95. 1995.

[9] Rajen D. Shah, Jonas Peters. *The hardness of conditional independence testing and the generalised covariance measure*. The Annals of Statistics. 2020.

[10] Anne Helby Petersen, Merete Osler , Claus T. Ekstrøm. *Data-driven model building for life-course epidemiology*. American Journal of Epidemiology. 2021.

[11] Jithendara Subramanian, Yashas Anandani, Ivaxi Sheth, Rosemary Ke, Tristan Deleu, Stefan Bauer, Derek Nowrouzezahrai, Samira Ebrahimi Kahou. *Learning Latent Structural Causal Models*. Mila, McGill University, KTH Stockholm, ETS Montréal. 2022.

[12] Stanley Chan. *Tutorial on Diffusion Models for Imaging and Vision*. 2025.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. *Denoising diffusion probabilistic models*. Advances in Neural Information Processing Systems (NeurIPS). 2020.

[14] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. *Deep unsupervised learning using nonequilibrium thermodynamics*. Proceedings of International Conference on Machine Learning (ICML), volume 37, pages 2256–2265. 2015.

[15] Diederik P. Kingma and Max Welling. *Auto-encoding variational Bayes*. International Conference on Learning Representations (ICLR). 2014.

[16] Jiaming Song, Chenlin Meng, and Stefano Ermon. *Denoising diffusion implicit models*. International Conference on Learning Representations (ICLR). 2023.

[17] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer. 2006.

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

[19] Yang Song and Stefano Ermon. *Generative modeling by estimating gradients of the data distribution*. Advances in Neural Information Processing Systems (NeurIPS). 2019.

[20] Hannes Risken. *The Fokker-Planck Equations: Methods of solutions and applications*. Springer, 2 edition. 1989.

[21] Lasha Ephremidze. *Stochastic Differential Equations*. Kutaisi International University. 2025.

[22] Brian Anderson. *Reverse-time diffusion equation models*. 1982.

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. *Score-based generative modeling through stochastic differential equations*. International Conference on Learning Representations (ICLR). 2021.

[24] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. *beta-VAE: Learning basic visual concepts with a constrained variational framework*. International Conference on Learning Representations. 2017.

[25] Stochastic-Batman. *CausalDiffAE_FZJ*. GitHub repository. Available at: `https://github.com/Stochastic-Batman/CausalDiffAE_FZJ.git`.