Conditional Generation & Editing

Conditional Generation

Conditioning introduces external control into the generation process.

It is the technique that makes generative models really useful.

Conditions may be very specific, allowing to generate images with peculiar, possibly bizarre content.

Conditions

Eternal conditions allow to parametrize the generation process towards specific objectives.

Conditions can be of different kinds:

- a **label**: generate a *given* mnist digit
- an **attribute**: generate the face of an *old* man *wearing glasses*
- a set of **keywords**: *red*, *sportiv car*
- a **textual prompt**: "a picture of a lepoard riding a bike"
- ...

Two main approaches

- **training time**: Integrate the condition inside the generative model as an additional input, and train (or fine tune) the model to generate outputs given that condition.

- **sampling time**: use the condition as a mechanism to influence the generation process, without retraining the model.

Do you trust the generative capacities of your model?

Training-Time Conditioning (a.k.a. explicit conditioning)

You provide the conditioning signal (e.g., class label, prompt, image, etc.) as part of the input during training, and the model learns to generate outputs given that condition.

Examples:

- Class-conditional GANs
- Text-to-image diffusion models (e.g., Stable Diffusion)
- LoRa-like and others light-adapters.

Advantages:

- Model learns to bind generation tightly to the condition
- More robust generalization if the condition is well-represented

Limitations:

- Need to retrain the model on the specific conditions
- Conditioning must be included in the dataset structure

Sampling-Time Conditioning (a.k.a. guidance)

You guide the output only during sampling, acting on the latent representation, or modifying the generation process to steer the output toward desired condition.

Examples:

- Classifier-free guidance in diffusion models
- Latent space trajectories
- DDIM inversion + editing

Advantages:

- No need to retrain the model
- Easy to plug into pretrained systems

Limitations:

- Weaker binding to the condition
- Less control or more trial-and-error

Training Time conditioning

Two issues

Instead of learning $P(X|z)$ we need to learn $P(X|z, c)$, where c is the condition.

Two main issues:

- Integrate the condition inside the generative model
- Concrete handling of the condition
  (mixing input and condition)

Conditional VAE (CVAE)

Both the decoder $Q(z|X)$ and the decoder $P(X|x)$ are now parametrized w.r.t. a given condition c: $Q(z|X, c)$ and $P(X|z, c)$.

What about the prior?

- We can still work with a single, condition independent prior (e.g. a normal gaussian)
  $\Rightarrow$ simpler, a little more burden on the decoder side
- We can also use a different - possibly learned - prior (e.g. a different Gaussian) for each condition
  $\Rightarrow$ slightly more complex; not clearly beneficial

Conditional VAE (CVAE)

Both the decoder $Q(z|X)$ and the decoder $P(X|x)$ are now parametrized w.r.t. a given condition c: $Q(z|X,c)$ and $P(X|z,c)$.

What about the prior?

- We can still work with a single, condition independent prior (e.g. a normal gaussian)
  $\Rightarrow$ simpler, a little more burden on the decoder side
- We can also use a different - possibly learned - prior (e.g. a different Gaussian) for each condition
  $\Rightarrow$ slightly more complex; not clearly beneficial

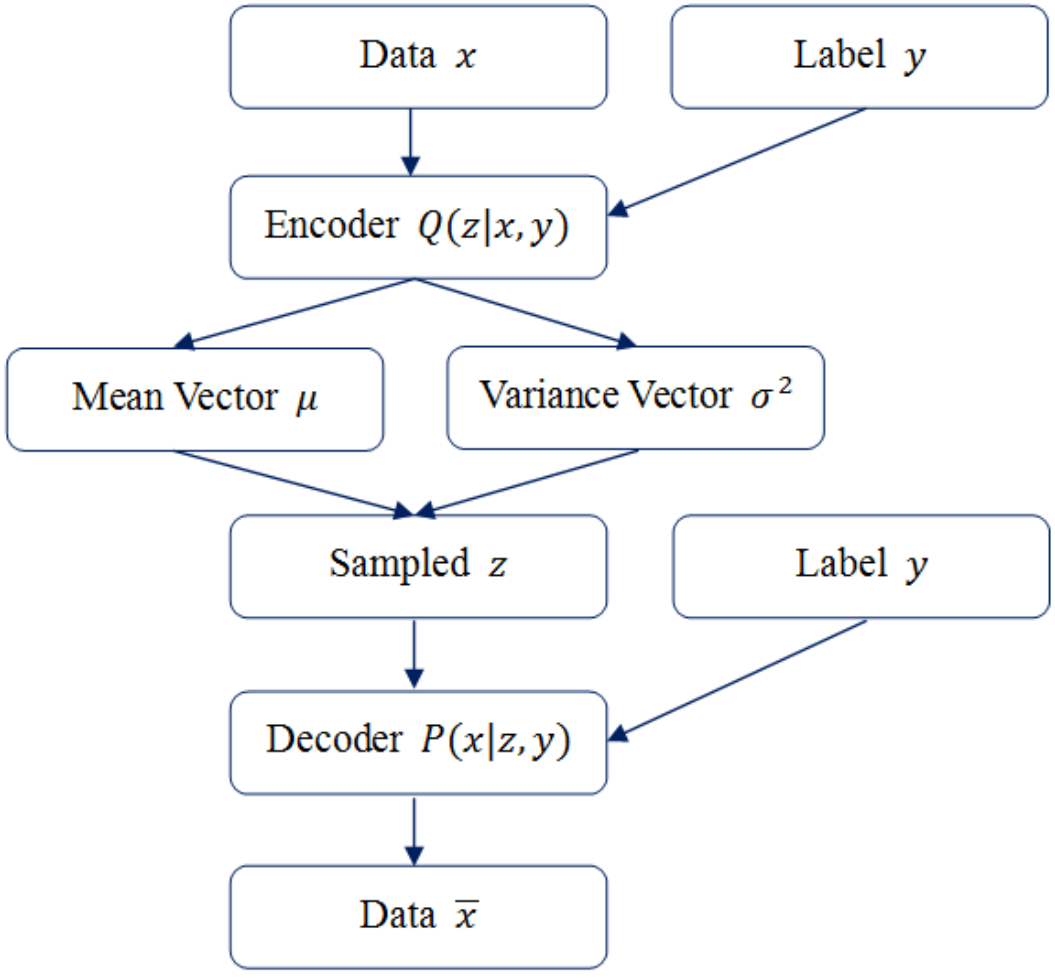
Conditional VAE (CVAE)

Both the decoder $Q(z|X)$ and the decoder $P(X|x)$ are now parametrized w.r.t. a given condition c: $Q(z|X,c)$ and $P(X|z,c)$.

What about the prior?

- We can still work with a single, condition independent prior (e.g. a normal gaussian)
  $\Rightarrow$ simpler, a little more burden on the decoder side
- We can also use a different - possibly learned - prior (e.g. a different Gaussian) for each condition
  $\Rightarrow$ slightly more complex; not clearly beneficial

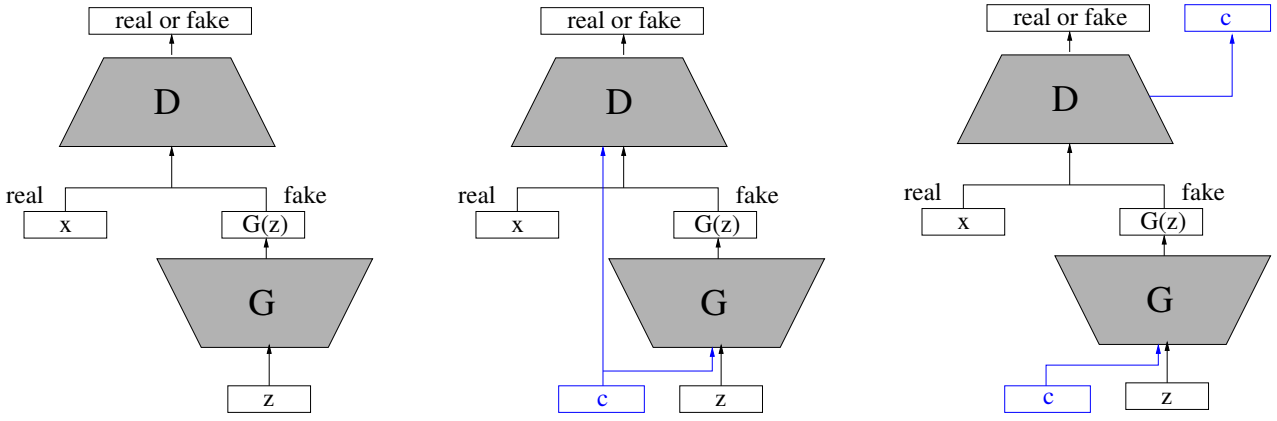
CVAE architecture



Conditional GANs

The generator takes in input the condition, in addition to the noise.

What about the discriminator?



- use the condition to discriminate fakes for real of the given class (Conditional GAN)
- try to classify w.r.t different conditions in addition to true/fake discrimination (Auxiliary Classifier GAN)

AC-GAN loss function

Notation:

- $p(x, c)$ is **true** image-condition joint distribution
- $p_\theta(x, c)$ is the joint distibution of **generated** data
- $q_\theta(c|x)$ is the **classifier**

In addition to the usual GAN objective, we also try to minimize the following quantities:

$$\underbrace{-\mathop{\mathbb{E}}_{p(x,c)} ln(q_\theta(c|x))}_{term\ 1} \underbrace{-\mathop{\mathbb{E}}_{p_\theta(x,c)} ln(q_\theta(c|x))}_{term\ 2}$$

term 1: the classifier should be consistent with the real distribution
term 2: the generator must create images easy to classify

AC-GAN vs InfoGan

AC-Gans are closely related to InfoGan.
In InfoGan, we only have the first term:

$$\underbrace{-\mathop{\mathbb{E}}_{p(x,c)} ln(q_\theta(c|x))}_{term\ 1} \underbrace{-\mathop{\mathbb{E}}_{p_\theta(x,c)} ln(q_\theta(c|x))}_{term\ 2}$$

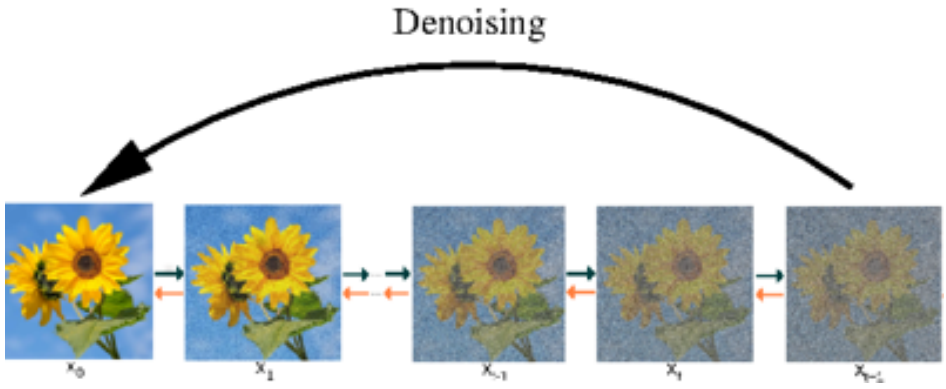
The second term helps to generate images far from boundaries between classes, hence, likely more sharp.

But what if real images **are** close to boundaries?
Suggested reading: AC-GAN learns a biased distribution

Explicit conditioning in diffusion models

Include the condition c as input to the denosing network.

The training objective consists in guessing the noise $\epsilon_\theta(x, t, c)$, where c is the condition (e.g. text embedding, labels, images). A prori, the knowledge of c could even simplify the denoising operation.



$\Longleftarrow$ *A sunflower with a blue sky on the background*

For textual prompts, one typically use LLM embeddings, e.g.:

- DALL·E 3: GPT embeddings
- Stable Diffusion: CLIP embeddings

Architectural handling of the condition

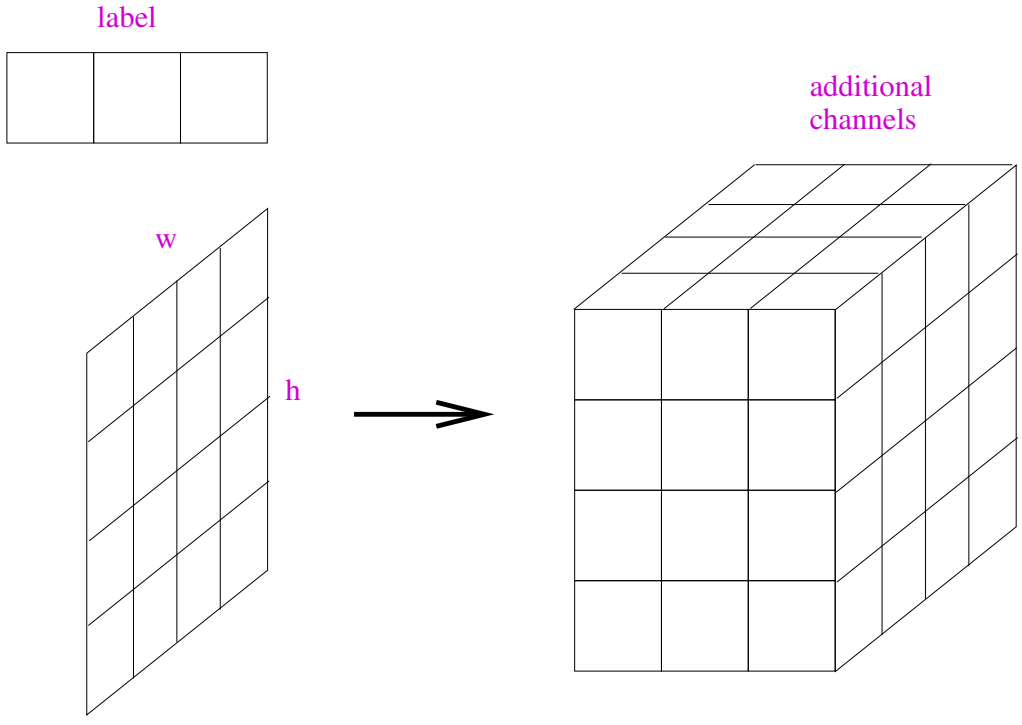
handling the condition

In conditional networks, we pass the label/condition as an additional input. This is typically a vector with a single dimension.

How is this input going to be processed?

- **Concatenation**, possibly after reshaping of vector dimensions, e.g. though **vectorization**
- **Channel modulation**, e.g. through Feature-wise Linear Modulation (FILM) or Adaptive Instance Normalization.
- **Adapters**, like e.g. Low-Rank Adaptation of Large Language Models (LoRA).

vectorization



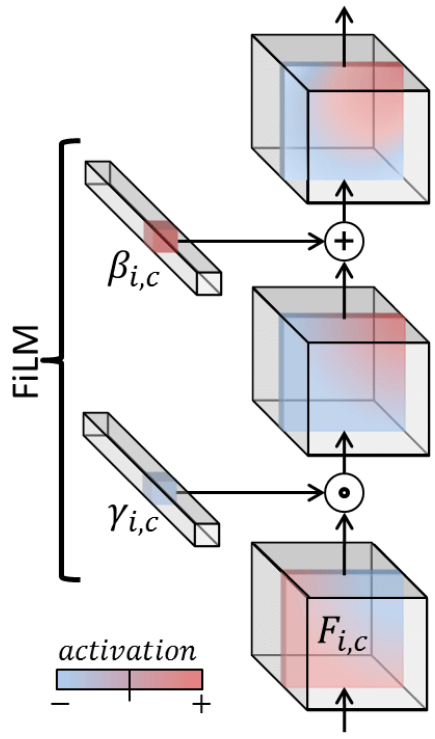
repeat the label (typically in categorical form) for every input neuron, and stack them as new channels

Feature-wise Linear Modulation

Idea: use the condition to give a different weight to each feature (each channel)

Use the condition to generate two vectors γ and β with size equal to the channels of the layer.

rescale layers by γ and add β



Less invasive than parametrizing the weights.

AdaIN: Adaptive Instance Normalization

Adaptive Instance Normalization is a normalization method that aligns the mean and variance of the content features with those of a different sample (style features).

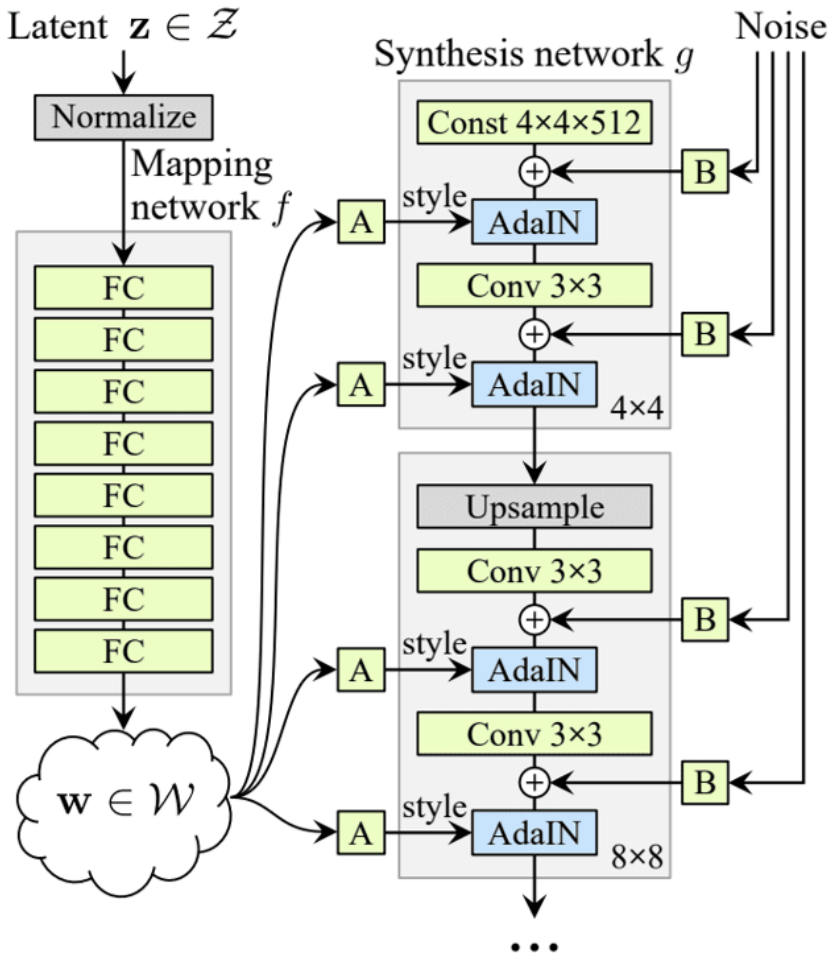
In AdaIN, we receive a "content" input x and a "style" input y and align the channel-wise mean and variance of x to match those of y:

$$AdaIN(x, y) = \sigma(y) + \frac{x - \mu(x)}{\sigma(x)} + \mu(y)$$

where the shape of x is (B, C, W, H) and the shape of $\mu(x), \sigma(x), \mu(y), \sigma(y)$ is (B, C) (intance normalization!)

It essentially differ from FiLM for the fact of performing Instance Nornalization before the modulation.

Example: Use of AdaIN in StyleGan



Lightweight Adapters (e.g., LoRA)

Only tiny trainable modules are inserted in selected sub-layers.

A×B decomposition: express a matrix with dimension $N \times N$ as the product of two matrices of dimension $N \times m$ and $m \times N$, with m small.

Where LoRA is applied: In attention projections (e.g., Q, V matrices)

So inside the attention layer:

$$Q = (W_q + \Delta W_q) \cdot x$$

↑ ↖ LoRA A× B (learnable)

└ Base model (frozen)

Other layers remain frozen, so we save memory, preserve general knowledge, and only learn task-specific deltas.

Benefits of Lightweight Adapters

- Parameter-efficient: Train $< 1\%$ of total model
- Faster training: Works on consumer GPUs
- Modular: One base model, many adapters
- Open source friendly: you can freely share the adapter's weights

Adapters make LLMs customizable and efficient. You can fine-tune a 7B model on a laptop, and keep the base model intact for other tasks.

Example Use Cases

- Alpaca-LoRA (fine-tuned LLaMA using LoRA)
- Personal assistants (train a private adapter on your data)
- Multitask inference (swap adapters at runtime)

Sampling time conditioning

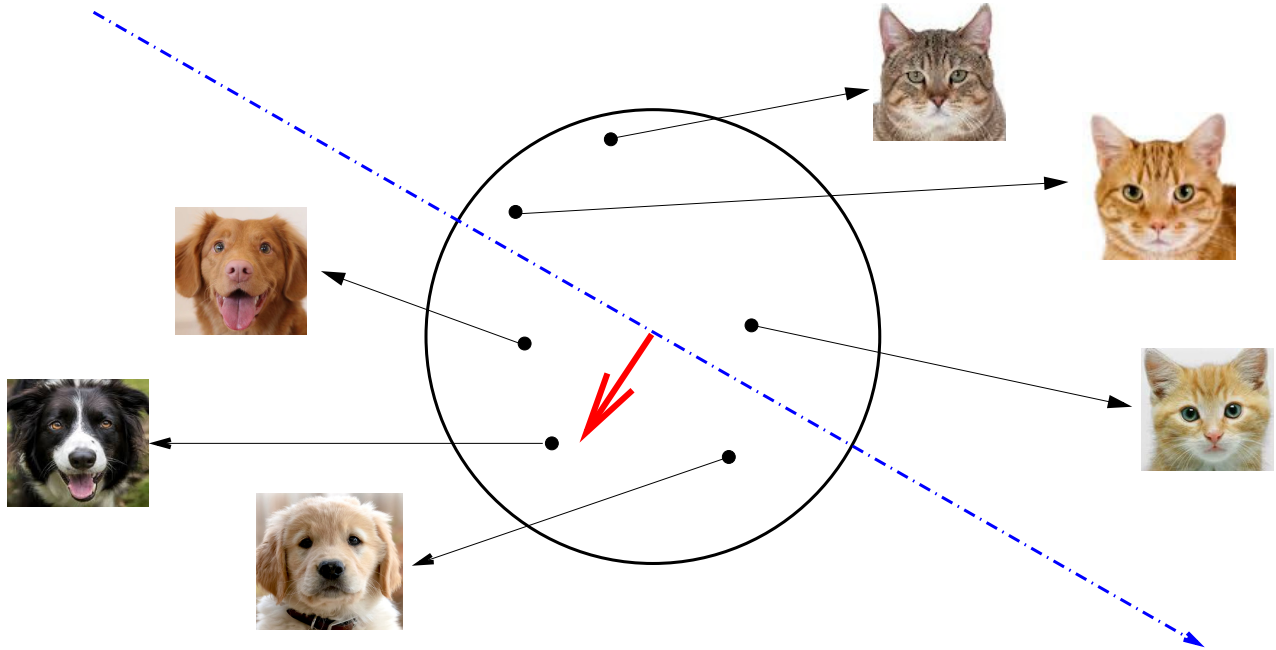
Latent space exploration

Latent space exploration is the process of navigating the internal representation space of a generative model to uncover and control meaningful variations in the output.

It involves manipulating latent vectors to probe structure, interpolate between samples, or perform targeted edits.

Interpreting the Latent Space of GANs for Semantic Face Editing

Attribute editing



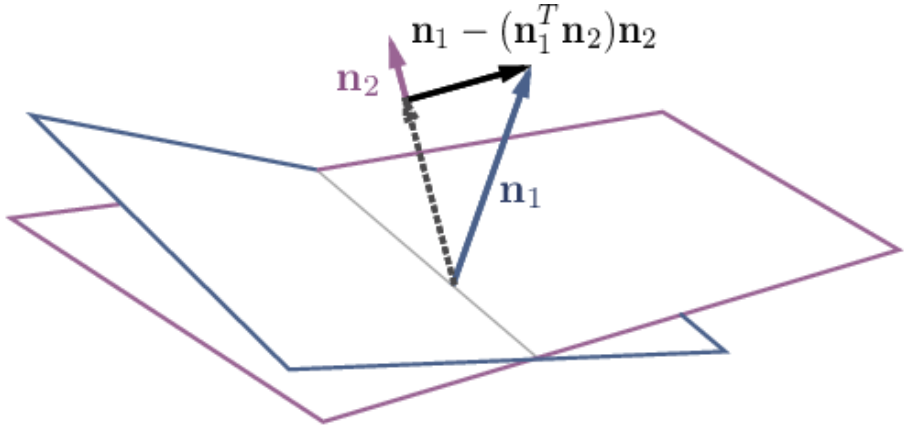
Key ideas behind Representation Learning

The generative process is **continuous**: a small deplacement in the latent space produces a small modification in the visible space.

Real-world data depends on a relatively **small number of explanatory factors of variation** (latent features) providing compressed internal representations.

Understanding these features we may define **trajectories** producing desired alterations of data in the visible space.

Entanglement and disentanglement

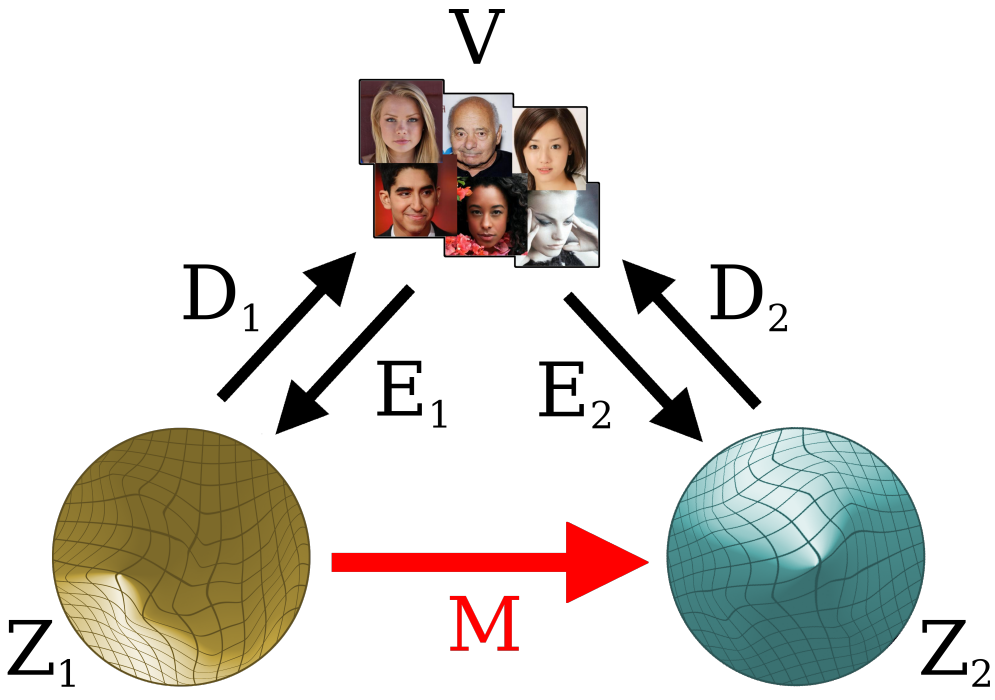


When there is more than one attribute, editing one may affect another since some semantics can be coupled with each other (entanglement).

To achieve more precise control (disentanglement), we can use projections to force the different directions of variation to be orthogonal to each other.

Comparing spaces

Learn a direct map between spaces



Comparing the latent space of generative models

Main findings

We can pass from a latent space to another by means of a simple **linear map** preserving most of the content.

The organization of the latent space seems to be independent from

- the training process
- the network architecture
- the learning objective: GAN and VAE share the same space!

The map can be defined by a small set of points common to the two spaces: the **support set**. Locating these points in the two spaces is enough to define the map.

See also my blog for a discussion.

Prompt Guided generation

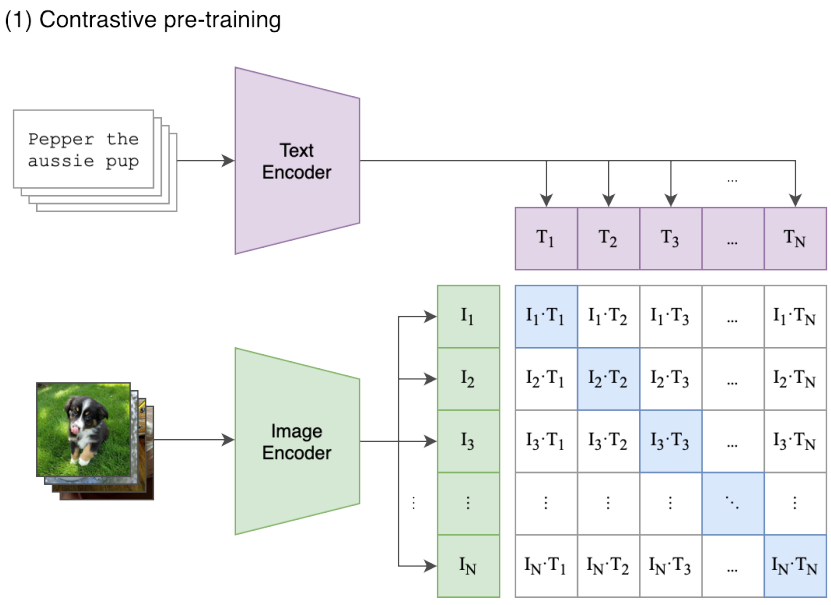
Bridging Text and Images

For text-to-image tasks, text is the condition that defines the desired output.

Text inputs are converted into high-dimensional vectors (embeddings) using pretrained models like CLIP or similar language encoders.

Contrastive Language-Image Pre-training (CLIP)

- Trained on a massive dataset of text-image pairs.
- Maps text and images to a shared latent space, where similar text and image pairs have closer embeddings.
- Key for aligning textual descriptions with visual features.



CLIP-Guided Iterative Optimization

Methodology:

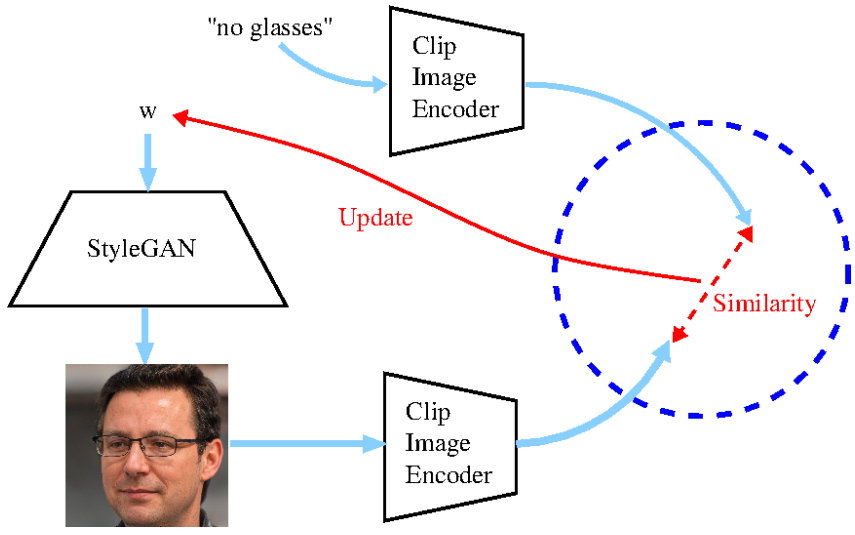
- Start with a pretrained generative model (e.g., StyleGAN).
- Input a text prompt (e.g., "a dog in sunglasses").
- Use CLIP to score how well the generated image aligns with the text.

Iterative Optimization Process:

- Adjust the generator's latent code w iteratively to maximize alignment:

$$w_{t+1} = w_t + \frac{\partial Score(I, c)}{\partial w}$$

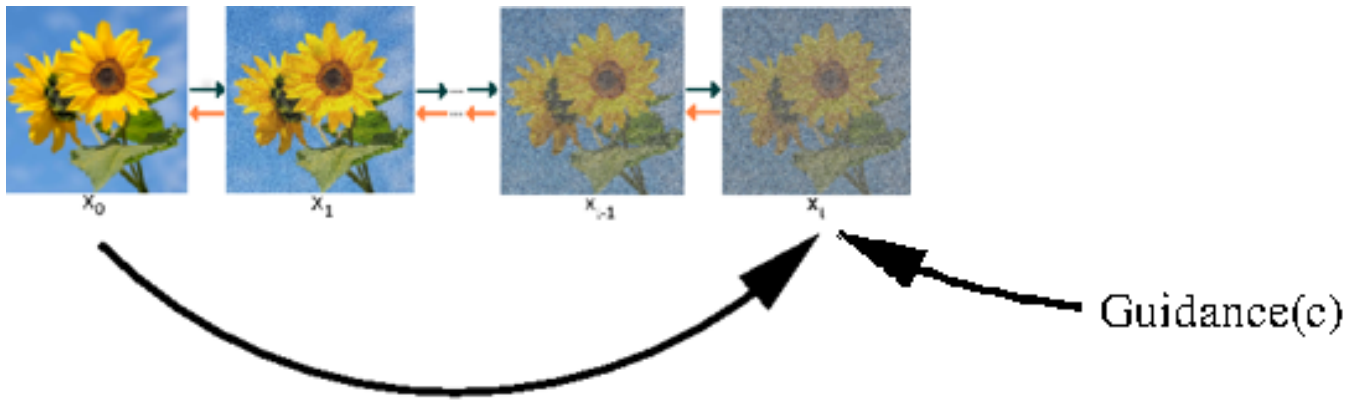
- Continue until the generated image matches the prompt.



Guided sampling in diffusion models

The diffusion sampling phase can incorporate a guiding signal derived from c to steer generation, modifying the latent representation.

This guiding signal is added to the noise reinjected at each step.



There are wwo main classes of guidance:

- **Classifier Based**, using a pretrained classifier to guide the reverse process.
- **Classifer Free**, using text and image embeddings (e.g. CLIP) to align generated images with conditions.

Classifier Guidance

- Define a classifier acting on **noisy images**, computing a probability $p(c|x_t)$.
- When (re)injecting noise, add a component proportional to the **gradient** of the classifier w.r.t. x_t.

Clip-based Guidance

Adjust the sampling trajectories to maximize alignement between the generate image and the text prompt using CLIP gradients:

$$\nabla_{x_t} log p_{CLIP}(c|x_t)$$

where, tipically

$$p_{CLIP}(c|x_t) \approx \|CLIP_{img}(\hat{x}_0^t), CLIP_{text}(c)\|$$

and $\hat{x}_0^t$ is the denoised reconstruction at stage t.

Recap and conlcusion

- Training-time conditioning integrates signals directly into the training pipeline.
- Sampling-time conditioning guides the generation process dynamically. It works on pre-trained models.
- Prompts are more and more elaborated. The condition is a textual embedding of the prompt.
- To bridge text and images we need tools like CLIP, trained to align them in a shared embedding space.

Future trends and advanced imaging applications

Multi-modality

Diffusion models are expanding beyond image to handle:

- **Text-to-Video**: Generating coherent video sequences from text (e.g. Gen-2 by Runwa, or Sora by OpenAI).
- **Audio**: Synthesizing realistic soundscapes.
- **Multimodal Systems**: Combining text, image, video, and audio

The research focus is on enhancing temporal consistency in video generation.

Ethical considerations and mitigations

Diffusion models unlock immense potential but also pose major challenges:

- **Misuse**: Deepfakes, misinformation, unauthorized use of data.
- **Bias**: Potential for reproducing harmful biases in the learning process.

Mitigation stategies:

- Developing watermarking techniques to identify AI-generated content.
- Incorporating safety filters during sampling.

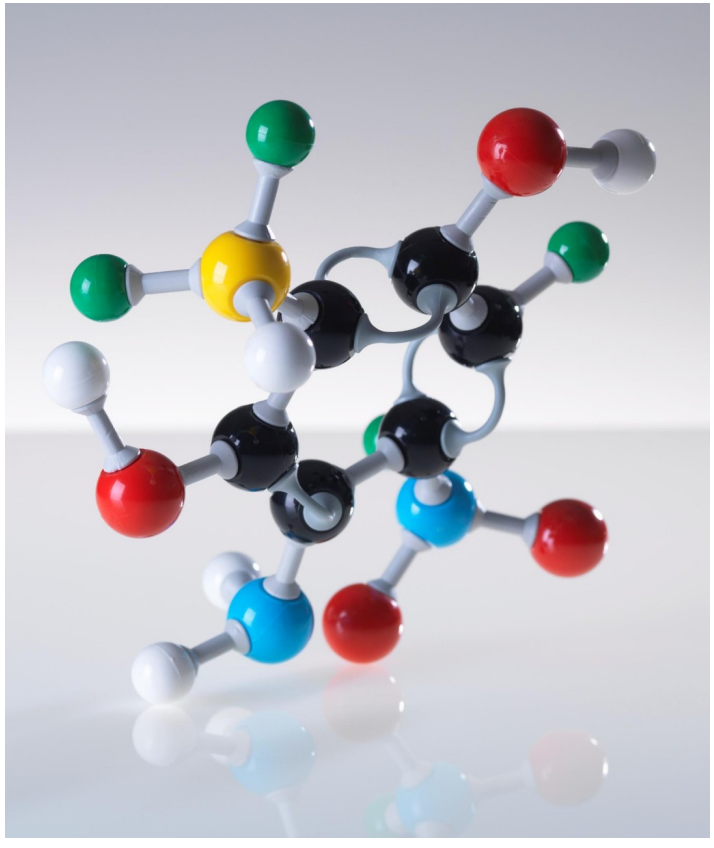
Future focus: Transparent AI systems with built-in safeguards.

Advanced Imaging Applications

Diffusion models are revolutionizing scientific and medical imaging:

- **MRI Reconstruction**: accelerating image acqusition and improving resolution
- **Molecular Design**: generating novel drug candidates.
- **Astronomy**: enhancing image quality in astrophotograaphy.
- **Meteorology**: reanalysis, downscaling and forecasting.

Future trends: integration with domain-specific models for specialized tasks.



Creative Workflows

Diffusion models as tools for artistis and designers:

- Streamlining concept generation and prototyping
- Applications in advertising, marketing, and entertainment

Future trends: seamless integration into creative pipelines with multimodal capabilities.