Computer Vision Notes

Ana Maria Sousa

Hugging Face

(Part II)

Content

1. Multimodal Models

1.1. What is Multimodality?

A modality refers to a way in which something exists or is done. Unimodal models and datasets use only one modality, which has been extensively studied with many tasks and benchmarks but has limitations.

Relying on a single modality might miss important information. Combining multiple modalities increases the information content and reduces the chances of missing crucial cues.

1.1.1. Why is Multimodality important?

Fun fact: *"It Has been studied by the American Psychologist, in particular by Albert Mehrabian who stated this as the 7-38-55 rule of communication, the rule states: In communication, 7% of the overall meaning is conveyed through verbal mode (spoken words), 38% through voice and tone and 55% through body language and facial expressions."*

Communication effectiveness significantly improves when visual cues like body language and facial expressions are present. In the same way, adding modalities in deep learning enhances our understanding of situations/scenario, this suggests that it improves the information content.

The most commonly used modalities in deep learning are: vision, audio and text. Other modalities can also be considered for specific use cases like LIDAR, EEG Data, eye tracking data etc.

1.2. Multimodal Datasets

- **Vision + Text:** Visual Storytelling Dataset, Visual Question Answering Dataset, LAION-5B Dataset.
- **Vision + Audio:** VGG-Sound Dataset, RAVDESS Dataset, Audio-Visual Identity Database (AVID).
- **Vision + Audio + Text:** RECOLA Database, IEMOCAP Dataset.

1.3. Multimodal Tasks

Some multimodal tasks and their variants:

1.3.1.1. Vision + Text:

- Visual Question Answering or VQA: Aiding visually impaired persons, efficient image retrieval, video search, Video Question Answering, Document VQA.
- Image to Text: Image Captioning, Optical Character Recognition (OCR), Pix2Struct.
- Text to Image: Image Generation
- Text to Video: Text-to-video editing, Text-to-video search, Video Translation, Text-driven Video Prediction.

1.3.1.2. Audio + Text:

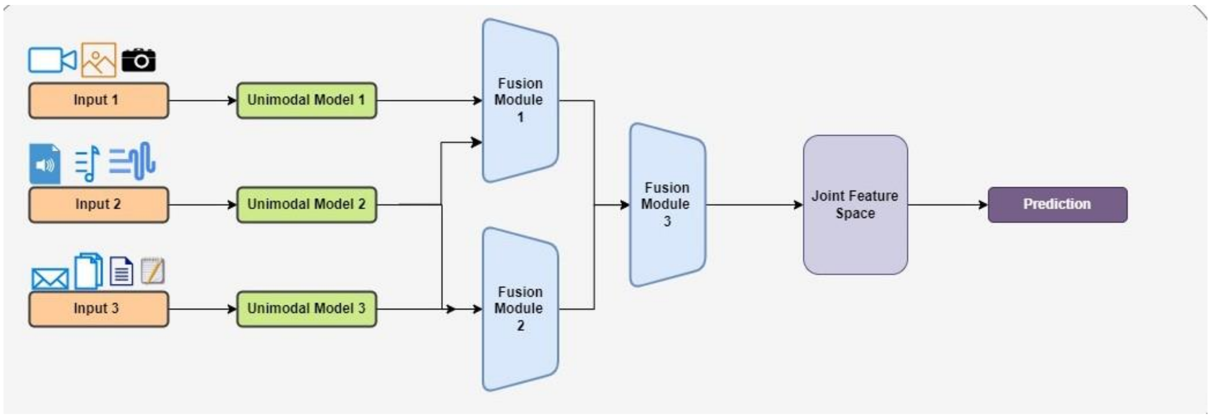
- Automatic Speech Recognition (or Speech to Text): Virtual Speech Assistants, Caption Generation.
- Text to Speech: Voice assistants, Announcement Systems.

1.3.1.3. Multimodal Emotion Recognition (MER)

The **MER** task involves recognition of emotion from two or more modalities like **audio+text**, **text+vision, audio+vision or vision+text+audio.**

As we discussed in the example, MER is more efficient than unimodal emotion recognition and gives clear insight into the emotion recognition task.

1.4. Multimodal Model



A **multimodal model** processes data from multiple sources or modalities simultaneously to perform various tasks. By combining strengths from different modalities like images, text, and audio, these models provide a comprehensive representation of data, enhancing performance across multiple tasks.

<u>To build a multimodal model</u>, we start with <u>individual unimodal models</u> for each modality. These models process data from their respective sources and produce encoded representations. These encoded representations are then <u>fused together</u> using strategies like early fusion, late fusion, or hybrid fusion by a fusion module.

The fusion module's job is to create a unified representation of the encoded data from the unimodal models. Finally, a classification network uses this fused representation to make predictions for the desired task.

1.5. Challenges in Multiple modalities

In pursuit of making a AI capable to understand the world, the field of machine learning seeks to develop models capable of processing and integrating data across multiple modalities.

However, some challenges like representation and **alignment** must be tackled. <u>Representation</u> involves summarizing multimodal data effectively to capture connections among elements of different modalities. <u>Alignment</u> focuses on identifying interactions across all elements. However, handling multimodal data comes with inherent difficulties:

1) One modality may dominate others.
2) Additional modalities can introduce noise.
3) Full coverage over all modalities is not guaranteed.
4) Different modalities can have complicated relationships.

2. Vision Language Models (VLMs)

Vision Language Models (VLMs) understand and process both vision and text, making them efficient for tasks like Visual Question Answering and Text-to-image search.

They excel at multimodal search by mapping text and image pairs into a joint embedding space, where each pair is represented as an embedding.

Downstream tasks and searches can then be performed using these embeddings. This joint space ensures that similar text and image embeddings are close together, enabling searches based on text or images.

2.1. Base paradigm

In VLMs, we employ the paradigm proposed in ULMFIT paper and extend it to combine visual images, yielding the desired results.

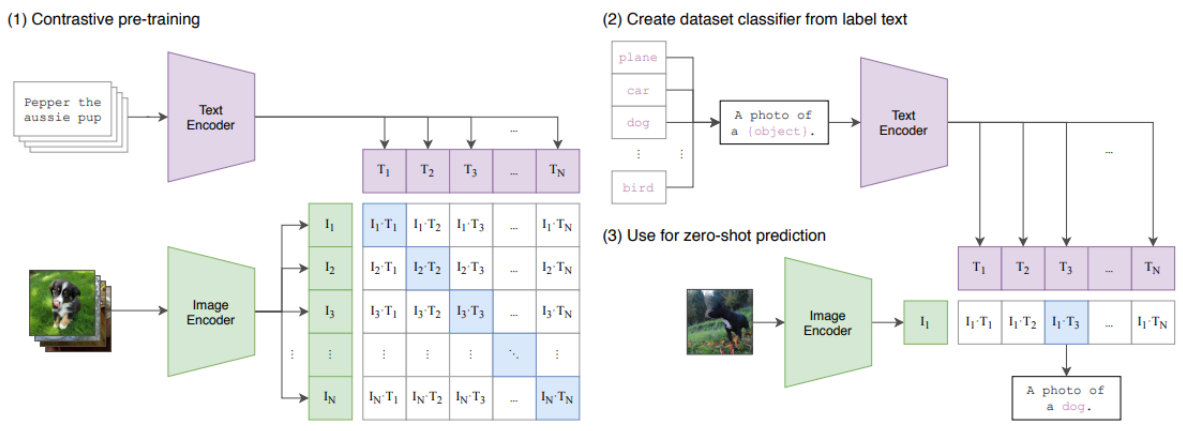
1. Pre-training the model with extensive training data.
2. Fine-tuning the pre-trained model with **task-specific** data.
3. Utilizing the trained model for **downstream tasks** such as classification.

In 2021, OpenAI released a groundbreaking paper called **CLIP,** which had a significant impact on the adoption of vision-language models (VLMs).

CLIP uses an image-text contrastive objective, where paired images and texts are brought close together while pushing others far away in the embedding space. This pre-training method enables VLMs to learn rich vision-language correspondence knowledge, allowing for zero-shot predictions by matching the embeddings of any given images and texts.

2.2. Mechanism for Text & Images combination

To make Vision Language Models (VLMs) effective, it's crucial to combine text and images **meaningfully** for joint learning. One common method is by providing image-text pairs.


(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ... I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

1. Extract image and **text features** using text and **image encoders**. It can be CNN or transformer-based architectures.

2. Learn the vision-language correlation with certain pre-training objectives*.
3. With the learned vision-language correlation, VLMs can be evaluated on unseen data in a **zero-shot** manner by matching the embeddings of any given images and texts.

*The pre-training objective can be divided into three groups:

- **Contrastive** objectives teach VLMs to discern between samples by bringing paired examples closer and separating others in the embedding space.
- **Generative** objectives to make VLMs learn semantic features by training networks to generate image/text data.
- **Alignment** objectives align the image-text pair via global image-text matching or local region-word matching on embedding space.

2.2.1. Research

Existing research primarily aims to improve Vision Language Models (VLMs) from three main angles:

- by gathering large-scale, informative image-text datasets;
- by devising efficient models to learn from extensive data;
- by developing new pre-training methods to enhance the understanding of vision-language correlation.

VLM pre-training endeavors to equip models with the ability to understand the relationship between images and text, with the ultimate goal of achieving effective **zero-shot** predictions on various visual recognition tasks such as segmentation and classification.

2.3. Modes of Learning

VLMs can be categorized based on how they utilize the two learning modes:

1. *Joint Training of Embedding Features:* Images are split into smaller patches, each treated as a "token," and jointly trained with text tokens to create combined representations. Examples include VisualBERT, SimVLM.

2. *Image Embeddings as Prefixes:* Image embeddings are learned to be compatible with a pre-trained, frozen language model, without altering its parameters. Examples include Frozen, ClipCap.

3. *Cross-Attention Mechanisms:* Visual information is integrated into the language model using specially designed cross-attention layers to balance text and visual input. Examples include VisualGPT.

4. *Combining Without Training:* Recent methods like MAGiC use guided decoding to sample the next token without fine-tuning.

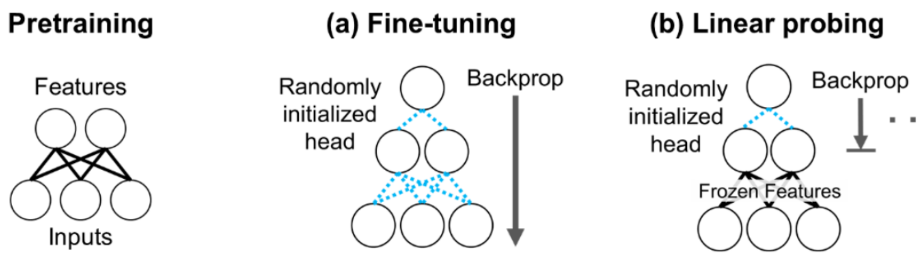
2.3.1. Some common VLM Datasets

- **MSCOCO** contains images paired with 5 independent captions each.
- **NoCaps** designed to measure generalization to unseen classes and concepts, where in-domain contains images portraying only COCO classes, near-domain contains both COCO and novel classes, and out-of-domain consists of only novel classes.

- **Conceptual Captions** contains million pairs of images and captions, mined from the web and post-processed.
- **ALIGN** a noisy dataset of over one billion image alt-text pairs, obtained without expensive filtering or post-processing steps in the Conceptual Captions dataset.
- **LAION** dataset consists of image-text-pairs. The image-text-pairs have been extracted from the Common Crawl web data dump and are from random web pages crawled between 2014 and 2021.
- **Hateful Memes** are created to address this problem by understanding the models capability to an extreme by adding difficult exemples
- Winoground was designed to figure out how good CLIP is.

2.4. Downstream Tasks and Evaluation

To evaluate VLMs, two main methods are used: zero-shot prediction and linear probing.



Zero-shot prediction is the most common, where pre-trained VLMs are applied directly to new tasks (downstream task) without any additional fine-tuning specific to those tasks.

In linear probing, we freeze the pre-trained VLM and train a linear classifier on the VLM-encoded embeddings to assess its representation quality. To evaluate these models, we measure their performance on various datasets.

2.5. Multimodal Tasks and Models

- *Visual Question Answering (VQA) and Visual Reasoning:* Teach machines to answer questions about images, going beyond recognition to infer relationships and understand context.

Models: BLIP-VQA; DePlot; VLIT

- *Document Visual Question Answering (DocVQA):* Enables computers to understand and answer questions about documents by combining computer vision and natural language processing.

Models: LayoutLM, Donut, Nougat

- *Image captioning:* Bridges vision and language by crafting descriptive sentences for images, allowing computers to narrate the visual world.

Models: ViT-GPT2, BLIP, git-base

- *Image-Text Retrieval:* Acts as a matchmaker for images and their descriptions, enabling powerful applications like image search and automatic captioning.

Models: CLIP

- Visual grounding: Links language with specific parts of an image, allowing AI to pinpoint objects based on natural language descriptions.

Models: OWL-VIT

- Text-to-Image generation: Transforms written descriptions into unique images, creating visual worlds from the power of words.

Models: Auto-regressive Models, Stable diffusion Models

2.6. CLIP and Relatives

2.6.1. Pre-CLIP

Before CLIP, several influential papers made significant strides in multimodal AI using deep learning:

1. Multimodal Deep Learning: Foundational paper demonstrated the integration of different data types using deep learning, setting the stage for future multimodal AI innovations.
2. Deep Visual-Semantic Alignments for Generating Image Descriptions: This study improved the alignment of textual data with specific image regions, enhancing the interpretability and understanding of visual-textual relationships.
3. Show and Tell: A Neural Image Caption Generator: This paper combined CNNs and RNNs to generate descriptive language from visual information, marking a significant advancement in practical multimodal AI.

2.6.2. Post-CLIP

The CLIP revolutionized multimodal models, leading to several significant developments:

4. **CLIP:** OpenAI's CLIP was a game-changer by learning from vast internet text-image pairs and enabling zero-shot learning, unlike earlier models.
5. **GroupViT:** This model advanced segmentation and semantic understanding by integrating these aspects with language, demonstrating sophisticated language and vision integration.
6. **BLIP:** BLIP introduced bidirectional learning between vision and language, enhancing the ability to generate text from visual inputs.
7. **OWL-ViT:** Focusing on object-centric representations, OWL-ViT improved the understanding of objects within images in the context of text.

Additional refs:

- *Meta released the first multimodal AI model to bind information from 6 different modalities*
- *ULMFIT Paper*

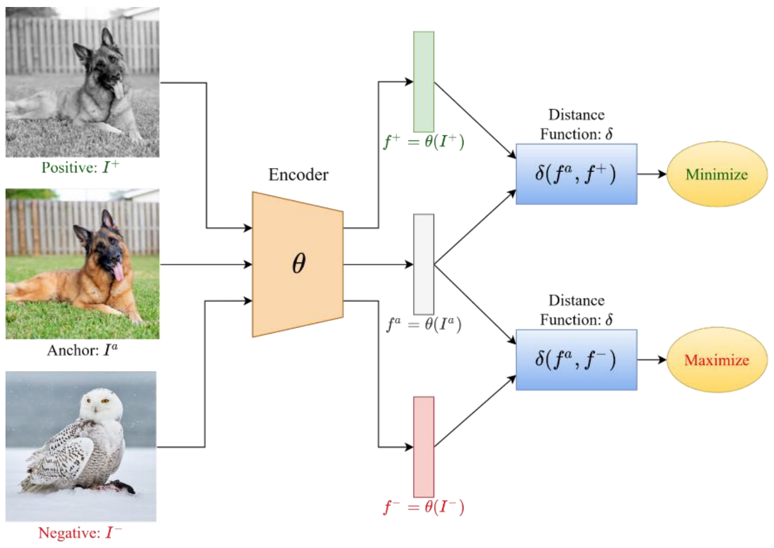
2.6.3. Contrastive Language-Image Pre-training (CLIP)

CLIP is a neural network adept at grasping visual concepts through natural language supervision. It trains a text encoder and an image encoder together to match captions with images.

This design allows CLIP to perform well on different visual classification tasks without additional training, similar to the "zero-shot" learning seen in GPT-3 models. It can recognize visual categories just by receiving their names.

2.6.3.1. Contrastive Learning

Contrastive Learning is an unsupervised deep learning technique used for learning data representations. It works by ensuring that similar items are close together in the representation space, while dissimilar items are far apart.



Contrastive loss is a fundamental training objective in contrastive learning. It processes pairs of samples, aiming to position similar samples close together and dissimilar samples far apart in the embedding space.

For instance, given a list of input samples from various classes, the goal is to create a function that **ensures embeddings of samples from the same class are close**, while embeddings from **different classes are distant**.

$$L = 1[y_i = y_j]||x_i - x_j||^2 + 1[y_i \neq y_j]max(0, \epsilon - ||x_i - x_j||^2)$$

Explaining in simple terms:

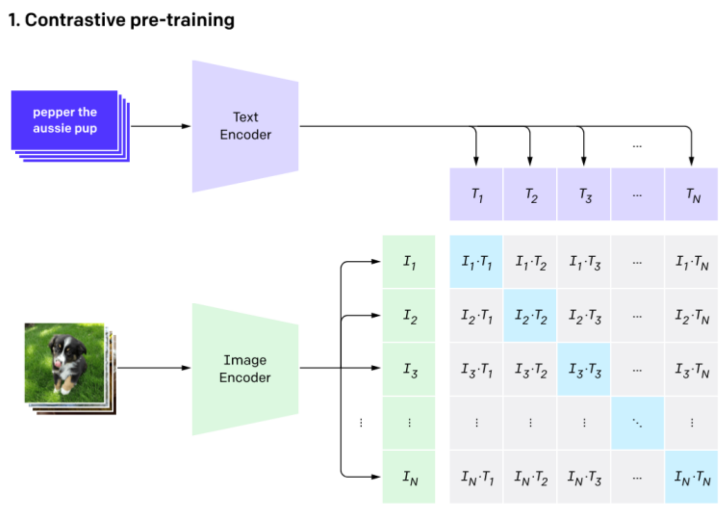
- If the samples are similar$y_i = y_j$, then we minimize the term$||x_i - x_j||^2$ that corresponds to their Euclidean distance, i.e., we want to make them closer;
- If the samples are dissimilar$(y_i \neq y_j)$, then we minimize the term$max(0, \epsilon - ||x_i - x_j||^2)$ that is equivalent to maximizing their euclidean distance until some limitϵ, i.e., we want to make them distant from each other.

2.6.3.2. Contrastive pre-training

Given a batch **of image-text pairs, CLIP calculates the cosine similarity** between every possible image and text pair in the batch.

The goal is to increase the similarity for the correct pairs and decrease it for the incorrect ones. This is achieved by optimizing a symmetric cross-entropy loss based on these similarity scores.

Text Encoder and Image Encoder: CLIP's design features independent encoders for images and text, allowing flexibility in their choice (Vision Transformer, ResNet…)


1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

2.6.3.3. Limitations & Relevance

While **CLIP proficiency in zero-shot classification**, it is unlikely to outperform a specialized, fine-tuned model. Its generalization is also limited, especially with unfamiliar data. The CLIP paper reveals that its performance and biases are influenced by the chosen categories, as seen in tests with the Fairface dataset.

In summary, **CLIP stands out** for its expertise in zero-shot learning, enabling it to categorize images into classes it hasn't been explicitly trained on. This exceptional generalization ability is a result of its unique training approach, which involves learning to associate images with text descriptions.

2.6.4. Multimodal Text Generation (BLIP)

BLIP stands to Bootstrapping Language-Image Pre-training and extends the capabilities of multimodal models to include text generation, by address the issue of noise in the data.

Multimodal models rely on large datasets gathered from the internet, including pairs of images and alternative text (alt-text). Alt-texts often don't accurately describe the images, creating noise in the data that hinders learning.

BLIP introduced **CapFilt**, a mechanism that filters out noisy pairs and generates accurate captions for images. CapFilt consists of two deep learning models fine-tuned with human-annotated data.

2.6.4.1. BLIP Architecture and Training

BLIP architecture integrates:

1. **Vision Transformer (ViT):** Utilizes self-attention and feed-forward blocks, along with a [CLS] token for embedding representation.
2. **Unimodal Text Encoder:** Resembles BERT's architecture, employing contrastive loss for aligning image and text representations.
3. **Image-Grounded Text Encoder:** Utilizes cross-attention layers to integrate image and text embeddings, with an [Encode] token replacing [CLS]. It assesses the congruence of image-text pairs using a linear layer.

4. **Image-Grounded Text Decoder:** Trained via cross-entropy loss in an autoregressive manner, featuring causal self-attention for tasks like caption generation or answering visual questions.

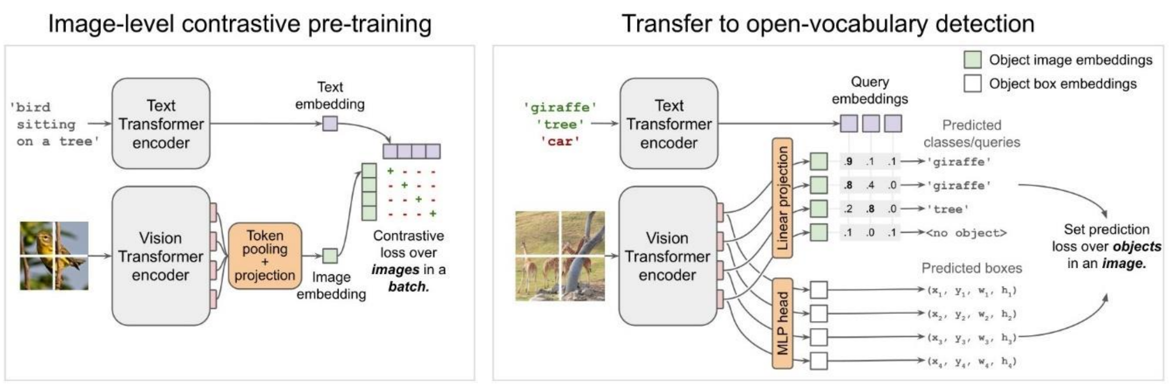
This architecture enables BLIP to proficiently process diverse tasks involving image and text interactions.

2.6.5. Multimodal Object Detection (OWL-ViT)

Object detection, a crucial aspect of computer vision, has progressed notably with models such as YOLO. Yet, conventional models like YOLO encounter challenges in detecting objects that differ significantly from those in their training data.

OWL-ViT signifies a significant advancement in open-vocabulary object detection. It initiates with a training process similar to CLIP, emphasizing the development of a vision and language encoder through contrastive loss. This enables the models to learn a shared representation space for both visual and textual data.

2.6.5.1. Fine-tuning for Object Detection



In OWL-ViT, the fine-tuning stage for object detection introduces a key innovation.

Unlike CLIP, which uses token pooling and a final projection layer, OWL-ViT opts for a linear projection of each output token to generate image embeddings per object. These embeddings are utilized for classification, while box coordinates are inferred from token representations via a small MLP.

This novel approach enables OWL-ViT to not only detect objects but also accurately determine their spatial locations within images, marking a notable advancement in traditional object detection models.

2.6.5.2. Relevance

After fine-tuning, OWL-ViT demonstrates remarkable performance in **open-vocabulary** object detection.

Its ability to recognize objects not explicitly included in the training data is attributed to the shared embedding space of its vision and text encoders. This unique feature enables OWL-ViT to utilize both images and textual queries for object detection, expanding its flexibility and utility.

OWL-ViT's object detection approach is a significant advancement in AI models' comprehension and interaction with the visual world. By merging language understanding with visual perception, it expands the scope of object detection, leading to more precise and adaptable models capable of identifying a wider array of objects. Those model capabilities are essential for applications demanding nuanced comprehension of visual scenes, especially in dynamic, real-world settings.

Additional refs:

- *An Introduction to Contrastive Learning*
- *Contrastive Representation Learning*
- *CLIP paper*
- *CLIP by Lilian Weng*
- *BLIP paper*
- *BLIP-2 paper*
- *OWL-VIT paper*
- *OWL-ViT 2 model*
- ***Blog on Multimodality and LLMs by Chip Huyen***
- *Vision-Language Pre-training: Basics, Recent Advances, and Future Trends*
- *Document Collection Visual Question Answering*
- *Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection*

2.7. Transfer Learning of Multimodal Models

There are several approaches for adapting multimodal models to different tasks:

1. **Zero/few-shot learning:** This method involves utilizing large pretrained models capable of solving tasks not present in the training data. It proves useful when labeled data for a task is scarce or unavailable entirely.
2. **Training the model from scratch**: In situations where pretrained model weights are unavailable or the model's dataset significantly differs from your own, training from scratch is necessary. This involves initializing model weights randomly or through advanced methods like He initialization and conducting standard training. However, it requires substantial amounts of training data.
3. **Transfer learning:** Unlike training from scratch, transfer learning utilizes the weights of a pretrained model as initial weights, leveraging prior knowledge to adapt to new tasks more efficiently.

Transfer learning involves leveraging knowledge gained from solving one problem to address another similar problem. In deep learning, this often means initializing a model with weights learned from a pretrained model trained on a large dataset. These pretrained models capture valuable patterns and relationships in the data, which are embedded in their weights. By transferring this knowledge to a new model, we can kickstart its learning process and improve its performance on the target task, especially when labeled data for the new task is limited.

2.7.1.1. Advantages & Challenges

Advantages of transfer learning include:

- **Resource Efficiency:** Leveraging pretrained models trained on extensive datasets reduces the need for computing resources and labeled data.
- **Reduced Data Requirements:** Less data is needed to achieve decent quality on the test sample.

- **Knowledge Transfer:** The model capitalizes on pre-existing knowledge encoded within the pretrained model's weights, often resulting in enhanced performance on the new task.

Challenges of transfer learning include:

- **Domain Shift:** Adapting knowledge from the source domain to the target domain can be challenging if data distributions differ substantially.
- **Catastrophic Forgetting:** During the fine-tuning process, the model may adjust its parameters to adapt to the new task, potentially leading to the loss of previously learned knowledge or representations.

3. Generative Models

3.1. Introduction

Generative models focus on understanding and learning the underlying distribution of the data. These models aim to generate new samples that are similar to the original data by modeling the joint probability distribution P(X,Y).

Machine learning models can generally be separated into two large families, generative models and discriminative models. Discriminative models are the most well-know and just concentrate on learning the boundaries that separate different classes within the data.

In summary:

- **Generative Models:** Aim to model the actual data distribution and can generate new data points. Learn P(X,Y) the joint probability distribution of inputs and outputs.
  Example: Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs).

- **Discriminative Models:** Aim to find decision boundaries to classify data into different categories. Learn P(Y|X), the conditional probability of outputs given the inputs.
  Example: Convolutional Neural Networks (CNNs), Support Vector Machines (SVMs).

Generative Models are used for tasks such as image generation, inpainting, style transfer, and data augmentation. They can produce realistic images from learned distributions. Here we will consider some tasks as:

- *noise to image (DCGAN);*
- *text to image (diffusion models);*
- *image to image (StyleGAN, cycleGAN, diffusion models)*

3.2. Metrics

Evaluating generative models can be challenging because there often isn't a clear "ground truth" for the generated images, making it difficult to measure image quality.

3.2.1.1. Fréchet Inception Distance (FID)

FID is a crucial metric in assessing the quality of images generated by models, especially in the realm of GANs.

Unlike simpler metrics, FID offers a more robust evaluation by capturing high-level features and then comparing the distributions of features extracted from both real and generated images.

Lower FID scores signify a closer resemblance between generated and real images, indicating **higher quality**. Introduced as an improvement over the Inception Score, FID is known for its resistance to noise and artifacts, making it a reliable measure of image fidelity.

3.2.1.2. The Inception Score (IS)

IS another a metric used to evaluate the quality and diversity of images generated by generative models. It assesses the likelihood and distinctiveness of generated images based on their class predictions.

The score is calculated by first using a pre-trained classifier network (typically Inception-v3) to classify generated images and then computing the entropy of the class distribution and the marginal entropy of the generated images.

A higher Inception Score generally indicates **better quality and diversity** in the generated images, although it has limitations, such as being sensitive to mode collapse in GANs.

3.2.1.3. PSNR (peak signal-to-noise ratio)

PSNR measures the quality of images by comparing them to a reference image, typically by calculating the mean squared error.

It serves as a benchmark for assessing image quality by quantifying the level of noise present in an image compared to a reference image. It operates on the principle of mean squared error, providing a numerical value that indicates the fidelity of the image reconstruction process.

PSNR values typically range between 25 and 34, with results above 34 indicating exceptionally high-quality images where noise levels are minimal, thus making it a widely-used metric in various image processing applications.

3.2.1.4. SIM (Structural Similarity Index)

SSIM assesses the similarity between two images by considering their luminance, contrast, and structure components.

Ranging between 0 and 1, SSIM scores closer to 1 suggest a high degree of similarity between images, considering both pixel-wise and structural differences. This metric is particularly valuable in scenarios where precise visual fidelity is essential, such as medical imaging or video compression.

3.2.1.5. CLIP Score

Offers a novel approach to evaluating text-to-image models, focusing on the alignment between generated images and textual prompts.

By leveraging the CLIP model to compute cosine similarity, this metric assesses the model's ability to translate textual descriptions into visually coherent representations.

With scores ranging from 0 to 100, higher CLIP Scores indicate a stronger correspondence between textual input and generated images, facilitating the assessment and refinement of text-to-image generation models in natural language processing tasks.

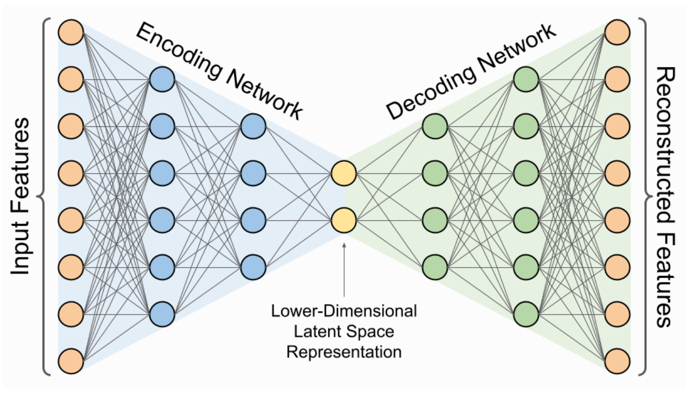
3.3. Autoencoder

Autoencoders, a class of neural networks primarily utilized for **unsupervised learning** and **dimensionality reduction,** operate on the principle of encoding input data into a lower-dimensional representation and subsequently decoding it back to its original form while minimizing **reconstruction error.**

The architecture of an autoencoder comprises two main components:

- The encoder: The encoder transforms input data into a compressed or latent representation through one or more layers of neurons, progressively reducing dimensions.
- The decoder. The decoder takes the compressed representation from the encoder and endeavors to reconstruct the initial input data. It comprises layers arranged in the reverse order, gradually increasing dimensions to recreate the original data.

This process enables autoencoders to learn meaningful representations of input data and find efficient encodings for various tasks, including data compression, denoising, and anomaly detection.

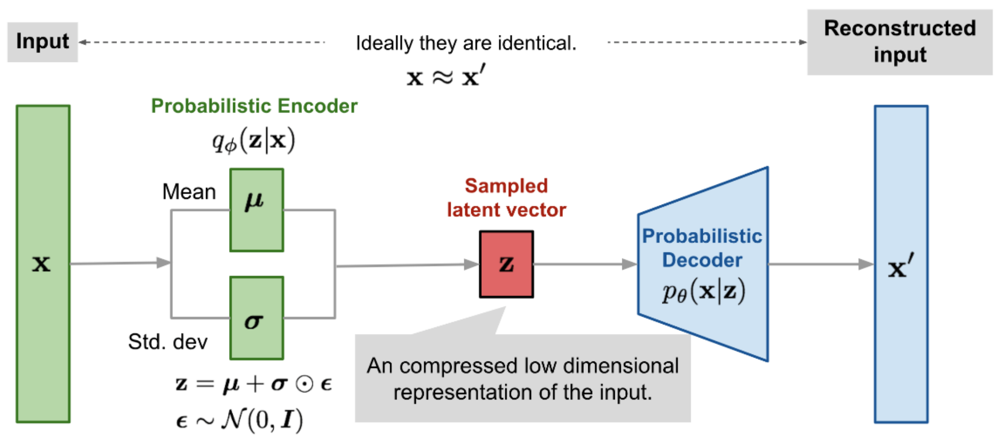


3.3.1. Variational Autoencoders (VAEs)

Variational Autoencoders (VAEs) represent a significant advancement over traditional autoencoders by introducing a probabilistic framework to the encoding and decoding process.

Unlike deterministic autoencoders, which produce a single fixed vector representation of input data, VAEs model the latent space as a probability distribution. This probabilistic nature enables VAEs to capture the uncertainty inherent in the data, allowing for a more nuanced representation of input encodings. In VAEs, the decoder samples from this probability distribution rather than relying on a deterministic latent vector.

The latent space in VAEs serves as a structured and continuous representation of the input data. This latent space facilitates smooth interpolation between different data points, enabling seamless transitions and ensuring that similar points in the latent space yield similar generations. This continuous nature of the latent space makes VAEs particularly suitable for tasks such as data generation and interpolation.


Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
Probabilistic Encoder
$q_\phi(\mathbf{z}|\mathbf{x})$
Mean
$\boldsymbol{\mu}$
Std. dev
$\boldsymbol{\sigma}$
$\mathbf{x}$
$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$
Sampled latent vector
$\mathbf{z}$
An compressed low dimensional representation of the input.
Probabilistic Decoder
$p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$

Variational Autoencoders (VAEs) represent this feature as a probabilistic distribution, allowing for variability in generated images by sampling from this distribution.

- Probabilistic Modeling

In VAEs, the latent space is modeled as a probability distribution, typically a multivariate Gaussian. This distribution is parameterized by mean and standard deviation vectors, produced by the encoder. The learned representation z is then sampled from this distribution and fed into the decoder.

- Loss Function

The VAE loss function combines reconstruction loss and KL divergence. The reconstruction loss measures how well the model reconstructs the input, while the KL divergence ensures the learned distribution resembles a chosen prior (usually Gaussian).

Together, these components encourage learning a latent representation that captures both data distribution and the specified prior.

- Encouraging Meaningful Latent Representations

Incorporating the KL divergence into the loss function encourages the VAE to learn a structured latent space where similar data points are closer.

This balance between latent loss and reconstruction loss is crucial: a smaller latent loss can lead to generated images that closely resemble the training set but may lack quality, while a smaller reconstruction loss results in well-reconstructed images during training but may hinder the generation of novel images.

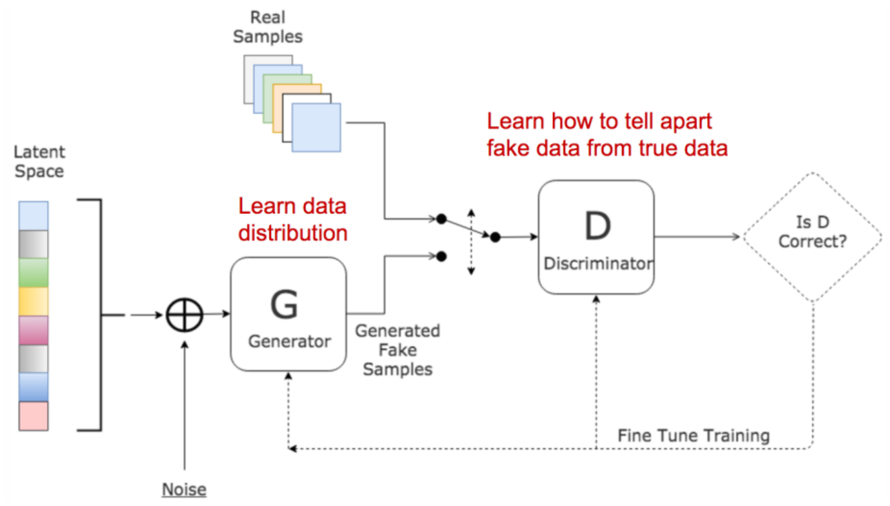
Achieving a balance between these two aspects is essential for optimal image reconstruction and generation.

3.4. Generative Adversarial Networks

Generative Adversarial Networks (GANs) were introduced by Goodfellow et al. in 2014 as a novel approach for generating new data samples by learning the statistical distribution of existing data through an adversarial process.

GANs consist of two sub-models trained simultaneously: **the Generator and the Discriminator**.

- The Generator learns to produce artificial samples from random vectors sampled from a defined latent space distribution.
- The Discriminator acts as a binary classifier, distinguishing between real and generated samples, and outputs 0 for artificial and 1 for real samples.



3.4.1.1. Training Process

GANs are trained through an adversarial game where the Generator tries to create realistic samples to fool the Discriminator, while the Discriminator aims to accurately distinguish between real and fake samples.

The models are improved iteratively until they reach a **Nash equilibrium**, where the Discriminator can no longer reliably tell the difference between real and generated samples, outputting equal probabilities for both. At this point, the GANs are considered to have converged, and the Generator can produce high-quality artificial samples.

$$\min_G \max_D f(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}\big[\log(1 - D(G(z)))\big]$$

However, this balance is difficult to achieve. The learning process consists of training two models simultaneously, most of the times they are affected by some instability, because each time the weights are updated, the improvements of one model appear to the detriment of the other. Formally, G and D play a minmax game where the evaluation function is defined by equation 1. When the training process is unstable, the model can end up in model collapse or other failure modes, producing undesired results.

3.4.1.2. GANs VS VAEs

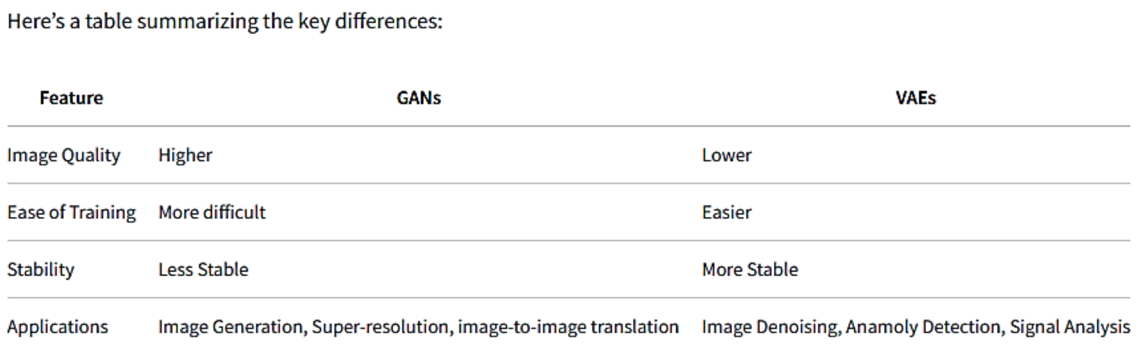
Generative Adversarial Networks (GANs) excel in generating high-quality images with sharp details and realistic textures, making them ideal for tasks such as image generation.

However, their training process can be challenging and prone to instability, requiring careful tuning and management to achieve desirable results. Despite these challenges, GANs are well-suited for tasks like super-resolution and image-to-image translation, where they can produce visually stunning outputs.

On the other hand, **Variational Autoencoders (VAEs)** offer a more stable training process and are easier to train compared to GANs.

However, they may produce images with lower resolution and less detail, often appearing blurry or with unrealistic features.

VAEs are commonly used for tasks such as **image denoising and anomaly detection**, where their ability to learn meaningful latent representations proves beneficial.



Here's a table summarizing the key differences:

Feature	GANs	VAEs
Image Quality	Higher	Lower
Ease of Training	More difficult	Easier
Stability	Less Stable	More Stable
Applications	Image Generation, Super-resolution, image-to-image translation	Image Denoising, Anamoly Detection, Signal Analysis

While VAEs may not match the image quality of GANs, they still find utility in various applications that prioritize interpretability and stability over visual fidelity. Ultimately, the choice between GANs and VAEs depends on the specific requirements of the task at hand, weighing factors such as image quality, stability, and the nature of the data being generated or manipulated.

3.5. StyleGAN Variants

3.5.1.1. What is missing in Vanilla GAN?

Generative Adversarial Networks (GANs) generate realistic images **but lack control over specific feature**s in the generated images. In a standard/**Vanilla GAN** setup, the Generator creates images from **random noise**, while the Discriminator distinguishes between real and generated images. Despite training, traditional GANs don't allow for direct manipulation of image features, as these features are entangled.

StyleGAN addresses this limitation by **modifying the Generator** architecture while keeping the **Discriminator unchanged.** This new architecture **allows for explicit control over image features**. For example, with StyleGAN, you can generate images of a female wearing glasses by controlling high-level attributes (like gender and accessories) and low-level details (like skin texture and hair placement).

3.5.1.2. StyleGAN 1 components & benefits

StyleGAN improves upon traditional GANs by providing users with the ability to manipulate specific image characteristics, making it useful for tasks like privacy preservation and image editing.

The easiest way for GAN to generate **high-resolution images** is to remember images from the training dataset and while generating new images it can add random noise to an existing image. In reality, StyleGAN doesn't do that rather it learn features regarding "human face" and generates a new image of the *"human face"* that doesn't exist in reality.

Diagram (a) shows ProgressiveGAN, a Vanilla GAN variant that progressively generates higher resolution images for more realistic results. The generator in ProgressiveGAN starts with low resolution (e.g., 4x4) and gradually increases it (e.g., 8x8).

Diagram (b) represents the StyleGAN architecture, which introduces several key components:

- **Mapping Network:** Transforms the input noise vector into an intermediate latent space.
- **AdaIN (Adaptive Instance Normalization):** Normalizes features to control style and content separately.
- **Noise Vector Concatenation**: Adds randomness to fine details, enhancing image diversity.

These modifications allow **StyleGAN** to generate images with controllable features, addressing the limitations of traditional GANs.

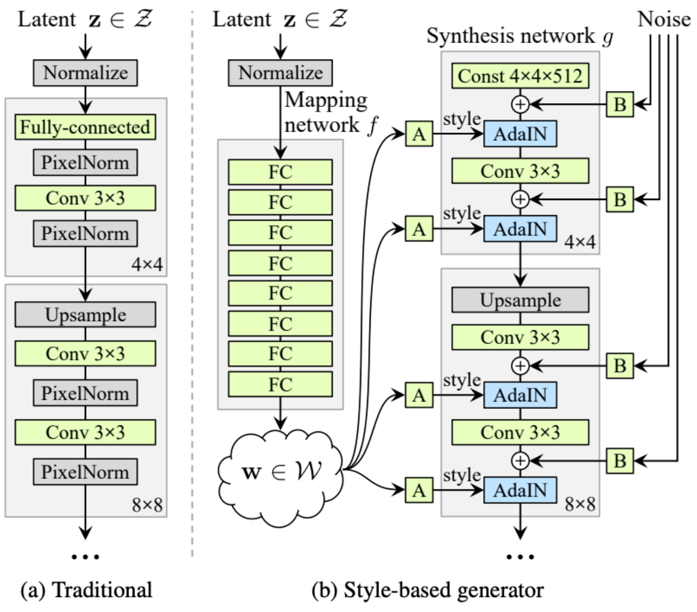


Figure 1. While a traditional generator [30] feeds the latent code though the input layer only, we first map the input to an intermediate latent space $\mathcal{W}$, which then controls the generator through adaptive instance normalization (AdaIN) at each convolution layer. Gaussian noise is added after each convolution, before evaluating the nonlinearity. Here "A" stands for a learned affine transform, and "B" applies learned per-channel scaling factors to the noise input. The mapping network f consists of 8 layers and the synthesis network g consists of 18 layers — two for each resolution ($4^2 - 1024^2$). The output of the last layer is converted to RGB using a separate 1×1 convolution, similar to Karras et al. [30]. Our generator has a total of 26.2M trainable parameters, compared to 23.1M in the traditional generator.

Mapping Network

In StyleGAN, instead of directly feeding the latent code (or noise vector) z to the Generator as in traditional GANs, it is first mapped to an intermediate latent space ww using a series of 8 Multi-Layer Perceptron (MLP) layers.

This process has two main **advantages**:

1. ***Disentanglement of Feature Space:*** Mapping z to w helps disentangle the feature space. In a disentangled space, changing a single feature value in the latent code should affect only one specific aspect of the generated image. For example, in a 512-dimensional latent code, if you alter the 4th value and this value corresponds to the **'smile'** feature, only the smile should change in the generated image.
2. ***Passing Latent Code to Each Layer:***
    i. Instead of providing the latent code w only to the initial layer of the Generator, StyleGAN passes w to every layer of the Synthesis Network (the Generator).
    ii. This approach allows different layers to control different aspects of the image. Lower layers, which handle lower resolutions, control high-level features like pose, general hairstyle, face shape, and eyeglasses.
    iii. Higher layers, which handle higher resolutions, control finer details like small-scale facial features, hairstyle intricacies, and whether the eyes are open or closed.

This enhances the ability to manipulate and generate images with specific desired characteristics.

Adaptive instance normalisation (AdaIN)

AdaIN (Adaptive Instance Normalization) in StyleGAN allows dynamic adjustment of normalization parameters (mean and standard deviation) based on style information derived from the latent code w.

Instead of directly using w , it is transformed into a 'style' representation y and applied to different blocks of the synthesis network. This enables the generator to modulate its behavior and apply different styles or characteristics to various parts of the generated image, enhancing control and flexibility in the image generation process.

$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i}\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

Concatenation of Noise vector

In traditional GANs, the generator must independently learn to create fine, stochastic features like hair positions and skin pores. This pixel-level randomness, which should vary across images, is difficult for the generator to achieve without explicit structure, often resulting in less diversity.

StyleGAN addresses this by **adding a noise map to the feature map** in each block of the synthesis network (generator). This noise map helps each layer introduce diverse stochastic details without relying solely on the generator's inherent capabilities. This approach effectively produces more varied and realistic fine details in the generated images.

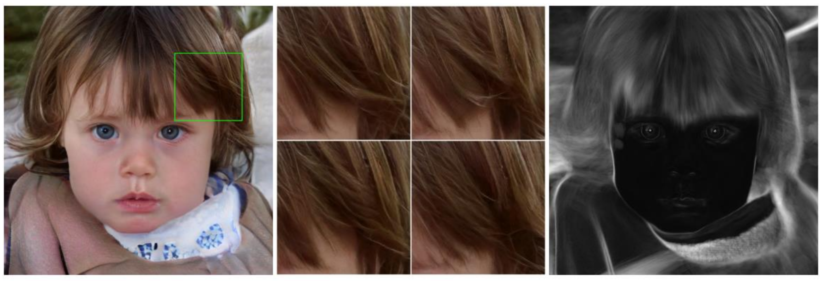
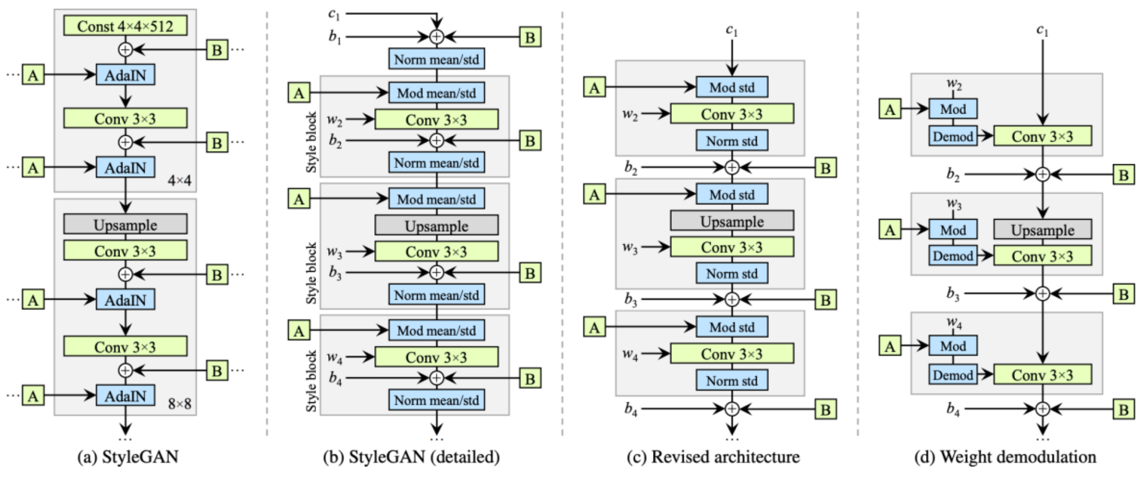


Figure 4. Examples of stochastic variation. (a) Two generated images. (b) Zoom-in with different realizations of input noise. While the overall appearance is almost identical, individual hairs are placed very differently. (c) Standard deviation of each pixel over 100 different realizations, highlighting which parts of the images are affected by the noise. The main areas are the hair, silhouettes, and parts of background, but there is also interesting stochastic variation in the eye reflections. Global aspects such as identity and pose are unaffected by stochastic variation.

StyleGAN is known for generating high-quality images, but it has some issues addressed in StyleGAN2.

3.5.1.3. StyleGAN 2 & StyleGAN 1



Blob-like Artifacts:

- Problem: In StyleGAN, there are common blob-like artifacts in the images, which are believed to come from the normalization process.

- Solution: StyleGAN2 introduces a new architecture that eliminates these blob-like artifacts by modifying how normalization is handled.

Location Preference Artifact:

- Problem: The original **StyleGAN** architecture, which uses progressive growing, tends to show a strong location preference. When changing the latent code w to adjust features like pose, the images often look unrealistic despite their high quality.

- Solution: StyleGAN2 avoids progressive growing and instead uses a skip generator and a residual discriminator. This approach helps produce more realistic images without the location preference issue.

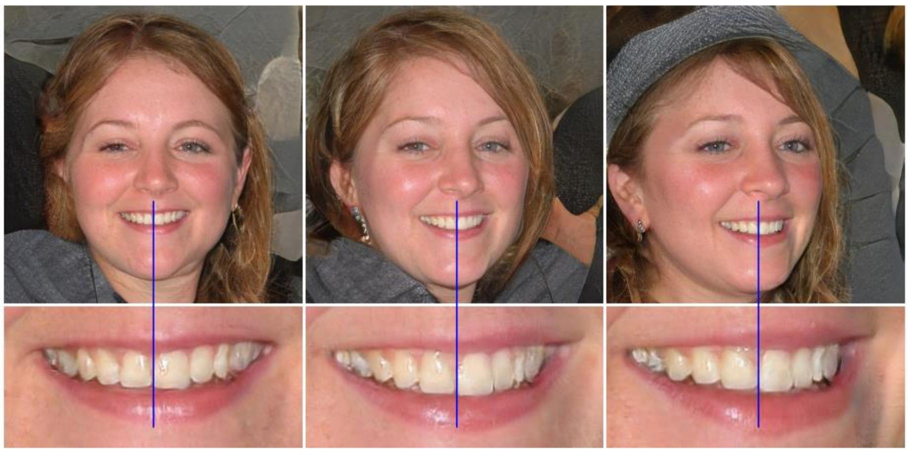


Figure 6. Progressive growing leads to "phase" artifacts. In this example the teeth do not follow the pose but stay aligned to the camera, as indicated by the blue line.

These improvements in StyleGAN2 address key issues from the original StyleGAN, resulting in better image quality and more realistic outputs.

3.5.1.4. StyleGAN 2 & StyleGAN 1

The authors of StyleGAN2 discovered that the synthesis network's **reliance on absolute pixel** coordinates caused an aliasing effect. This issue means that when the latent code w is interpolated, textures in the generated images appear fixed in place, while only high-level attributes like face pose or expression change. This results in an artificial look in animations.

StyleGAN3 addresses this problem by **redesigning the architecture to remove this unhealthy dependence** on absolute pixel coordinates. The improvements in StyleGAN3 can be seen in animations where the textures move naturally along with the high-level attributes, resulting in more realistic and coherent changes.

3.5.1.5. StyleGAN kind use cases

StyleGAN's ability to produce **photorealistic images** has enabled various applications, including image editing, privacy preservation, and creative exploration.

In **image editing**, it excels in tasks like image inpainting and style transfer. For privacy-preserving applications, StyleGAN generates synthetic data to replace sensitive information and anonymizes images by altering identifiable features.

In creative fields, it generates diverse fashion designs and creates realistic virtual environments for gaming and education, exemplified by tools like **Stylenerf**, which offers style-based, 3D-aware high-resolution image synthesis.

3.6. CycleGAN Variants

CycleGAN, introduced by Zhu et al. in 2017, is a framework for image-to-image translation tasks that **doesn't** require paired examples. Traditional methods rely on large datasets of paired images, which can be difficult or impossible to obtain.

CycleGAN addresses this by enabling image translation **using unpaired datasets**, making it a significant advancement in computer vision and machine learning.

3.6.1. What Is Unpaired Image-to-Image Translation?

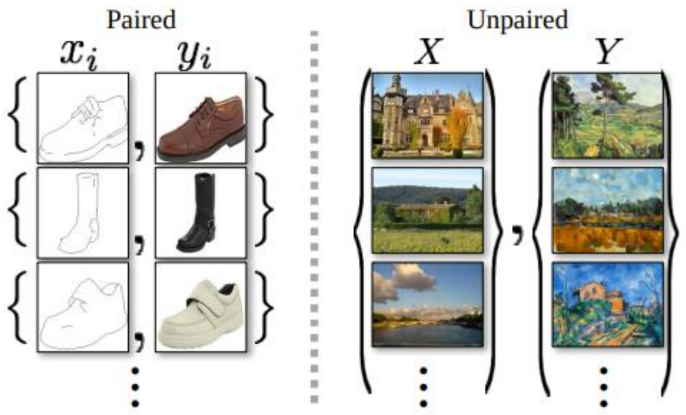


Figure 2: *Paired* training data (left) consists of training examples $\{x_i, y_i\}_{i=1}^N$, where the correspondence between x_i and y_i exists [22]. We instead consider *unpaired* training data (right), consisting of a source set $\{x_i\}_{i=1}^N$ ($x_i \in X$) and a target set $\{y_j\}_{j=1}^M$ ($y_j \in Y$), with no information provided as to which x_i matches which y_j.

In **unpaired image-to-image** translation, **datasets lack direct**, one-to-one image pairs. Instead, you have two distinct sets of images representing different styles or domains, such as realistic photographs and artworks, or different seasons like winter and summer.

The goal is for the model to learn the stylistic elements of each set and transform images from one domain to the other. This process works both ways, allowing for mutual transformation between the domains. This method is especially useful when exact image pairs are unavailable or difficult to obtain.

3.6.2. CycleGan components

CycleGAN employs two GANs (Generative Adversarial Networks), one for translating from the first set to the second (e.g., zebra to horse) and another for the reverse process (horse to zebra). This dual structure ensures realism (via the adversarial process) and content preservation (via cycle consistency).

3.6.2.1. Dual GAN Structure

CycleGAN uses two GANs to handle image translation between two domains, such as converting zebra images to horse images and vice versa.

- **PatchGAN Discriminators:** These discriminators evaluate patches of an image instead of the whole image, ensuring detailed and localized realism.
- **Generator Architecture:** The generators are based on U-Net and DCGAN architectures, involving encoding (downsampling), decoding (upsampling), and convolutional layers. They include residual connections to support deeper transformations and maintain identity functions.

3.6.2.2. Cycle Consistency Loss

Cycle consistency loss ensures that if you transform an image from one style to another and back again, it should closely resemble the original. This involves two stages:

1. **First Stage:** Transform the original image (e.g., sad face) to the target style (e.g., hugging face).
2. **Second Stage:** Transform the target style image back to the original style.

The loss is minimized by reducing the pixel difference between the original and the twice-transformed image. This loss is combined with the adversarial loss to form a comprehensive loss function for optimizing both generators.

3.6.2.3. Least-Square Loss

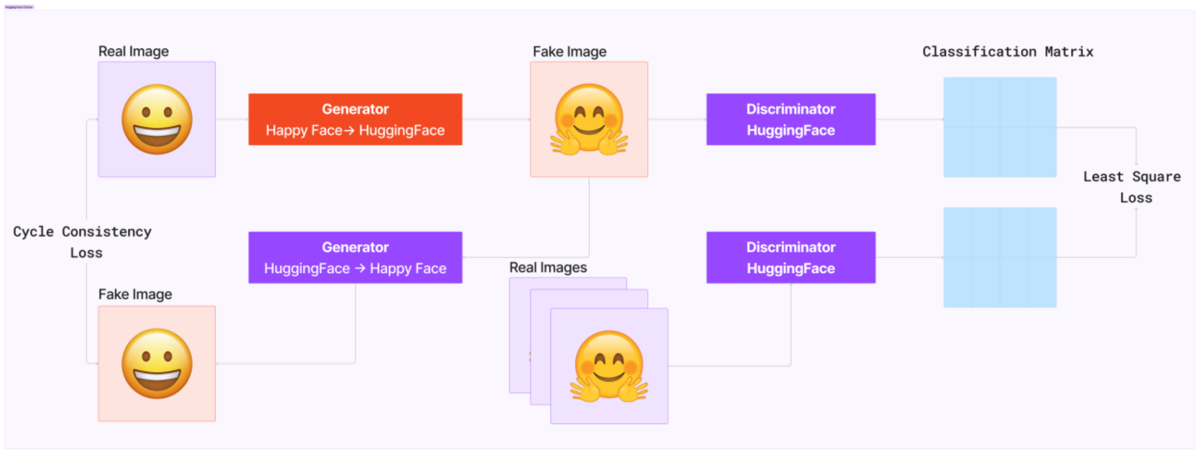
Least-square loss minimizes the squared differences between predicted values and actual labels (real or fake). For the **discriminator**, it reduces the squared distance between its predictions and **actual labels.** For the generator, it **aims to make its fake outputs appear real**.

This approach addresses issues like vanishing gradients and mode collapse seen with binary cross-entropy loss.

3.6.2.4. Identity Loss

Identity loss helps **preserve colors and finer details in generated images**. If an image from the target domain (e.g., a horse image) is input into the generator meant for transforming into the same domain (e.g., zebra-to-horse), the output should ideally be unchanged. The loss is calculated as the pixel distance between the input and the output image. This loss is added alongside adversarial and cycle consistency losses and is adjusted using a weighting factor (lambda term).

Summary: CycleGAN uses two GANs with PatchGAN discriminators and U-Net/DCGAN-based generators for image translation between unpaired domains. It employs **cycle consistency loss** to ensure realistic and consistent transformations**, least-square loss** for stability and accuracy, and optional **identity loss** to preserve image details. These combined techniques enable effective and realistic image-to-image translation without the need for paired datasets.



This figure shows the combined GAN architecture functionality for both GANs. These GANs are linked by **cycle consistency**, forming a cycle. For real images, the classification matrix contains ones. For fake images, it contains zeros. In summary, **CycleGAN** intricately combines two GANs with various loss functions, including **adversarial, cycle consistency, and optional identity loss**, to effectively transfer styles between two domains while preserving the essential characteristics of the input images.

3.7. Diffusion Models

Diffusion models are emerging in computer vision known for their ability to generate high-quality images.

These generative models work in two key stages: forward diffusion stage and the reverse diffusion stage.

- In the forward stage, the model gradually **adds noise** to the input data, effectively distorting it over time.
- In the reverse stage, the model learns to reverse this process, step by step, removing the noise to **reconstruct the original data.**

This iterative process of adding and then removing noise enables the generation of realistic and detailed images. These generative models have excelled in the field of generative modeling, especially with models like Imagen and **Latent Diffusion Models (LDMs).**

In the **forward Stage, Gaussian noise is added gradually** to the training images, eventually **turning** them **into completely noise, unrecognizable images.** Through this process, the model learns to remove the noise step-by-step, hence it is capable of turning any Gaussian noisy image into a new diverse image **(can also be conditioned based on text prompts).**

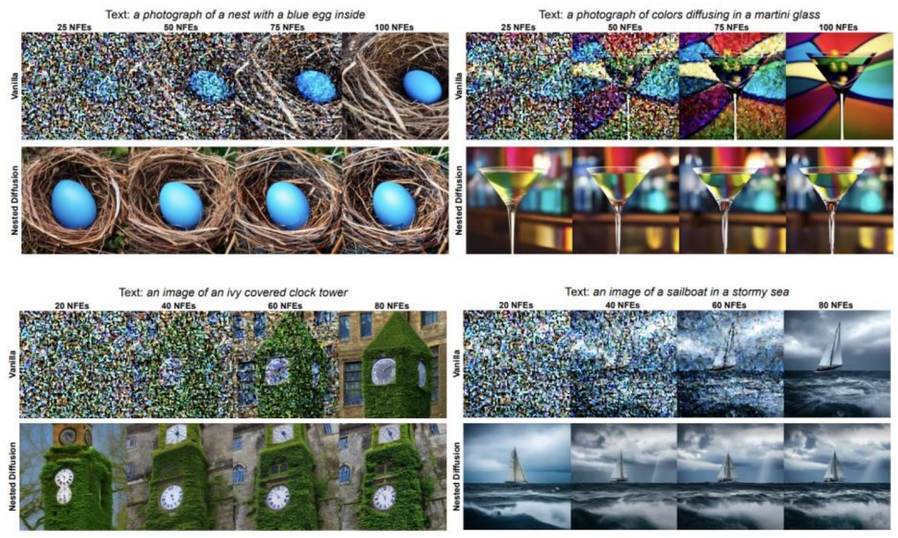


Figure 1. Results of intermediate predictions of Stable Diffusion from a reverse diffusion process of 100 steps (top) and 80 steps (bottom).

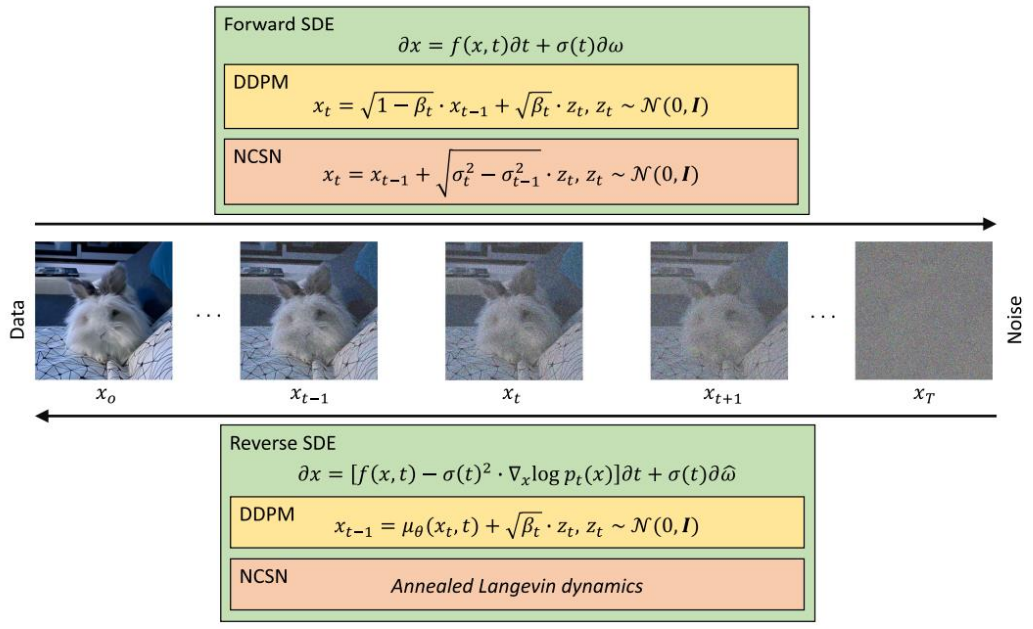
3.7.1. Variants of Diffusion models

There are some major Diffusion models frameworks:

- *Denoising diffusion probabilistic models (DDPMs):* DDPMs are a foundational framework in diffusion models. DDPMs can be thought of as a specific type of generative model that uses latent variables to estimate the probability distribution of data, similar to how Variational Autoencoders (VAEs) work. In this analogy, **the two stages** of the diffusion process in DDPMs correspond to the encoding and decoding processes in VAEs.
  This reverse process is **probabilistic**, meaning the model predicts the probability distribution of the denoised image at each step, rather than a single deterministic output.

- Noise conditioned score networks (NCSNs): NCSNs are another type of diffusion model that leverages **score matching**, which involves **estimating the gradient (or score)** of the data distribution. It estimates the score function (defined as the gradient of the log density) of the perturbed data distribution at different noise levels.
  Instead of adding noise progressively like DDPMs, NCSNs work by **training a neural network to predict the score of noisy images**—essentially, how to move the image towards higher likelihood under the data distribution. The model then samples new images by following these score estimates, which guide the noise removal process. This approach is more focused on learning the structure of the data directly from noisy samples.

- Stochastic differential equations (SDEs): It represents an alternative way to model diffusion. Modeling diffusion via forward and reverse SDEs leads to efficient generation strategies as well as strong theoretical results. This can be viewed as a **generalization over DDPMs and NCSNs.**
  They describe the process of adding and removing noise through differential equations that govern the random perturbations in the data. In the context of diffusion models, SDEs are used to model the evolution of data as it transitions from the original image to a completely noisy state and then back to a clean image. This continuous perspective allows for more flexible and theoretically grounded methods for sampling and denoising.


Forward SDE
$\partial x = f(x,t)\partial t + \sigma(t)\partial\omega$
DDPM
$x_t = \sqrt{1-\beta_t}\cdot x_{t-1} + \sqrt{\beta_t}\cdot z_t,\ z_t \sim \mathcal{N}(0,\boldsymbol{I})$
NCSN
$x_t = x_{t-1} + \sqrt{\sigma_t^2 - \sigma_{t-1}^2}\cdot z_t,\ z_t \sim \mathcal{N}(0,\boldsymbol{I})$
Data
Noise
x_o
x_{t-1}
x_t
x_{t+1}
x_T
Reverse SDE
$\partial x = [f(x,t) - \sigma(t)^2\cdot\nabla_x \log p_t(x)]\partial t + \sigma(t)\partial\hat{\omega}$
DDPM
$x_{t-1} = \mu_\theta(x_t,t) + \sqrt{\beta_t}\cdot z_t,\ z_t \sim \mathcal{N}(0,\boldsymbol{I})$
NCSN
Annealed Langevin dynamics

3.7.2. Diffusion models Vs GANs

Before the recent rise of diffusion models, GANs were widely regarded as the leading generative models in terms of the quality of the images they produced. However, **GANs** are **challenging to train** due to their **adversarial setup**, where a generator and discriminator compete against each other. This often leads to a problem known as **mode collapse.**

For instance, in the case of a model which task is to generate images of cats and dogs, which represent two different modes. If mode collapse occurs, the generator might only produce realistic images of either cats or dogs, but not both. This happens when the **discriminator,** which helps guide the generator, **gets stuck in a local minimum**, **consistently misclassifying one type of image** (either cat or dog) as fake.

In contrast, **diffusion models** offer a **more stable training** process and generate a **greater variety of images** because they are **based on likelihood, rather than an adversarial objective**.

One disadvantage is that **diffusion models** tend to be **computationally intensive** and require longer inference times compared to GANs due to the reverse step-by-step reverse process.

The **main drawback of diffusion models** is their **slow inference time**, requiring multiple steps to generate a single image. **Latent Consistency Models (LCMs)** have been proposed to speed up this process, allowing faster inference on pre-trained Latent Diffusion Models like **Stable Diffusion.** Besides that, **DDIM sampling** was proposed.

However, despite progress, GANs still outperform diffusion models in speed. Additionally, diffusion models using CLIP embeddings for text-to-image generation often struggle to render readable text, as CLIP embeddings lack detailed information about spelling, leading to issues with text accuracy in generated images.

3.7.3. Diffusion models' use cases.

Diffusion models have a wide range of applications, from **generating images based on text prompts (text-to-image)** to **enhancing image resolution, inpainting** damaged areas, and **editing images** while preserving their visual identity.

They are also used for tasks like **image-to-image** translation, such as changing backgrounds or attributes of a scene.

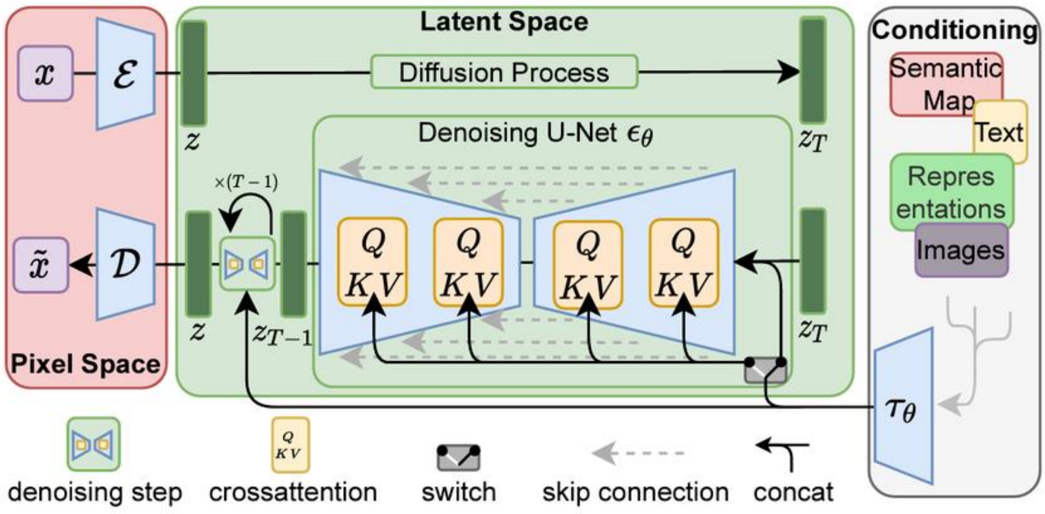
Beyond image manipulation, the **learned latent representations** from diffusion models can be leveraged for tasks like **image segmentation, classification, and anomaly detection.**

3.8. Stable Diffusion

Stable Diffusion is a type of Latent Diffusion Model (LDM) that gained significant attention for its ability to **generate high-quality images from text prompts**. Unlike traditional diffusion models, which operate directly on image pixels, **Stable Diffusion works in a lower-dimensional latent space**, making the process **more efficient** and allowing for faster image generation.

This approach involves **first mapping the image to a latent space,** applying the diffusion process within this space, **and then reconstructing the final image.**

Stable Diffusion is particularly popular because it balances high image quality with relatively fast inference times compared to other diffusion models. However, like other diffusion models, it **still requires multiple steps to generate an image**, which can be slower than alternative methods like GANs.



Stable Diffusion is a powerful model for generating images from text prompts. Key components of the Stable Diffusion process include:

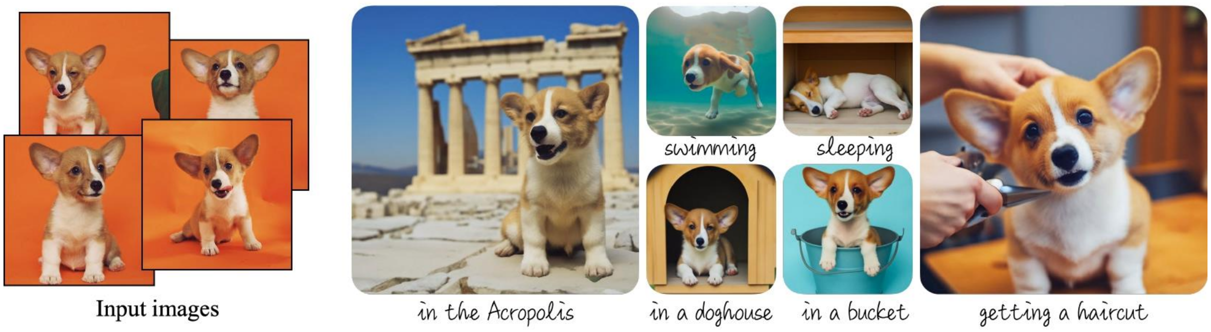
- **Image Compression:** To handle large images efficiently, Stable Diffusion **uses a Variational Auto-Encoder (VAE) to compress images** into a smaller, more manageable latent space. This compression is crucial because processing high-resolution images directly would require exponentially more computing power due to the quadratic increase in calculations with image size.

- **Text and Image Fusion:** For text-to-image generation, Stable Diffusion utilizes a **pre-trained transformer model known as CLIP**. CLIP encodes text prompts into high-dimensional vectors, which guide the diffusion process. During inference, a **text prompt is converted into a fixed-size vector representation,** which is **then used to condition** the model's denoising process.
- **Inductive Biases with U-Net and Cross-Attention:** To generate new and **diverse images,** Stable Diffusion employs a U-Net architecture with cross-attention mechanisms. The U-Net predicts the denoised image from noisy input, while cross-attention layers enable the model to incorporate relevant information from the text prompt at various spatial locations in the image.

3.9. Control over Diffusion Models

Although diffusion models and GANs can generate many unique images, they **can't always generate what you need exactly.** Some techniques can be used to personalize a model with just a few examples, avoiding the fine-tuning approach which usually requires a lot of data and computation.

Dreambooth is a technique developed by Google Research that allows for the customization of diffusion models with just a few example images. Dreambooth fine-tunes a model by associating a unique keyword with a few images of a specific subject or style. This allows the model to generate new images of the subject in various poses and backgrounds.



Another efficient method for customizing models is **Low Rank Adaptation (LoRA),** developed by Microsoft. LoRA improves the process of fine-tuning by focusing on updating only a small part of the model. It achieves this by breaking down the weight updates into two smaller, low-rank matrices while keeping the rest of the model fixed. This approach is more resource-efficient and faster, making it a popular choice for model adaptation.

3.9.1. Guided Diffusion via ControlNet

Diffusion models can be guided in various ways to produce specific outputs, such as using text **prompts, negative prompts, guidance scales, or inpainting.**

A notable method for enhancing guidance is **ControlNet**, introduced by Stanford University. ControlNet allows for more precise guidance by using an image that provides detailed information, like depth, pose, or edges, to steer the diffusion process. This helps achieve more consistent and controlled outputs from diffusion models, which often struggle with maintaining coherence.

ControlNet can be used in both text-to-image and image-to-image tasks. For example, if you input an image with edge information, ControlNet helps the model generate new images that follow the same shape as the input but with different colors. This makes the generated images more aligned with the provided guidance, ensuring the output closely matches the desired features.

Additional refs:

- *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*
- *FID*
- *SSIM*
- *Improved Techniques for Training GANs*
- *CLIPScore: A Reference-free Evaluation Metric for Image Captioning*
- *The Role of ImageNet Classes in Fréchet Inception Distance*
- *Lilian Weng's Awesome Blog on Autoencoders*
- *Generative models under a microscope: Comparing VAEs, GANs, and Flow-Based Models*
- *Autoencoders, Variational Autoencoders (VAE) and β-VAE*
- *Lilian Weng's Awesome Blog on GANs*
- *GAN — What is Generative Adversarial Networks*
- *What are the fundamental differences between VAE and GAN for image generation?*
- *Issues with GAN and VAE models*
- *VAE Vs. GAN For Image Generation*
- *Diffusion Models vs. GANs vs. VAEs: Comparison of Deep Generative Models*
- *StyleGAN3*
- *PatchGAN*
- *U-Net*
- *DCGAN*
- *Imagen*
- *Latent Diffusion Models*
- *Diffusers installation*