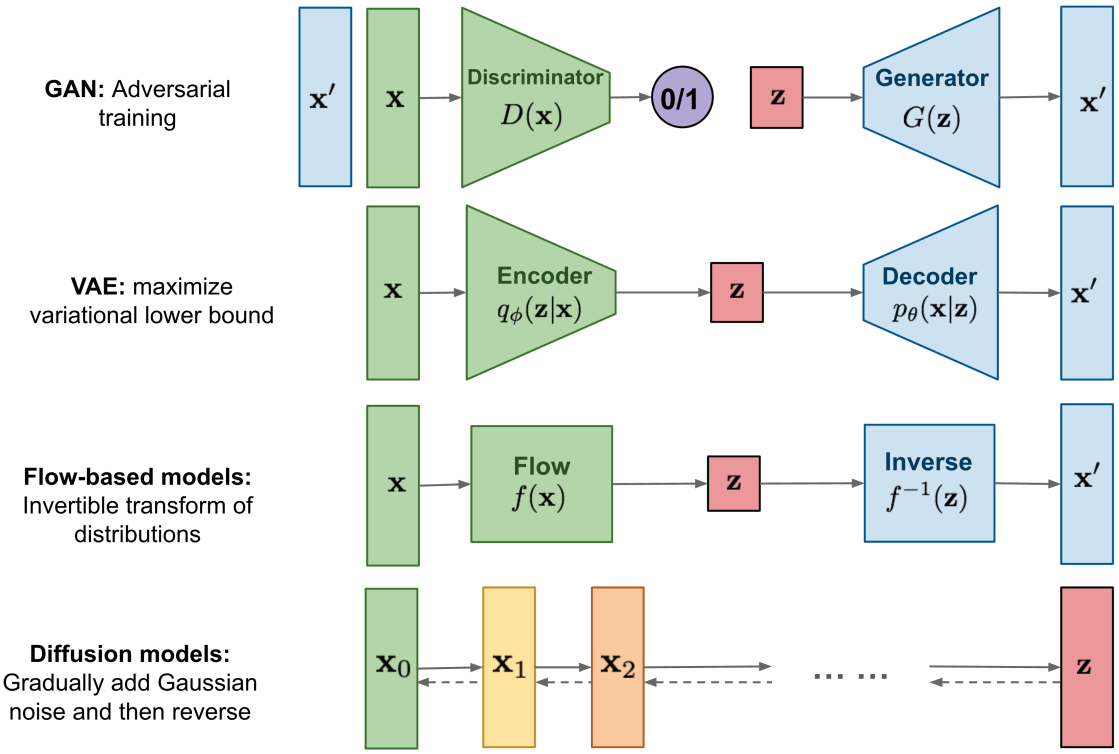
Diffusion Models Beat GANs on Image Synthesis

https://github.com/GaParmar/img2img-turbo?tab=readme-ov-file
https://www.media.io/lab/ai-face-editor/
https://pixlr.com/face-swap/
https://pixlr.com/blog/how-ai-is-transforming-the-creative-industries/
https://gencraft.com/generate
https://github.com/CompVis/latent-diffusion

Generativní sítě: Cesta k difuznímu modelu

Vysvětlení stable diffusion: https://aws.amazon.com/what-is/stable-diffusion/

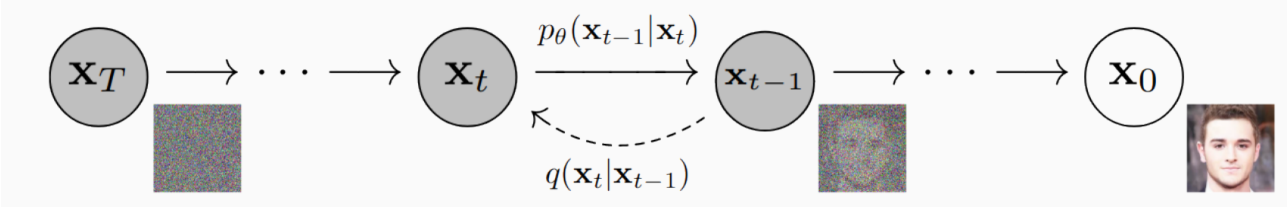


- **Generative Adversarial Networks (GANs):** GANs involve two neural networks, a generator and a discriminator, which are trained simultaneously. The generator produces samples aimed at passing for real data, while the discriminator tries to distinguish between actual and generated data.
- **Variational Autoencoders (VAEs):** VAEs are based on a probabilistic framework that learns a latent space representation of the input data. They reconstruct the input data by learning the distribution parameters.
- **Flow Based Models:** A flow-based generative model is a type of machine learning model designed to create complex data distributions from simpler ones. It does this by applying a series of transformations, known as normalizing flows, which are mathematically invertible. This approach allows for both direct sampling of data and exact likelihood computation,

making it a powerful tool for tasks that require understanding and generating high-dimensional data.

- **Diffusion Models:** Diffusion models are a type of generative model that simulates the gradual process of diffusion, or the spreading of particles from regions of higher concentration to lower concentration, to generate new data samples. This process involves a forward diffusion phase that gradually adds noise to the data, transforming it into a Gaussian distribution, followed by a reverse diffusion phase that carefully removes noise to construct new data samples.

https://www.ionio.ai/blog/beginners-guide-to-diffusion-models-and-generative-ai

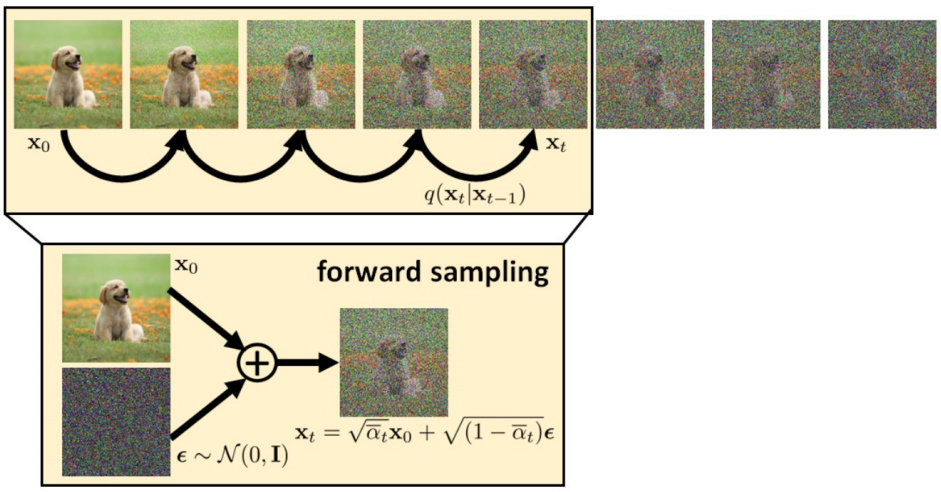


$$q(\mathrm{x}_t \mid \mathrm{x}_{t-1}) = \mathcal{N}\left(\sqrt{1-\beta_t}\mathrm{x}_{t-1}, \beta_t \boldsymbol{I}\right)$$

$$q\left(\mathrm{x}_t \mid \mathrm{x}_0\right) = \mathcal{N}\left(\sqrt{\prod_{i=1}^{t}(1-\beta_i)}\mathrm{x}_0, \left(1-\prod_{i=1}^{t}(1-\beta_i)\right)\boldsymbol{I}\right)$$

A nakonec:

$$q\left(\mathrm{x}_T \mid \mathrm{x}_0\right) \approx \mathcal{N}(0, \boldsymbol{I})$$



Stanley Chan: Tutorial on Diffusion Models for Imaging and Vision, https://arxiv.org/pdf/2403.18103

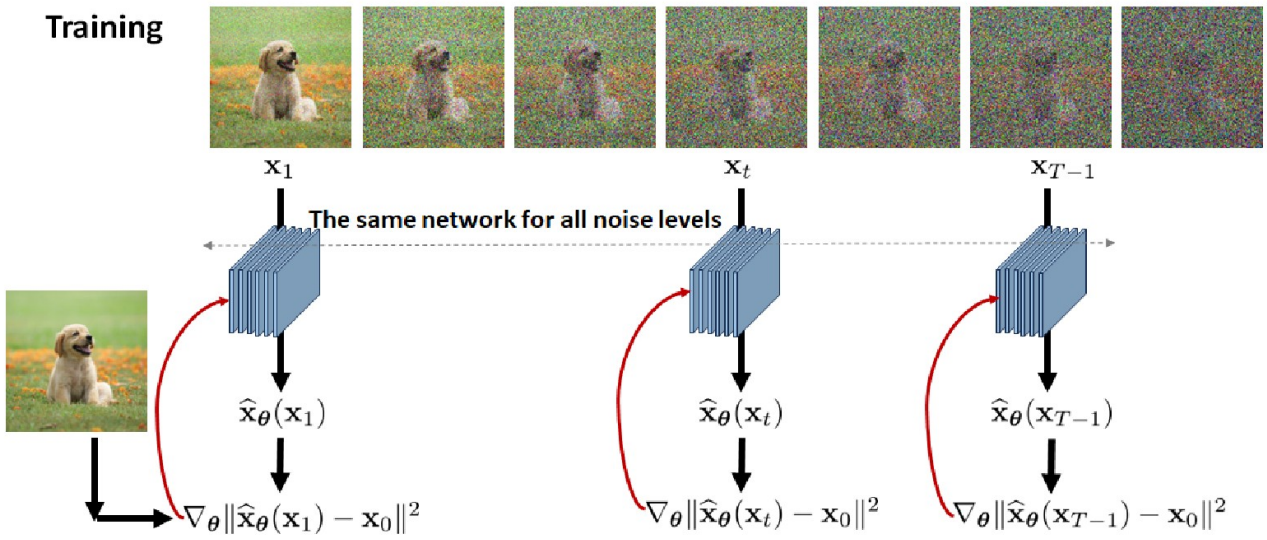
We want to learn the reverse processing, knowing that pθ(xt |xt+1) is Gaussian, but of unknown mean and variance:

The parameters of the Gaussian are predicted by a neural network from xt+1. Ho et al. (Denoising Diffusion Probabilistic Models, 2020) only predict the mean with a fixed variance schedule.

$$p_\theta(x_t \mid x_{t+1}) = \mathcal{N}(\mu_\theta(x_{t+1}, t), \sigma_t^2 \boldsymbol{I})$$

$$p_\theta(x_t \mid x_{t+1}) = \mathcal{N}(\mu_\theta(x_{t+1}, t), \Sigma_\theta(x_{t+1}, t))$$

Po delším odvozování odtud plyne následující postup.



Training Algorithm for DDPM. For every image $\mathbf{x}_0$ in your training dataset:

- Repeat the following steps until convergence.
- Pick a random time stamp $t \sim \text{Uniform}[1, T]$.
- Draw a sample $\mathbf{x}_t^{(m)} \sim \mathcal{N}(\mathbf{x}_t \,|\, \sqrt{\overline{\alpha}_t}\mathbf{x}_0,\ \ (1-\overline{\alpha}_t)\mathbf{I})$, i.e.,

$$\mathbf{x}_t^{(m)} = \overline{\alpha}_t \mathbf{x}_0 + \sqrt{(1-\overline{\alpha}_t)}\boldsymbol{\epsilon}_t^{(m)}, \qquad \boldsymbol{\epsilon}_t^{(m)} \sim \mathcal{N}(0, \mathbf{I}).$$

- Take gradient descent step on

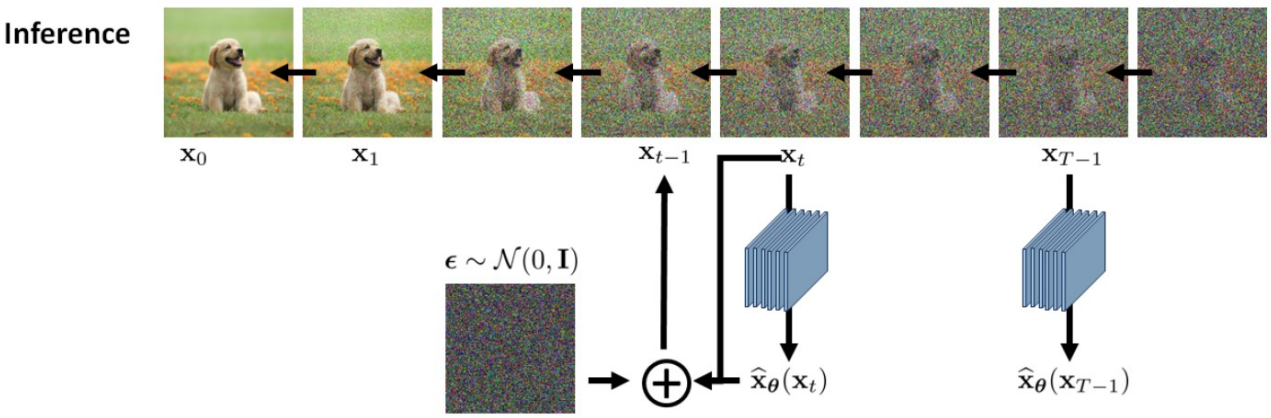
$$\nabla_{\boldsymbol{\theta}} \left\{ \frac{1}{M} \sum_{m=1}^{M} \left\| \widehat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t^{(m)}) - \mathbf{x}_0 \right\|^2 \right\}.$$

You can do this in batches, just like how you train any other neural networks. Note that, here, you are training **one** denoising network $\widehat{\mathbf{x}}_{\boldsymbol{\theta}}$ for **all** noisy conditions.

Inference of DDPM.

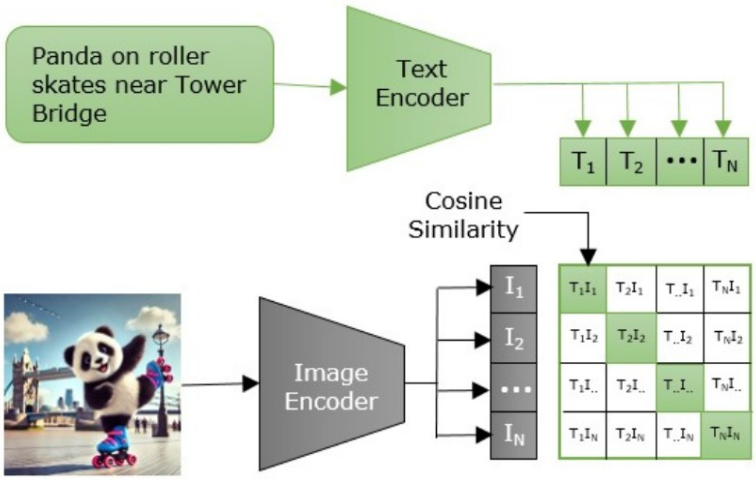
- You give us a white noise vector $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
- Repeat the following for $t = T, T-1, \ldots, 1$.
- We calculate $\widehat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t)$ using our trained denoiser.
- Update according to

$$\mathbf{x}_{t-1} = \frac{(1-\overline{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\overline{\alpha}_t}\mathbf{x}_t + \frac{(1-\alpha_t)\sqrt{\overline{\alpha}_{t-1}}}{1-\overline{\alpha}_t}\widehat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{x}_t) + \sigma_q(t)\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}).$$


Inference
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_{t-1}$
$\mathbf{x}_t$
$\mathbf{x}_{T-1}$
$\epsilon \sim \mathcal{N}(0, \mathbf{I})$
$\hat{\mathbf{x}}_{\theta}(\mathbf{x}_t)$
$\hat{\mathbf{x}}_{\theta}(\mathbf{x}_{T-1})$

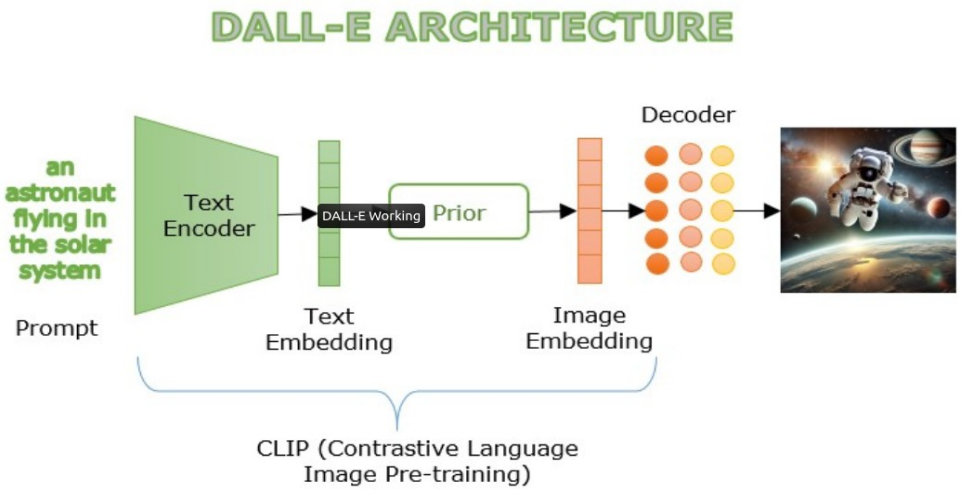
DALL-E - Architecture

CLIP architecture (Contrastive Language-Image Pretraining)



CLIP is a large language model developed by OpenAI exclusively for the functioning of the DALL-E model. It is trained on several images with associated captions to bridge the gap between textual description and images. As its name suggests, the "contrastive" model compares the given text prompt with the captions of the existing images in the dataset to check if the input matches with any image captions. Every image-caption pair is assigned a similarity score, and the pair with the highest similarity score is picked. To perform this task, the model relies on two components −

- **Text Encoder** − It converts the user's text prompt into text embedding, which are numerical values that are understood by DALL-E.
- **Image Encoder** − Similar to the text encoder, this component is used to convert images into image embedding.



The workflow of the model is described below −

- Once the textual description for an image is provided, it is given to **CLIP's text encoder**. The meaning of the prompt is understood using NLP, and then it is converted into a high-dimensional vector representation that captures semantic meaning. This vector representation is called text embedding.
- Next, the text embedding is then passed to **prior**, a type of generative model that can sample from a probability distribution to produce realistic images.

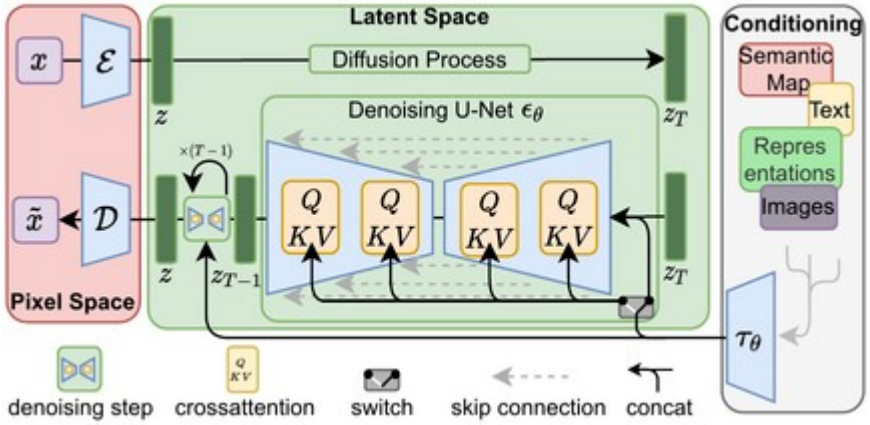
- In the final step, once the prior generated image embedding is passed through the **diffusion decoder**, which generates the final image.

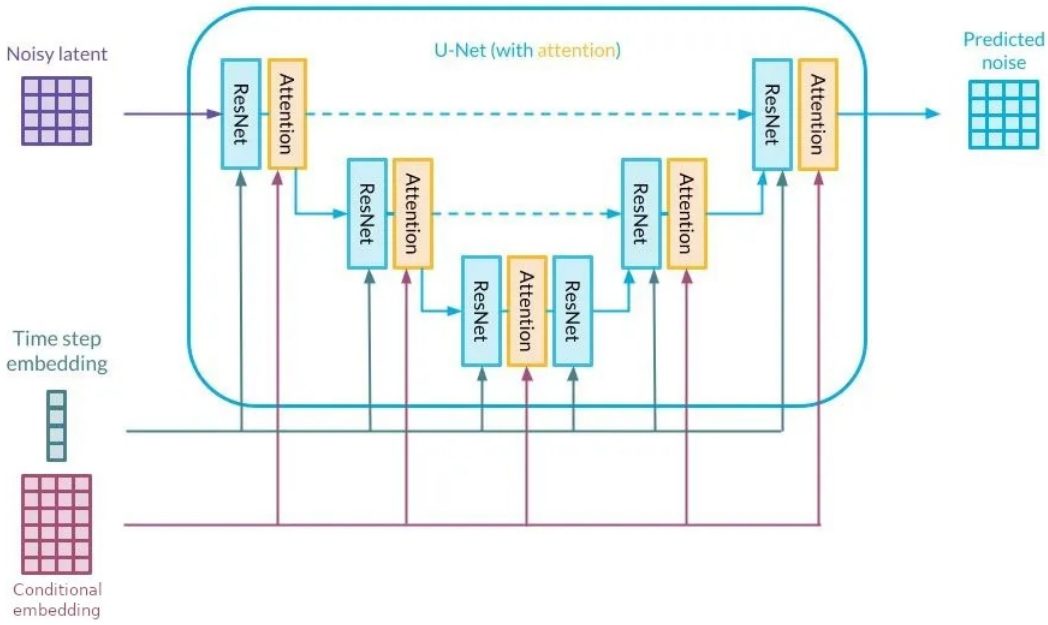
  Alex Nichol et al.:GLIDE: Towards Photorealistic Image Generation and Editing with text-Guided Diffusion Models.

Latent Diffusion Model

Rombach et al.: High-Resolution Image Synthesis with Latent Diffusion Models (2022)
https://github.com/CompVis/latent-diffusion

https://





$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \cdot V, \text{ with}$$

$$Q = W_Q^{(i)} \cdot \varphi_i(z_t),\ K = W_K^{(i)} \cdot \tau_\theta(y),\ V = W_V^{(i)} \cdot \tau_\theta(y).$$

Pozor na to, že attention přístpupů pro U-net je více. Nejstarší je následující
Ozan Oktay et al.: Attention U-Net: Learning Where to Look for the Pancreas (2019). Ten je ale jiný než předchozí.

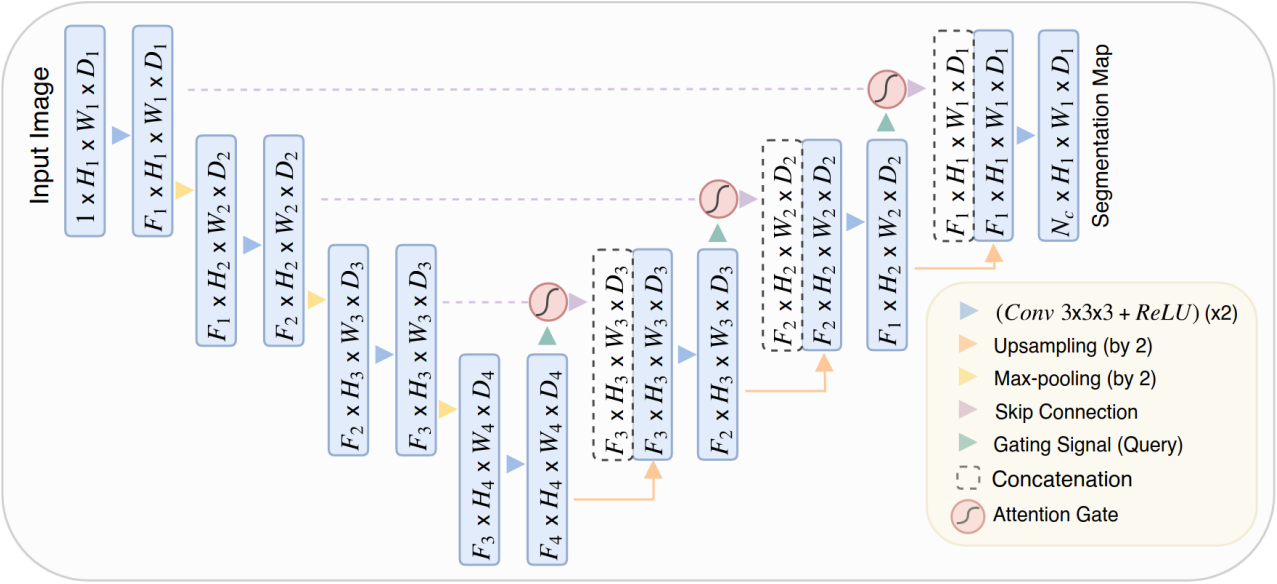


Figure 1: A block diagram of the proposed Attention U-Net segmentation model. Input image is progressively filtered and downsampled by factor of 2 at each scale in the encoding part of the network (e.g. $H_4 = H_1/8$). N_c denotes the number of classes. Attention gates (AGs) filter the features propagated through the skip connections. Schematic of the AGs is shown in Figure 2. Feature selectivity in AGs is achieved by use of contextual information (gating) extracted in coarser scales.

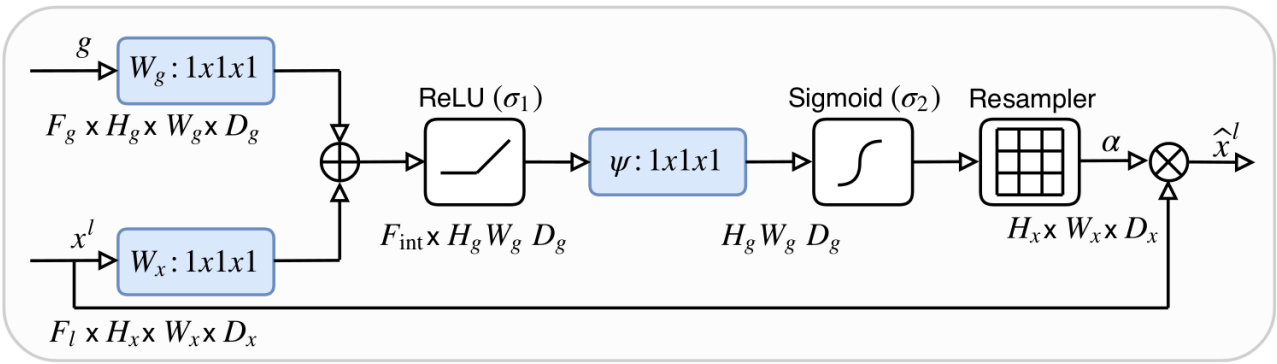


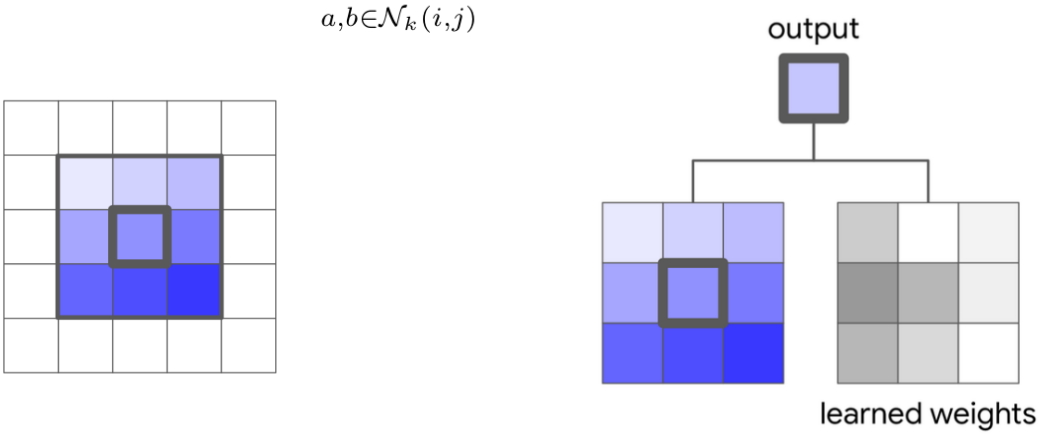
Figure 2: Schematic of the proposed additive attention gate (AG). Input features (x^l) are scaled with attention coefficients (α) computed in AG. Spatial regions are selected by analysing both the activations and contextual information provided by the gating signal (g) which is collected from a coarser scale. Grid resampling of attention coefficients is done using trilinear interpolation.

Stand-Alone Self-Attention

Prajit Ramachandran: Stand-Alone Self-Attention in Vision Models (2019)

Convolution:

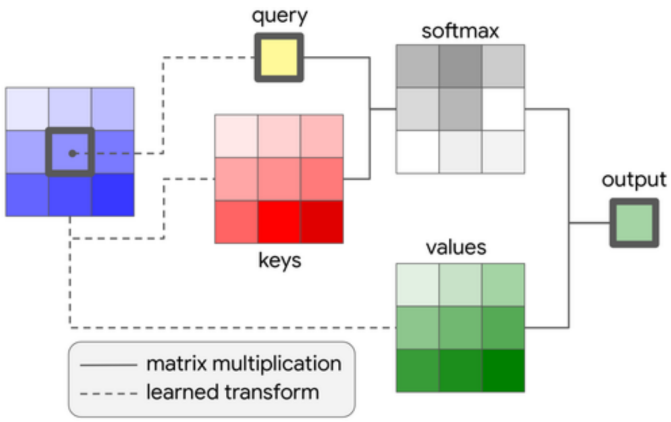
$$y_{ij} = \sum_{a,b \in \mathcal{N}_k(i,j)} W_{i-a,j-b}\, x_{ab}$$



Self-Attention

$$y_{ij} = \sum_{a,b \in \mathcal{N}_k(i,j)} \texttt{softmax}_{ab}\left(q_{ij}^{\top} k_{ab}\right) v_{ab}$$

where the *queries* $q_{ij} = W_Q x_{ij}$, *keys* $k_{ab} = W_K x_{ab}$, and *values* $v_{ab} = W_V x_{ab}$ are linear transformations of the pixel in position ij and the neighborhood pixels. $\texttt{softmax}_{ab}$ denotes a softmax applied to all logits computed in the neighborhood of ij. $W_Q, W_K, W_V \in \mathbb{R}^{d_{out} \times d_{in}}$ are all learned transforms.



In practice, multiple attention *heads* are used to learn multiple distinct representations of the input. It works by partitioning the pixel features x_{ij} depthwise into N groups $x_{ij}^n \in \mathbb{R}^{d_{in}/N}$, computing single-headed attention on each group separately as above with different transforms $W_Q^n, W_K^n, W_V^n \in \mathbb{R}^{d_{out}/N \times d_{in}/N}$ per head, and then concatenating the output representations into the final output $y_{ij} \in \mathbb{R}^{d_{out}}$.

The relative distance is factorized across dimensions, so each element $ab \in \mathcal{N}_k(i,j)$ receives two distances: a row offset $a-i$ and column offset $b-j$ (see Figure 4). The row and column offsets are associated with an embedding r_{a-i} and r_{b-j} respectively each with dimension $\frac{1}{2}d_{out}$. The row and column offset embeddings are concatenated to form $r_{a-i,b-j}$. This spatial-relative attention is now defined as

$$y_{ij} = \sum_{a,b \in \mathcal{N}_k(i,j)} \texttt{softmax}_{ab}\left(q_{ij}^\top k_{ab} + q_{ij}^\top r_{a-i,b-j}\right) v_{ab}$$

Thus, the logit measuring the similarity between the query and an element in $\mathcal{N}_k(i,j)$ is modulated both by the content of the element and the relative distance of the element from the query. Note that by infusing relative position information, self-attention also enjoys translation equivariance, similar to convolutions.

-1, -1	-1, 0	-1, 1	-1, 2
0, -1	0, 0	0, 1	0, 2
1, -1	1, 0	1, 1	1, 2
2, -1	2, 0	2, 1	2, 2

www.youtube.com/watch?v=hEJjg7VUA8g
https://towardsdatascience.com/a-detailed-explanation-of-the-attention-u-net-b371a5590831
https://github.com/LeeJunHyun/Image_Segmentation
https://reface.ai/unboring