Deep Networks Architectures

Introduction to Machine Learning – GIF-7015

Professor: Christian Gagné

Week 10

UNIVERSITÉ
LAVAL

10.1 Convolution and image processing

Convolution

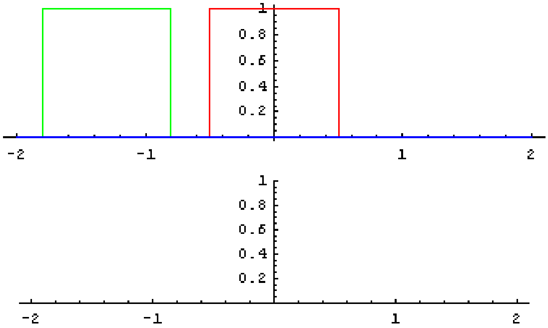
- Convolution: product of two functions on the same domain

$$f(x) * g(x) \equiv \int_{t=-\infty}^{\infty} f(x-t)\, g(t)\, \mathrm{d}t$$

- Discrete formulation

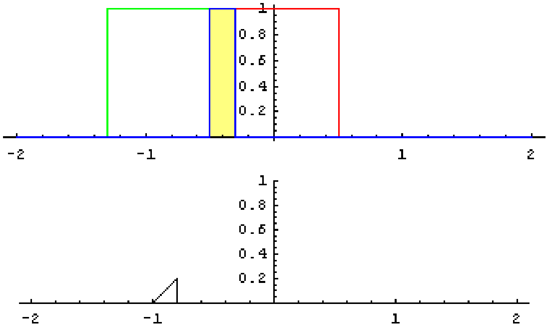
$$f(x) * g(x) \equiv \sum_{t=-\infty}^{\infty} f(x-t)\, g(t)$$

Convolution example



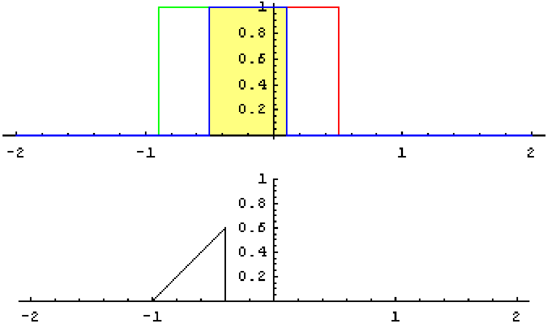
By Lautaro Carmona, CC-SA 4.0, `https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif`.

Convolution example



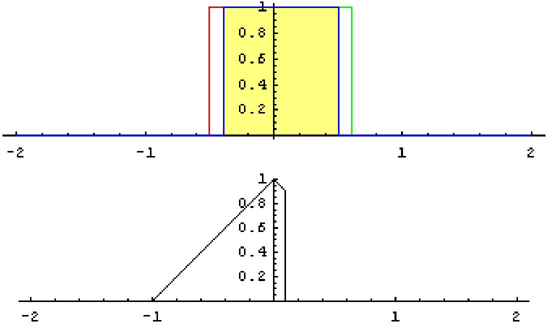
By Lautaro Carmona, CC-SA 4.0, `https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif`.

Convolution example



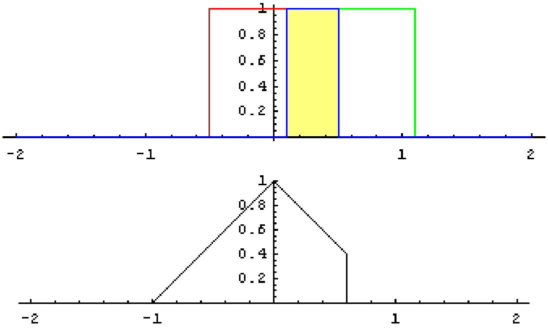
By Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Convolution example



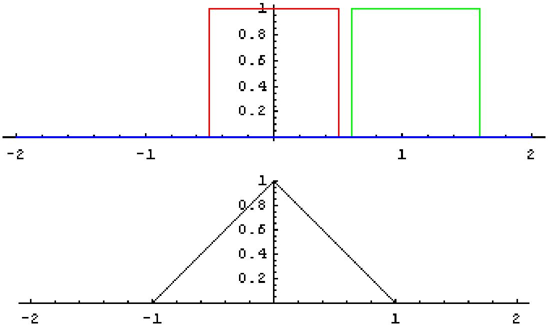
By Lautaro Carmona, CC-SA 4.0, `https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif`.

Convolution example



By Lautaro Carmona, CC-SA 4.0, https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif.

Convolution example



By Lautaro Carmona, CC-SA 4.0, `https://commons.wikimedia.org/wiki/File:Convolucion_Funcion_Pi.gif`.

Convolution and density estimation

- Off-center Dirac distribution

$$\delta(x-t) = \begin{cases} \infty & \text{if } x = t \\ 0 & \text{otherwise} \end{cases}, \quad \int_{x=-\infty}^{\infty} \delta(x-t)\,\mathrm{d}x = 1.$$

  - Convolution on off-center Diracs

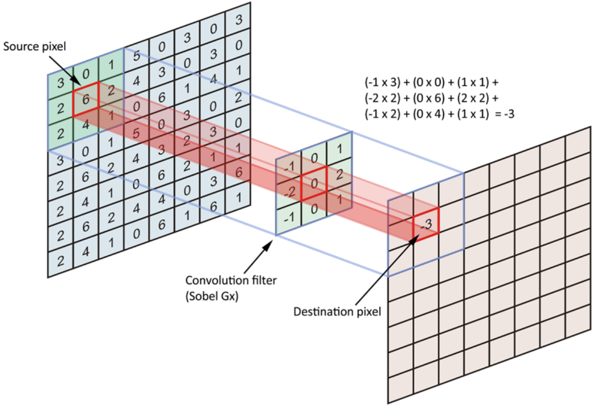
$$f(x) * \delta(x-u) = f(x-u)$$

- Kernel density estimation: kernel convolution with several Diracs centrered on the data

$$\hat{p}(x) = \frac{1}{Nh}\sum_{t=1}^{N} K\left(\frac{x-x^t}{h}\right) = \frac{1}{Nh}\sum_{t=1}^{N} K\left(\frac{x}{h}\right) * \delta(x-x^t)$$

Image processing

- 2D convolution is a building block for image processing

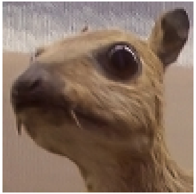


Source: `https://thigiacmaytinh.com/wp-content/uploads/2018/05/kernel.png`, accessed November 13, 2018.

Examples of filters

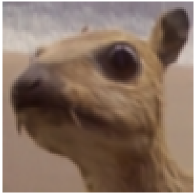
Identity (3×3):

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Gaussian blur:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

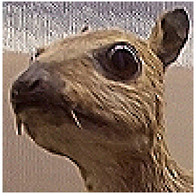


Edge detection:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Sharpen:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

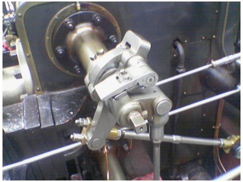


Sobel operator

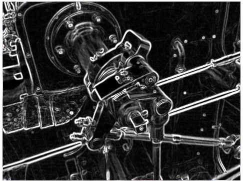
- Classic filter for edge detection
  - Compute local gradients of image intensity
  - Uses two convolutions to obtain the vertical gradient $\mathbf{G}_x$ and the horizontal gradient $\mathbf{G}_y$ of an image $\mathbf{A}$, the result is an image $\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2}$

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}, \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Original image:



Application of Sobel:

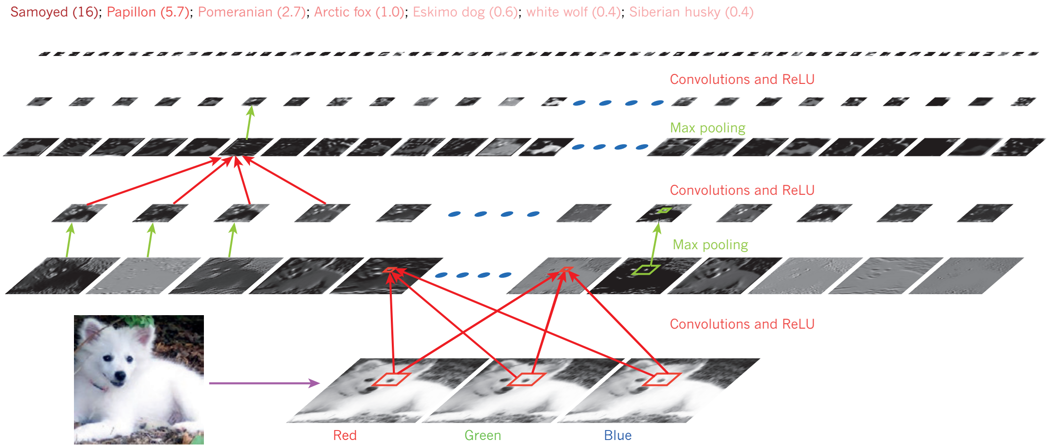


10.2 Convolutional neural networks

Convolutional neural networks

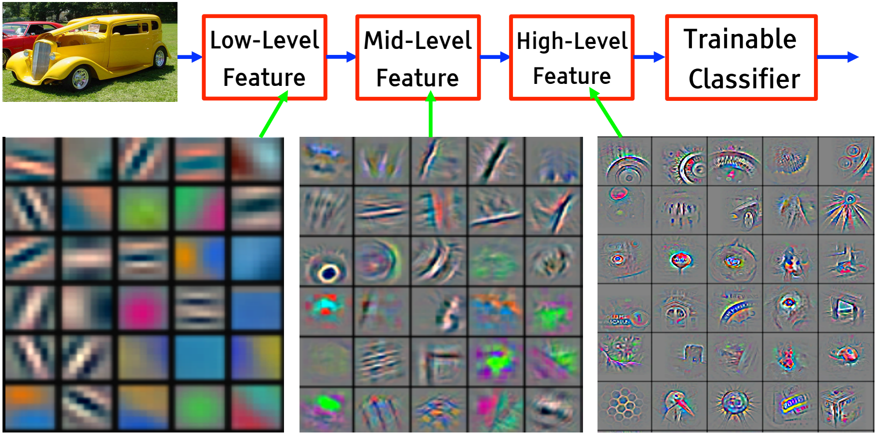
- Idea: neural networks with convolution operations
  - Learning the numerical values of convoluted filters
  - Define a network exploiting elements of the data structure
    - Sound or speech: temporal data (1D convolutions)
    - Image: spatial data (2D convolutions)
    - Video: spatiotemporal data (3D convolutions)
  - Sequence of convolution stages, filtering output of the previous layer
  - Allows for more compact modelling than fully connected networks and translation invariant
- Some components of a convolution network
  - Layer of convoluted filters on the different channels
  - Pooling: maximum (max pool) or average (avg pool) value in a certain convoluted window
  - Transfer functions: ReLU, etc.
  - Near output, fully connected layers (like with multi-layer perceptron)

Convolution network



From *Y. LeCun, Y. Bengio and G. Hinton, Deep Learning, Nature, vol. 521, 28 mai 2015.* Accessed online November 6, 2020 at `https://www.nature.com/articles/nature14539`.

Filters composition

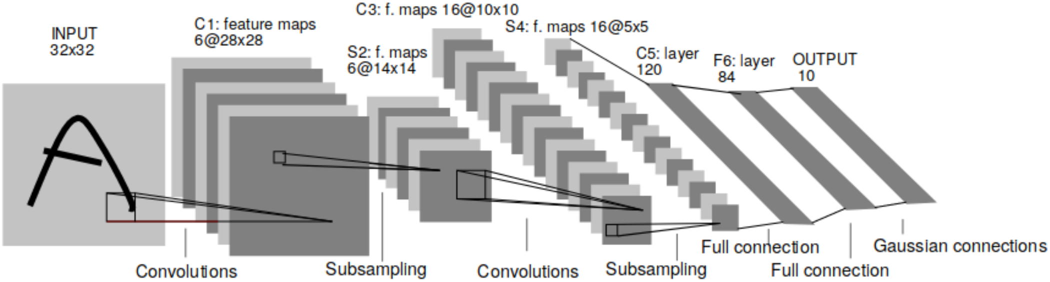


From *G. Hinton, Y. Bengio and Y. LeCun, Deep Learning NIPS'15 Tutorial, 2015.* Accessed online on November 6, 2020 at `https://nips.cc/Conferences/2015/Schedule?showEvent=4891`.

10.3 Examples of convolutional networks

LeNet5

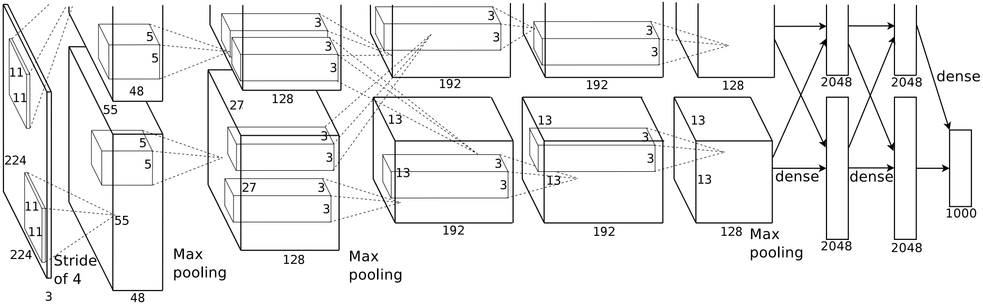
- LeNet5: classical convolutional network, proposed in the 1990s
  - 3 convolution layers, 2 average pooling layers, 2 fully connected layers
  - 60k parameters (from 10M to 100M with modern networks)



From *Y. LeCun, L. Bottou, Y. Bengio et P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86(11), 1998.* Accessed online on November 6, 2020, at `http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf`.

AlexNet

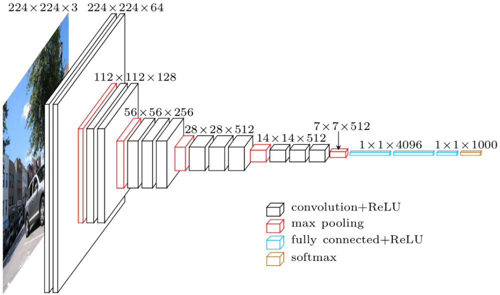
- AlexNet: network for object recognition
  - Winner of the ImageNet 2012 contest
  - Implemented for GPU Computing
  - Often used as a basic model for representation transfer
  - 8 convolution layers, some max pooling layers, 3 fully connected layers



From *A. Krizhevsky, I. Sutskever, and G. Hinton, Imagenet classification with deep convolutional neural networks. NIPS, 2012.* Accessed online November 6, 2020, at `https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

VGG

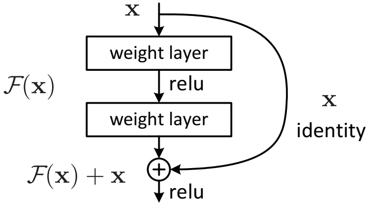
- VGGNet: greater depth with simplified topology
  - Winner of the ImageNet 2013 contest
  - Depth is critical for good performance
  - Similar to AlexNet, but with only 3×3 convolutions, 2×2 max pooling, 3 layers fully connected and 16 layers in total (VGG-16)



Source: `https://heuritech.files.wordpress.com/2016/02/vgg16.png`, accessed November 13, 2018.

ResNet

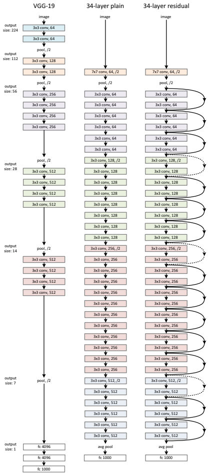
- Residual networks: allowing direct connections between non-adjacent layers (*skip links*)



From *K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition. CVPR, 2016.* Accessed online November 6, 2020, at `https://arxiv.org/abs/1512.03385`.

- Allows for much deeper and more efficient networks
  - Winner of ImageNet 2015 competition (3.57 % top 5 error)
  - Facilitates signal optimization and propagation across the network
  - Residual block must do better than a treatment directly on the previous block

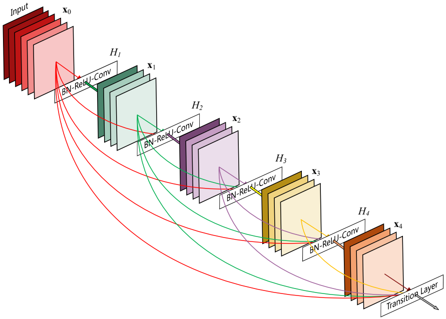
ResNet



From *K. He, X. Zhang, S. Ren, et J. Sun, Deep residual learning for image recognition. CVPR, 2016.* Accessed online November 6, 2020, at `https://arxiv.org/abs/1512.03385`.

DenseNet

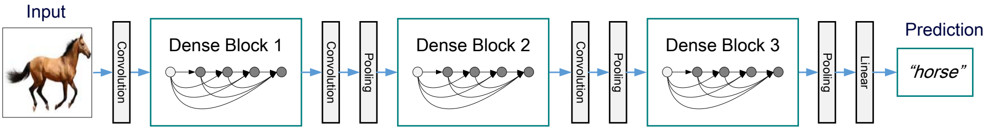
- Observation: convolution networks can be deeper and get better performance with close connections throughout the network at its input.
- DenseNet: connect each layer to all of the above layers
  - Network with L layers will have $L(L+1)/2$ direct connections between layers



From *G. Huang, Z. Liu, L. Van Der Maaten et K.Q. Weinberger, Densely Connected Convolutional Networks. CVPR, 2017.* Accessed online on November 6, 2020, at `https://arxiv.org/abs/1608.06993`.

DenseNet

- In practice, we create dense blocks separated by convolution and pooling layers



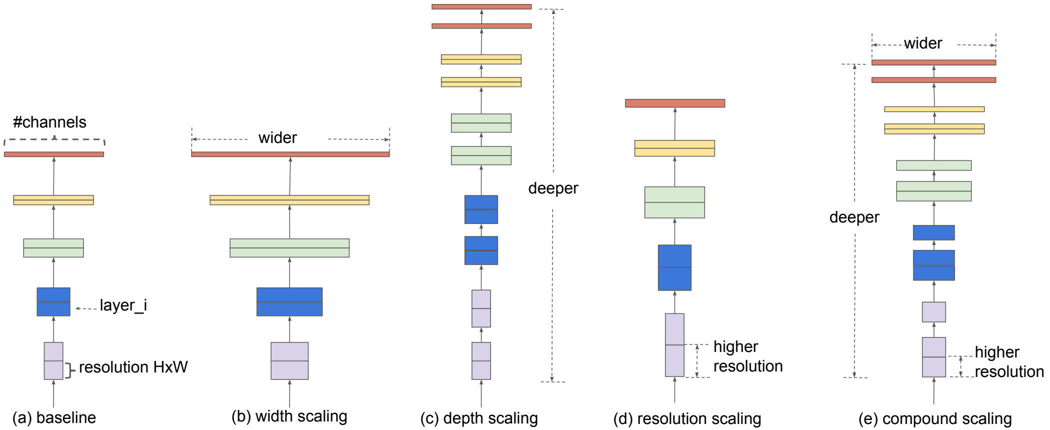
From *G. Huang, Z. Liu, L. Van Der Maaten et K.Q. Weinberger, Densely Connected Convolutional Networks. CVPR, 2017.* Accessed online on November 6, 2020, at `https://arxiv.org/abs/1608.06993`.

- Each layer in a dense block can be relatively narrow, i.e. can contain few neurons.

EfficientNet

- EfficientNet: optimal adjustment of convolution network size
  - How to adjust network architecture according to available resources?
- Idea: if the image resolution is higher, performance will be better, but the resources required (depth and width) are greater to properly capture image details.
- Proportional adjustment of depth, width and resolution according to ϕ factor
  - Depth: number of network layers, according to α^ϕ
  - Width: number of channels in each layer, according to β^ϕ
  - Resolution: input image resolution adjustment, according to γ^ϕ
  - Values of α, β and γ determined experimentally (grid search) for a network with doubled resources ($\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$)
- MobileNet V2-based architecture, with reverse bottleneck of residual connections

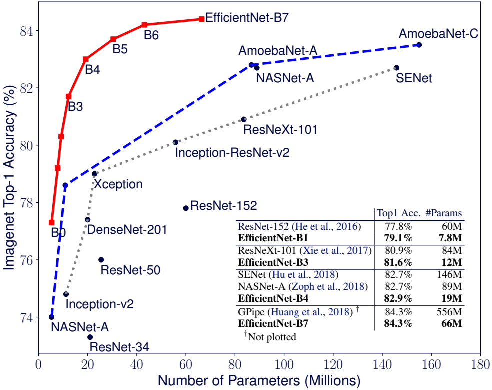
Size adjustment in EfficientNet



Taken from *M. Tan, Q.V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML, 2019.* Accessed online on October 29, 2023 at `https://arxiv.org/abs/1905.11946`.

EfficientNet performances

- For the same resources, EfficientNet offers superior performance
- Eight versions (EfficientNet-B0 to B7) have been proposed for different resource/performance trade-offs.
- Suitable for use in mobile devices and edge computing

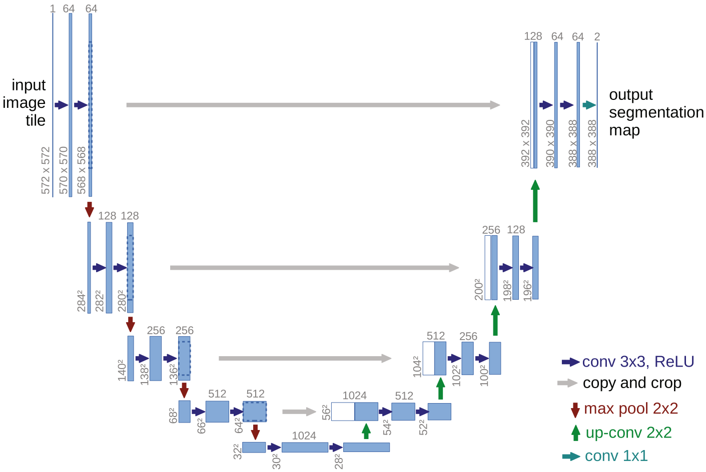


Taken from *M. Tan, Q.V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML, 2019.* Accessed online on October 29, 2023, at `https://arxiv.org/abs/1905.11946`.

U-Net

- Networks presented so far first proposed and tested for object recognition (classification)
  - Other possible tasks in vision: detection, tracking, etc.
- Segmentation: identify coherent regions of the image
  - Separate the different regions
  - Give a label to each region
- U-Net: network proposed for biomedical imaging
  - Fully convolutional network, gives an output image
  - Compression of information in a network environment, similar to an auto-encoder
  - Skip links allow to preserve spatial structure

U-Net



From *O. Ronneberger, P. Fischer, et T. Brox, U-net: Convolutional networks for biomedical image segmentation. MICCAI, 2015.* Accessed online on November 6, 2020 at `https://arxiv.org/abs/1505.04597`.

10.4 Images generation

Generation of examples

- Idea: generate input data based on a desired output
  - Generate a model of the data that can produce the output according to the neural network
- Approach: gradient descent on the input data

$$\Delta \mathbf{x} = -\eta \frac{\partial E(\mathbf{x}|\theta)}{\partial \mathbf{x}}$$

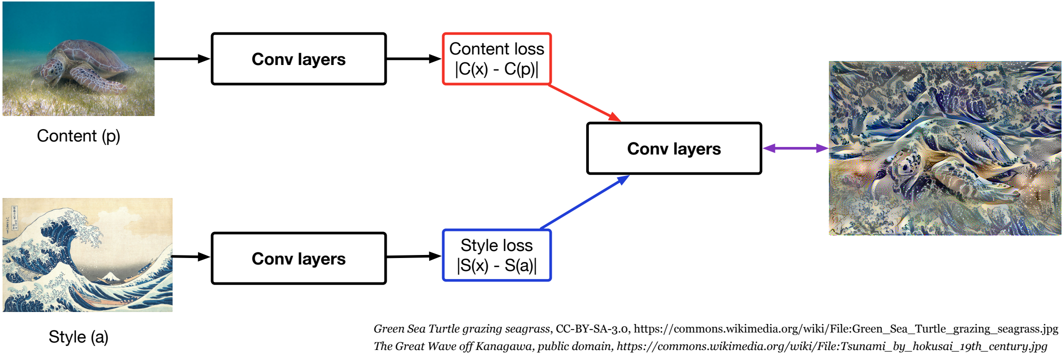
  - We will generate a new data from the initial value of $\mathbf{x}$ and the desired output $\mathbf{r}$.
  - Network weights do not change

Deep dream


Horizon
Trees
Leaves
Towers & Pagodas
Buildings
Birds & Insects

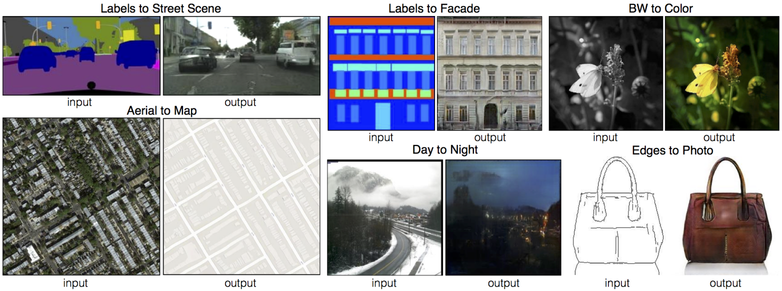
Style transfer

- Idea: transfer the style of an image into a new image
  - Compare the content in the convolution layers (e.g. VGG19) and the style (Gram matrix)



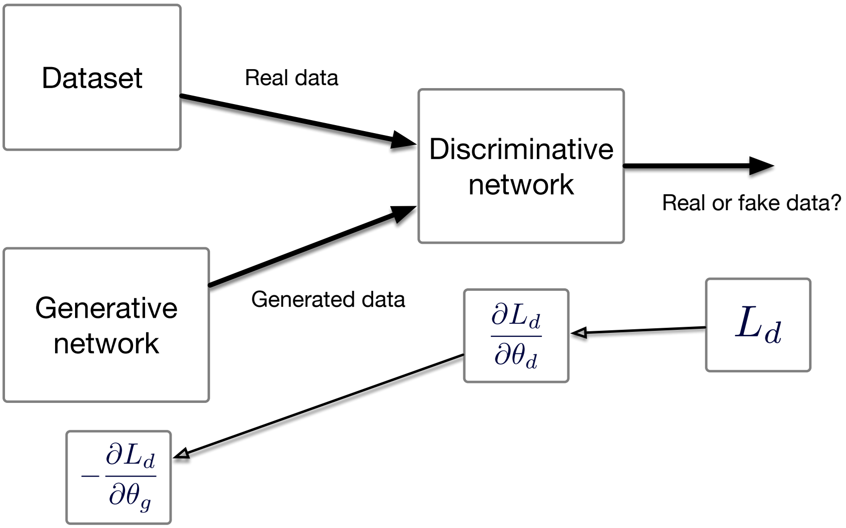
Generative Adversarial Networks (GAN)

- GAN model: putting in competition two neural networks
  - Discriminative network: distinguishing true data from the problem from generated data
  - Generative network: producing data that looks authentic
  - Allows various treatments based on unsupervised learning
- Example: image-to-image translation with conditional GANs



From *Isola, Zhu, Zhou and Efros, Image-to-Image Translation with Conditional Adversarial Networks, CVPR, 2017.* Accessed online on October 19, 2020, at `https://arxiv.org/pdf/1611.07004v3.pdf`.

Generative Adversarial Networks (GAN)



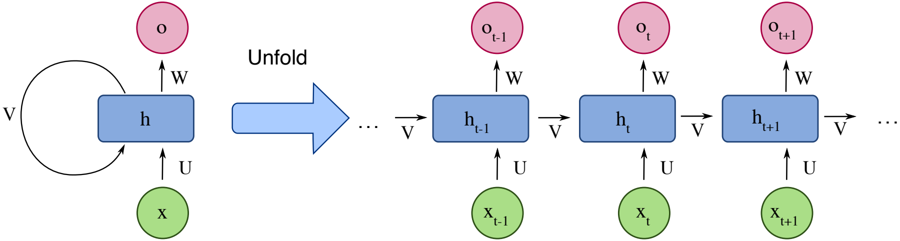
Features of GAN

- Key method in the development of generative models
  - Most historical generative models capable of realistic results are based on GANs
  - E.g., *This person does not exist* based on StyleGAN
- Self-supervised training, without requiring labelled data or explicit quality metrics
  - Triggering advances in the use of self-supervised approaches to train deep networks
  - No guarantee of the realism and quality of the data produced
- Model complex to train
  - Balance in training generative and discriminative models difficult to maintain, discriminative task easier than generative task
  - Loss of coverage in generation through mode collapse
  - Training can be quite computationally intensive

10.5 Sequence processing

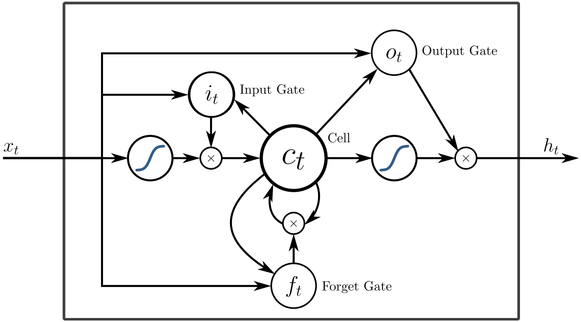
Recurrent network

- Usual networks (*feedforward*): data propagated in the network, independent of the following / previous data
  - Sequential data processing important in many contexts
- Recurrent networks: connections with previous values
  - Processing with usual algorithms by unrolling the network



By fdeloche, CC-SA 4.0, `https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`

Long Short-Term Memory (LSTM)



By Graves, Mohamed and Hinton, CC-SA 4.0, `https://en.wikipedia.org/wiki/File:Peephole_Long_Short-Term_Memory.svg`

- LSTM model: adding memory to the network
- Memory cell (state), with four neurons
  - Input
  - Input activation
  - Forgetfulness activation
  - Output activation

LSTM variants

- Bidirectional LSTM (BiLSTM): process sequence in the two directions
  - Additional cells to process data in reverse direction
  - Allows better use of sequence content
  - Particularly useful for natural language processing
- GRU (*Gated Recurrent Unit*): simplification of the LSTM model
  - Simplification of the LSTM cell model by combining input activation and forgetting.
  - Compromises between complexity and performance

LSTM strengths and weaknesses

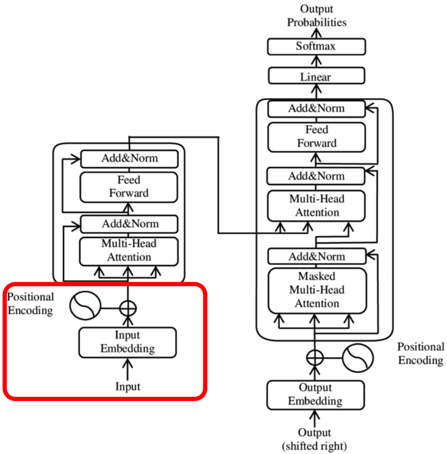
- Strengths of LSTMs
  - Able to capture distant relationships in sequences
  - Has demonstrated great versatility in its application to sequence processing (e.g. automated translation, speech recognition)
  - Offers better control over vanishing gradient, which is an issue with classical recurrent networks
- Weaknesses of LSTMs
  - Complex models, with a high number of parameters, requiring long training times and large datasets
  - Tends to overfit, especially on small datasets

10.6 Transformer networks

Transformer networks

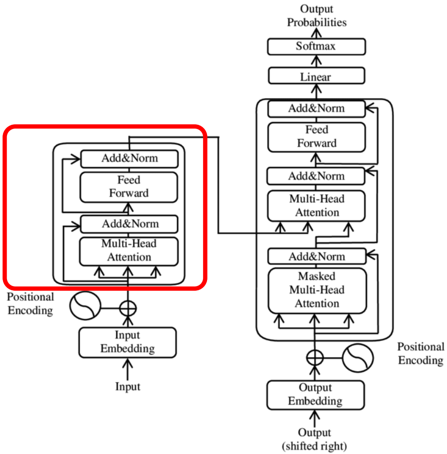
- Transformer networks
  - Uses an attention mechanism to establish relationships between elements in a sequence (e.g. words in a sentence)
  - Designed to enable parallel processing with multiple heads, allows efficient use of GPUs
  - Include an encoder component and a decoder component
  - Does not use recurrence, attention mechanism gives ability to use whole context (long-term memory)
- Central models for large language models (GPT, BERT)
  - Also used with images (*vision transformers* (ViT)), speech recognition, etc.

Transformer networks functioning



- Input: transform input sequence into a vector
  - For text, lexical embedding + positional encoding of each word

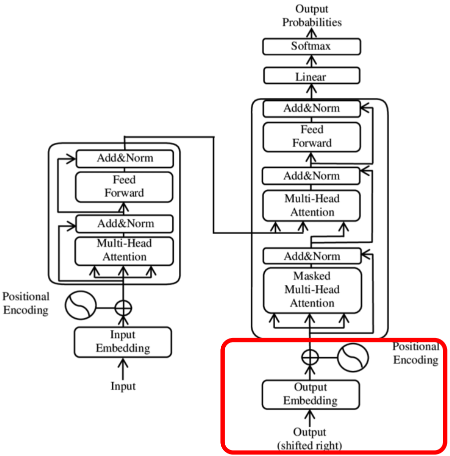
Transformer networks functioning



Par Yuening Jia, CC BY-SA 3.0 DEED, `https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png`.

- Input: transform input sequence into a vector
  - For text, lexical embedding + positional encoding of each word
- Encoder: multi-headed attention + renormalization
  - Attention calculated between all elements
  - Normalization by fully connected layers

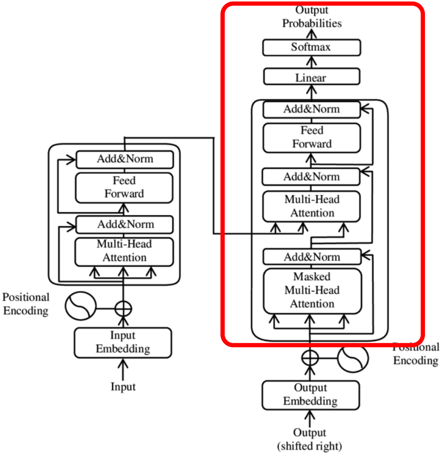
Transformer networks functioning



Par Yuening Jia, CC BY-SA 3.0 DEED, `https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png`.

- Input: transform input sequence into a vector
  - For text, lexical embedding + positional encoding of each word
- Encoder: multi-headed attention + renormalization
  - Attention calculated between all elements
  - Normalization by fully connected layers
- Output: transform output sequence into a vector

Transformer networks functioning



Par Yuening Jia, CC BY-SA 3.0 DEED, https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png.

- Input: transform input sequence into a vector
  - For text, lexical embedding + positional encoding of each word
- Encoder: multi-headed attention + renormalization
  - Attention calculated between all elements
  - Normalization by fully connected layers
- Output: transform output sequence into a vector
- Decoder: attention mechanism on output and input
  - First steps only on **masked** output
  - Next steps combining output and input representation
  - Fully connected layer normalization
  - Output next word probabilities

Attention mechanism

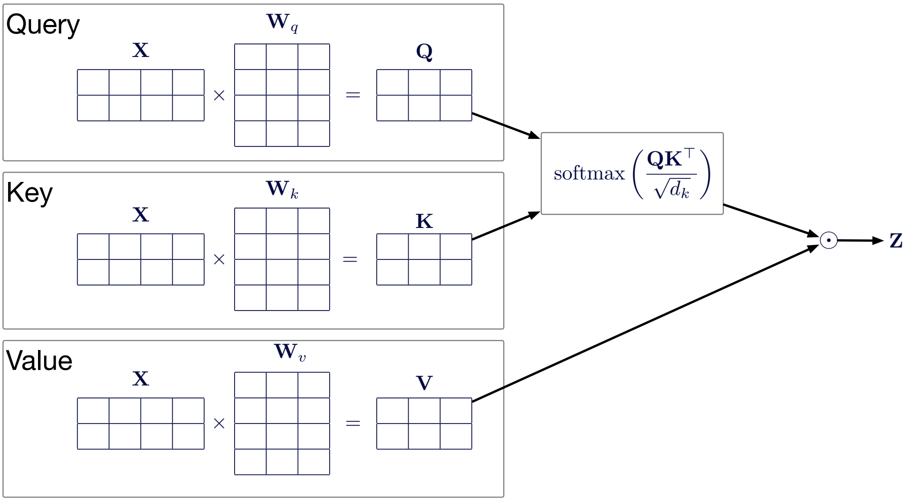
- Compute the attention between the query $\mathbf{Q}$, the key $\mathbf{K}$ and the value $\mathbf{V}$ according to:

$$\mathrm{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

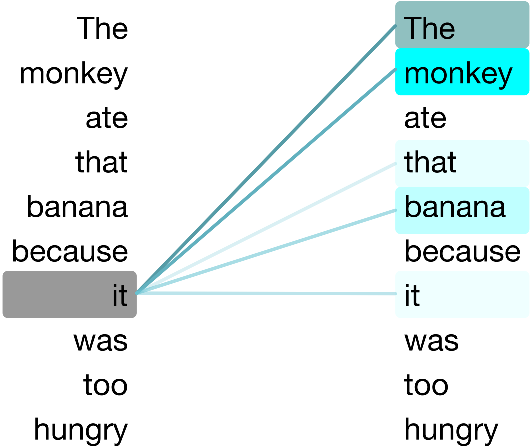
.

  - The values of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ result from the application of weights $\mathbf{W}_q$, $\mathbf{W}_k$ and $\mathbf{W}_v$ on the data $\mathbf{X}$
  - Division by $\sqrt{d_k}$ to stabilize the gradient (d_k: key size $\mathbf{K}$)
- Each head works in parallel with its own weights $\mathbf{W}_q$, $\mathbf{W}_k$ and $\mathbf{W}_v$.

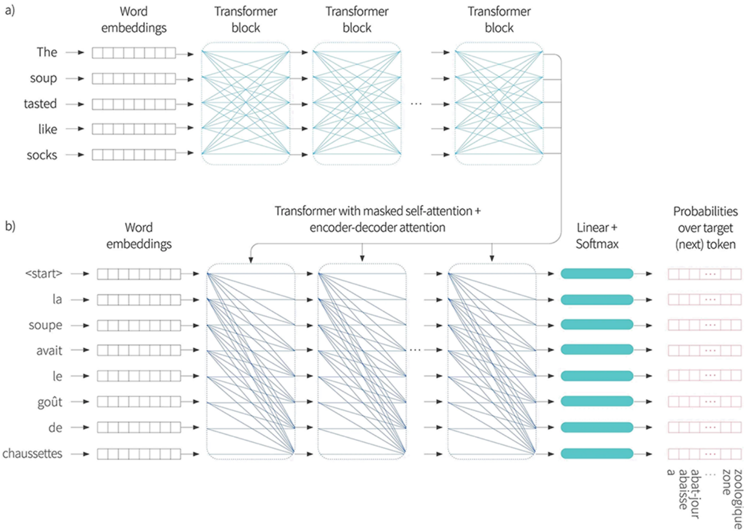
Attention mechanism



Attention mechanism



Example: application to translation



Source: `https://www.borealisai.com/wp-content/uploads/2021/06/T14_10.png`, accessed on November 1, 2023.

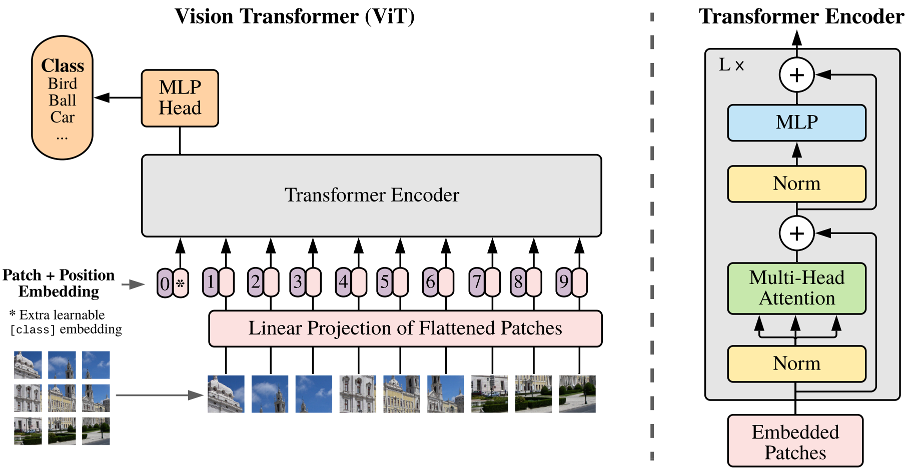
Large Language Models

- BERT (*Bidirectional Encoder Representations from Transformers*)
  - Proposed in 2018 by a team of Google researchers
  - Consisting of a lexical embedding module, several layers of self-attentive encoders, and conversion to probabilistic output
  - Rapidly becoming a central model in natural language processing
  - Since 2020, virtually all English search queries on Google are processed with BERT
- GPT (*Generative Pre-trained Transformer*)
  - OpenAI's family of transformer-based models, proposed in 2018 (GPT-1)
  - GPT-3: GPT-1 + modified normalization (GPT-2) + scaling, proposed 2020, 175G parameters trained on 500G tokens
  - GPT-4: undisclosed architecture, but estimated at 1.7T (1700G) parameters
  - ChatGPT: integration of GPT-3.5 / GPT-4 and reinforcement learning with human feedback (RLHF)

Vision transformers (ViT)

- Vision transformers (ViT): adapting the transformer architecture to computer vision
    - Instead of processing word sequences, processes fixed size, non-overlapping image patches
    - Each patch is represented by a 1D vector, with positional information added to the representation
    - Patch representation provided as a sequence to the transformer network
- ViT characteristics
    - Able to capture complex and distant relationships in images, without requiring convolution layers
    - Can achieve state-of-the-art performance with sufficient data and resources
    - Requires very large datasets and intensive training to perform well

Vision transformers (ViT)



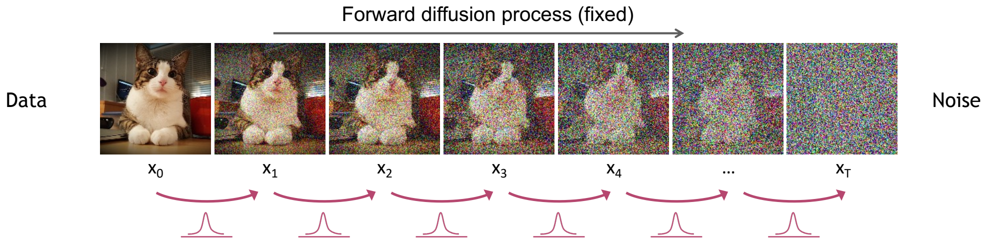
Taken from *Dosovitskiy et al., An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, ICLR, 2020.* Accessed on-line on November 2, 2023, at `https://arxiv.org/pdf/2010.11929.pdf`.

10.7 Diffusion models

Diffusion models

- Diffusion models: class of generative models simulating a random diffusion process transforming a data instance into a noise instance
  - Inspired by physics, with diffusion of particles from a medium of high concentration to a low concentration one
  - Used for image generation, denoising or inpainting
- Diffusion processes
  - Forward diffusion: start with a clear image to which light noise is successively added until the image is nothing but noise
  - Reverse diffusion: start the process with an image of pure noise, on which successive denoising operations are applied to obtain a clear image
  - Each forward or reverse diffusion step guided by a transition function, typically Gaussian, conditioned on the current state
  - Once the reverse diffusion (denoising) mechanics have been learned, they can be used to generate new images

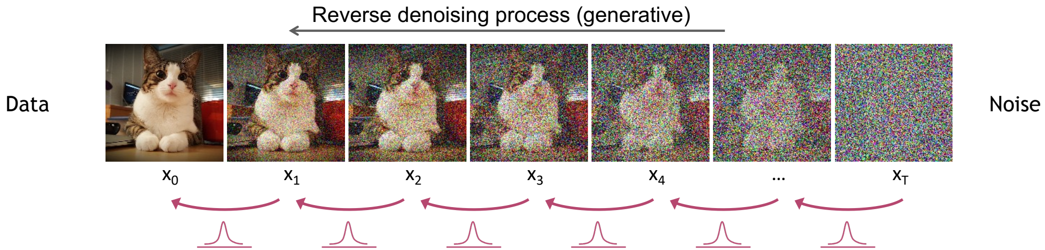
Forward diffusion process



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \text{(joint)}$$

Taken from Song, Meng and Vahdat, *Denoising Diffusion Models: A Generative Learning Big Bang*, CVPR 2023 tutorial, `https://cvpr2023-tutorial-diffusion-models.github.io/`, accessed on November 2, 2023.

Reverse diffusion process



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network (U-net, Denoising Autoencoder)}}, \sigma_t^2 \mathbf{I})$$

$\Rightarrow$

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

19

Taken from Song, Meng and Vahdat, *Denoising Diffusion Models: A Generative Learning Big Bang*, CVPR 2023 tutorial, `https://cvpr2023-tutorial-diffusion-models.github.io/`, accessed on November 2, 2023.

Training diffusion models

- Forward diffusion: typically consists of applying Gaussian noise to pixels
  - Repeated application of a small amount of Gaussian noise transforms the set of pixels into random values having a Gaussian distribution
  - Level of noise applied can vary in the sequence according to a schedule
- Reverse diffusion: neural network to remove noise
  - Use forward diffusion data to train the denoising network
  - Denoising network receives current noise level
  - U-Net commonly used as denoising network
- Reverse diffusion process can be conditioned
  - Specific class targeted
  - Text query, using vector representation (lexical embedding or transformer network)

Strengths and weaknesses of diffusion models

- Strengths of diffusion models
    - Capable of generating high-quality data
    - Flexible and can be adapted to different data types, works with complex data distributions
- Weaknesses of diffusion models
    - Generation process can be computationally heavy, with the many iterations required in reverse diffusion.
    - Training is computationally heavy
- These models form the basis of generative image models such as DALL-E (OpenAI), Midjourney or Stable Diffusion.