Unveiling the Mystery of Deep Learning: Past, Present, and Future

Dr. Elham Barezi,
AI Research scientist

Co-Sponsored by Rosen Center for Advanced Computing (RCAC), and IPAI

Spring 2025

PURDUE UNIVERSITY

Course Outline

1. History and Basics of DNN
    a. From traditional ML to DNN
2. Fundamental deep learning: from discriminative to generative
    a. CNN, RNN, Autoencoders, attention,
    b. Deep learning for Representation Learning and feature extraction
    c. Discriminative vs generative deep learning: VAE, GAN, Diffusion Models
3. Transformers Era
    a. self-attention, encoders, decoders, masking,
    b. Transformers for other modalities: text, image, video, speech,
4. LLMs in Practice
    a. Prompt Engineering Methods: COT, TOT, Self-Consistency, RAG, Agents,
    b. Fine-tuning Methods: instruct tuning, RLHF, Adapters like LORA,
5. Deep learning for different domains
6. AI safety and Governance

PURDUE UNIVERSITY

Course Outline-first session-March 5th 2025

1. History and Basics of DNN
    a. AI hypes and winters
    b. Deep learning from 1950s
    c. From single neurons to deep networks
    d. Deep learning challenges solved from 1950-present
        i. Model overfitting
        ii. Activation function saturation
        iii. Vanishing/exploding gradient
    e. Deep learning weaknesses

PURDUE UNIVERSITY

Course Outline-first session-April 18th 2025

2. Fundamental deep learning models, from discriminative to generative:

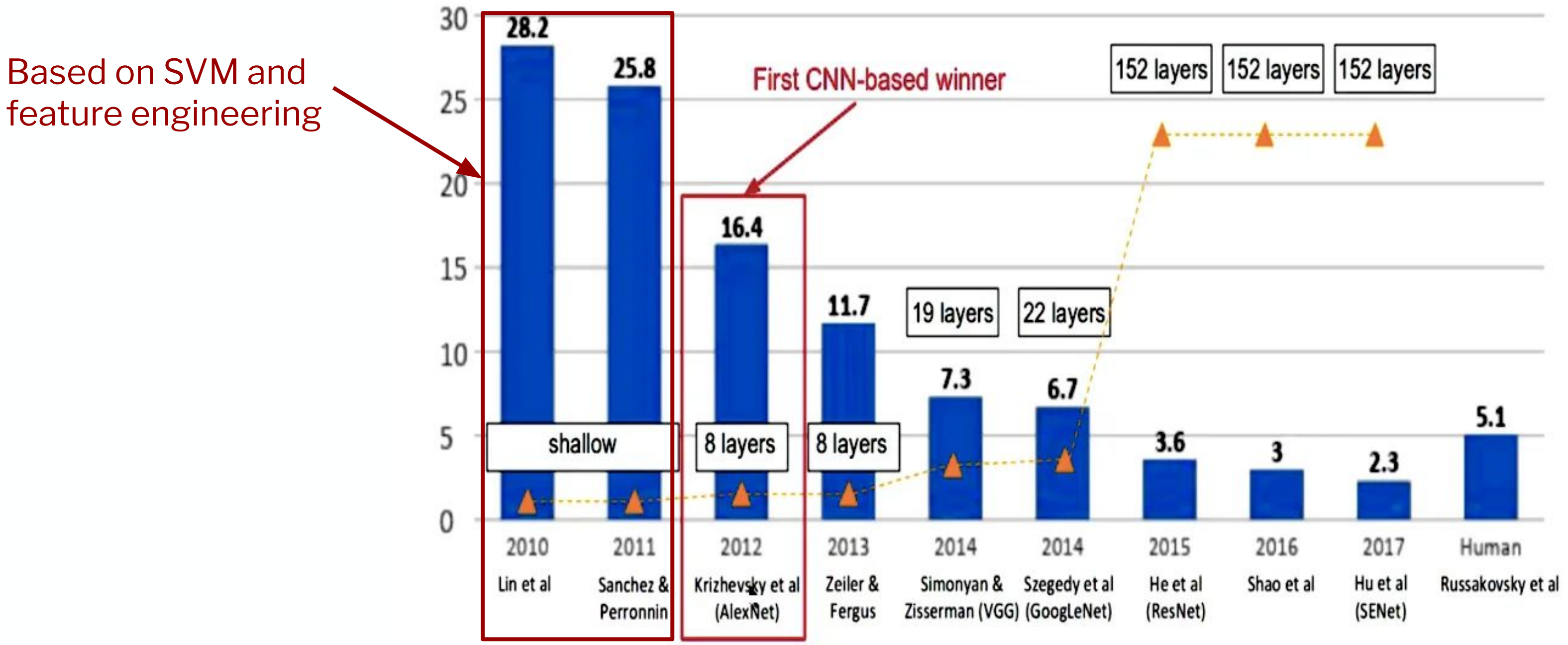
    a. CNN,
    b. RNN,
    c. Earlier version of **attention**,
    d. Deep learning for **Representation Learning and feature extraction**
    e. Earlier **Pre-Training** models
    f. Discriminative vs Generative deep learning

DNN era before Transformers (1990-2015): CNN, RNN, Attention, and Auto-Encoders

PURDUE UNIVERSITY

CNN for Image Processing

- **Neocognitron (1980):** Neocognitron was the first architecture of its kind, perhaps the earliest precursor of CNNs
- **LeNet-5 (1989–1998):** The name convolutional neural networks actually originated with the design of the LeNet by **Yann LeCun** and team
- **AlexNet (2012):** AlexNet was the first winner of the ImageNet challenge and was **based on a CNN**
- **ResNet (2015)**: Kaiming He et. al. from Microsoft Research came up with an idea of '**residual blocks**' which are connected to each other through identity (skip)



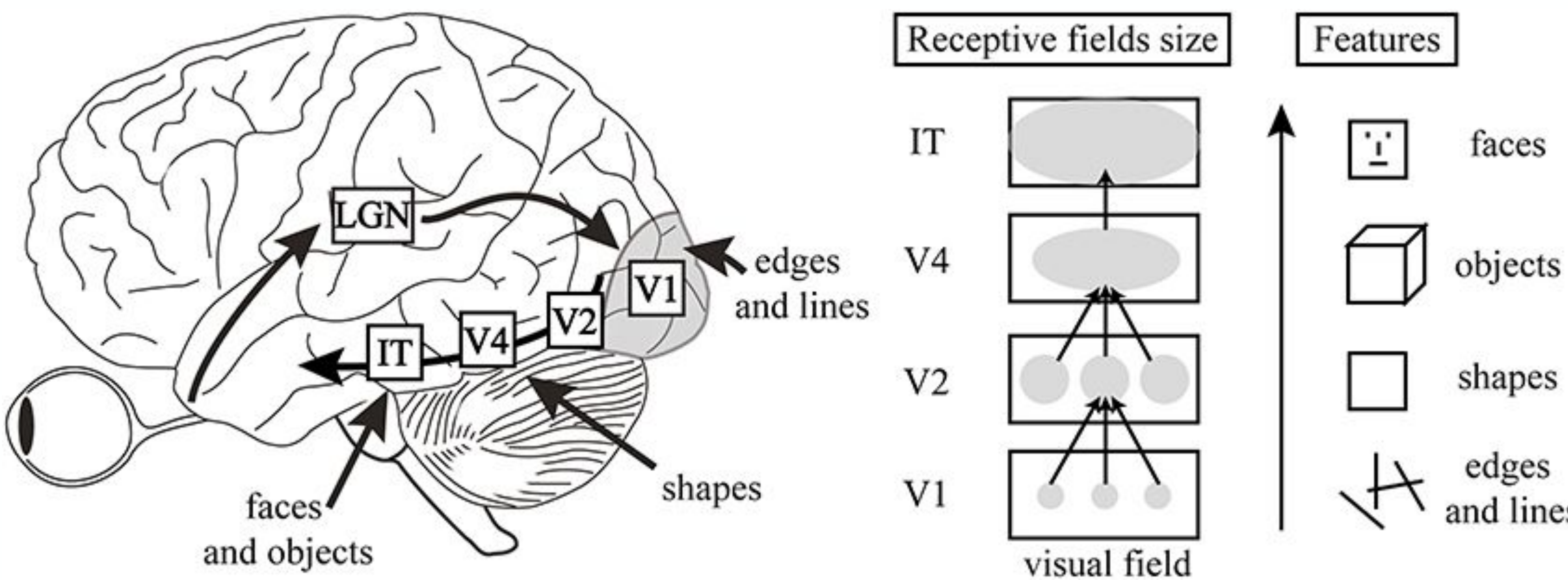
Winners of ImageNet Classification Challenge [7]

CNN Story!

Feature	Neocognitron	LeNet-5	AlexNet
Year Introduced	1980	1998 by Yann Lecun	2012
Purpose	Handwriting / pattern recognition	Digit recognition (e.g. MNIST)	Image classification (ImageNet)
Architecture Summary	Alternating **S-cells (feature extractors)** and **C-cells (pooling)**	Input → Conv1 → Pool1 → Conv2 → Pool2 → FC1 → FC2 → Output	Conv1 → LRN → Pool → Conv2 → LRN → Pool → Conv3 → Conv4 → Conv5 → Pool → FC6 → FC7 → FC8
# of Layers	~10+ (depends on config)	7 layers (2 conv, 2 pool, 3 FC)	8 layers (5 conv, 3 FC)
Training Method	hand-crafted weights	**Backpropagation** (supervised)	Backpropagation + **GPU** acceleration

CNN Inspiring from visual Cortex

- CNNs are loosely modeled after the way the **visual cortex** processes information.
- Neurons in the brain respond to small regions of the visual field, and CNNs mimic this with their **receptive fields**.

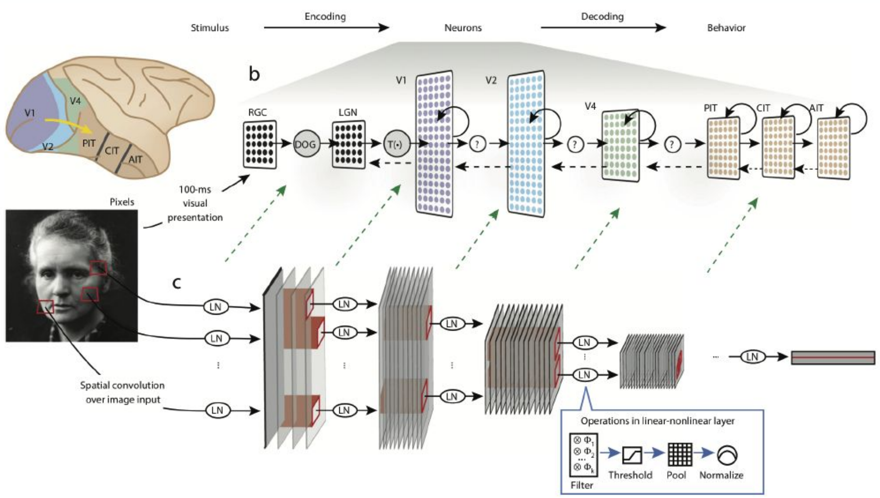


https://gracewlindsay.com/

CNN Inspiring from visual Cortex

Visual System	CNN Equivalent	Notes
LGN (Lateral Geniculate Nucleus)	Image Preprocessing / Pooling	Controls signal flow
V1 (Primary Visual Cortex)	First Conv Layer	Edge detectors
V2	Second Conv Layer	Shape detectors
V4	Mid Conv Layer	Color and complex shapes
IT (Inferotemporal Cortex):PIT, CIT, AIT	Deep Layers / Fully Connected	Object-level representation

side-by-side comparisons between biological and artificial neural networks, Yamins and DiCarlo [2016].

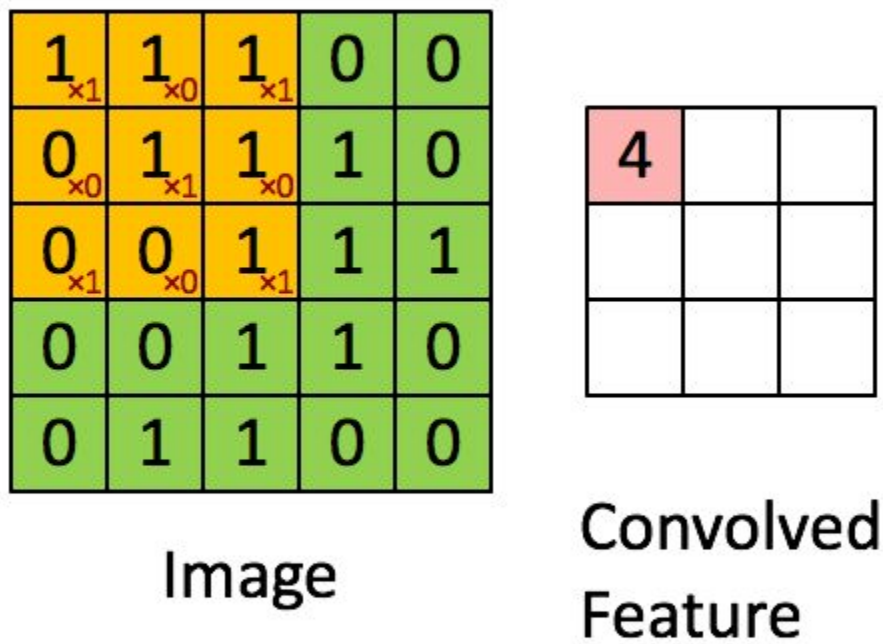


What is a Convolution?

$$(f * h)(t) = \sum_{k=-T}^{T} f(t)h(t-k)$$

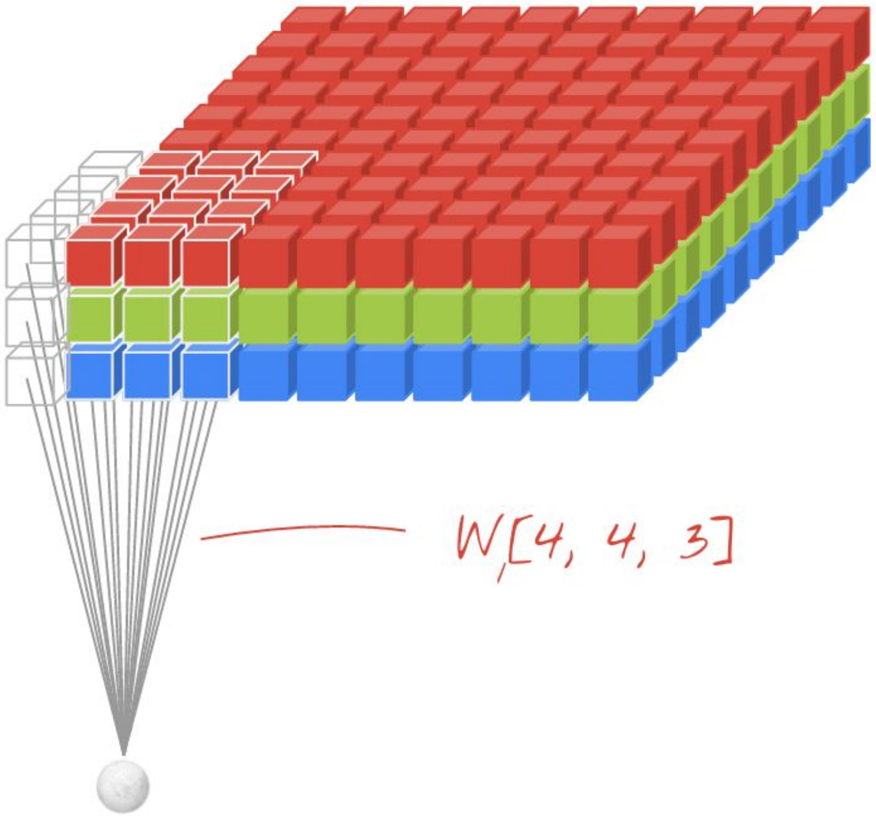
- h(t) is the discrete time input signal,
- f(t) is the **kernel function** that determines the outcome of the convolution.
- f(t) **slides over** the input sample, taking a **weighted average** at each step.
- weighted average is a linear **differentiable function**
  - Back propagation based on gradient descent optimization
  - The kernel parameters are learned in the training phase.

What is a Convolution?



http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

3D Convolution filters



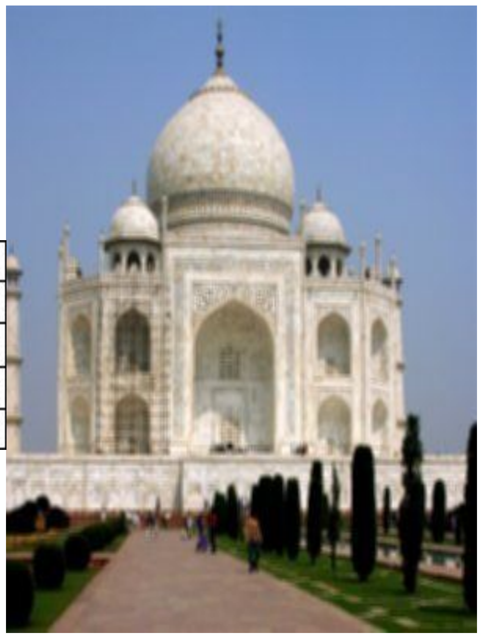
https://www.linkedin.com/pulse/convolutional-neural-networks-gaurav-jain/

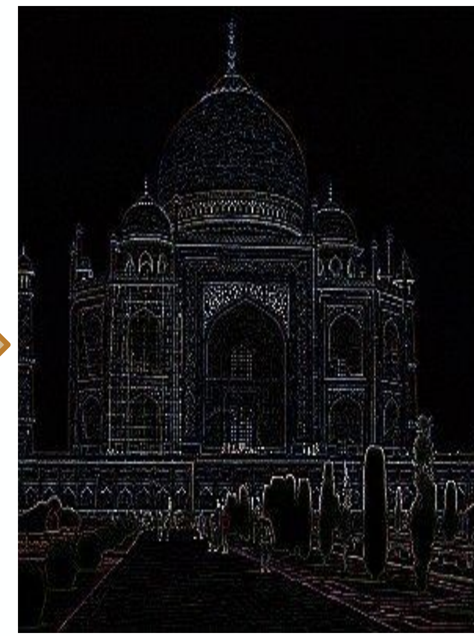
What is a Convolution?

smoothing: Average each pixel with its neighbors

Edge Detector: Difference of a pixel and surrounding neighbors

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0





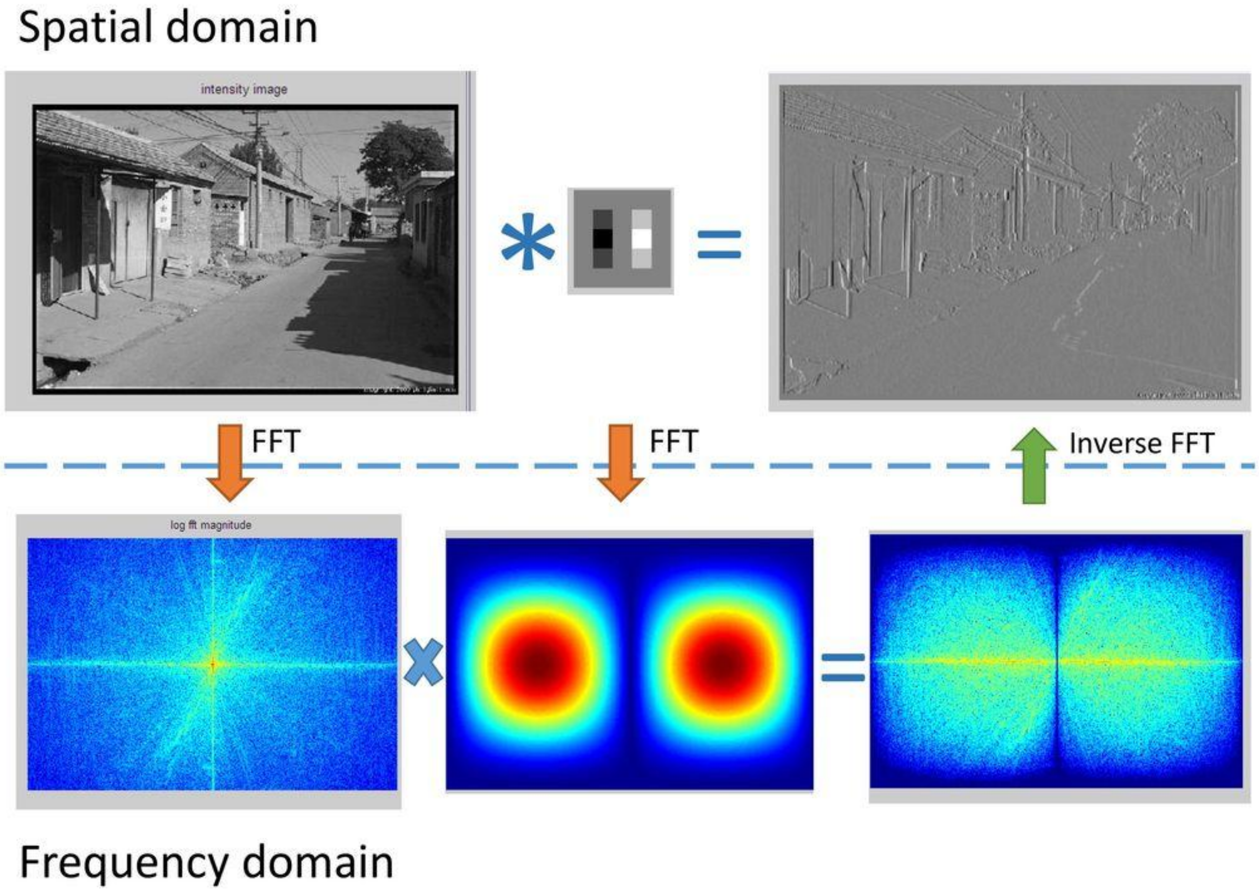
	0	1	0	
	1	-4	1	
	0	1	0	

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

What is a Convolution?

filter

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

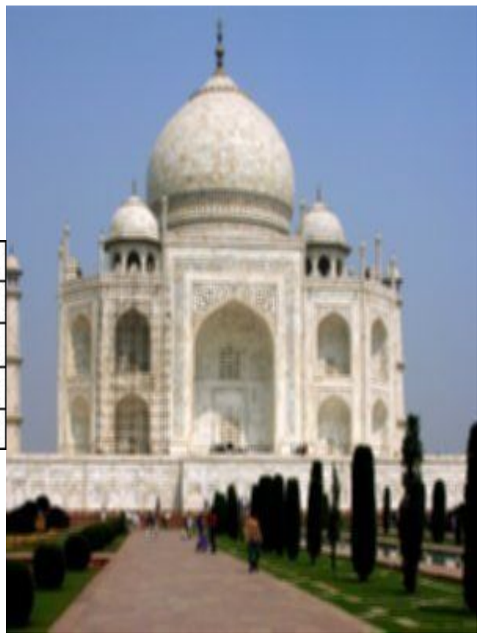


http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

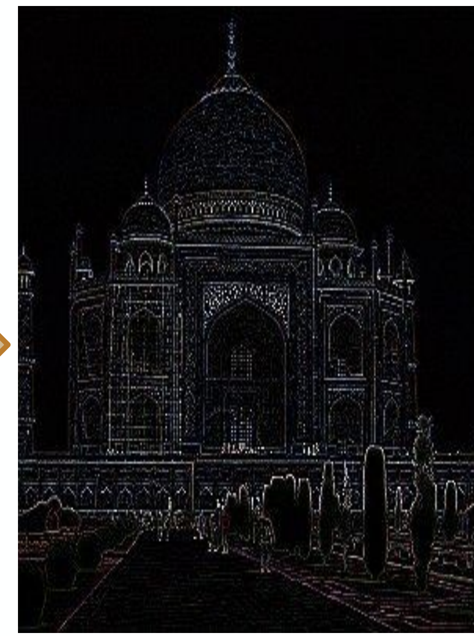
Convolutional Neural Networks elements : Filter

smoothing: Average each pixel with its neighbors

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0



Edge Detector: Difference of a pixel and surrounding neighbors

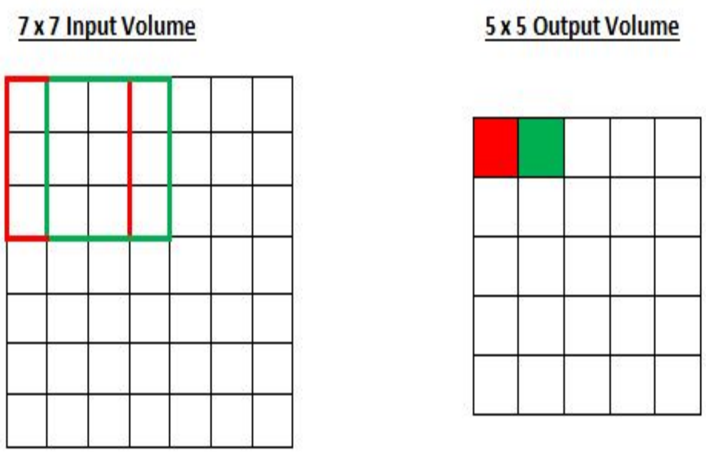


	0	1	0	
	1	-4	1	
	0	1	0	

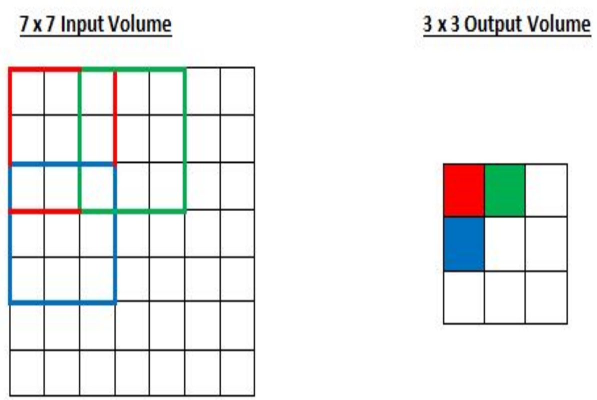
http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

CNN Elements : Stride

- The amount by which the filter shifts is the stride



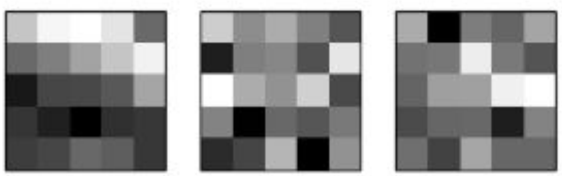
a 7 x 7 input volume, a 3 x 3 filter (Disregard the 3rd dimension for simplicity), and a stride of 1.



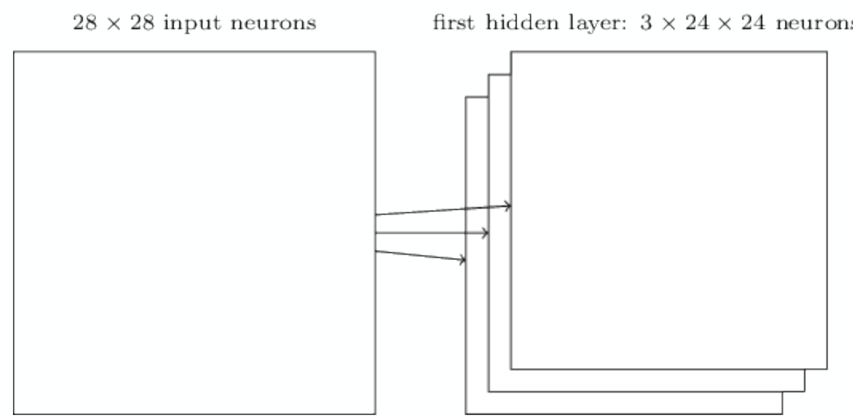
What will happen to the output volume as the stride increases to 2?

CNN Elements : Parameter Sharing for Strides

- Parameter sharing scheme is used in Convolutional Layers to control the number of parameters in CNN.



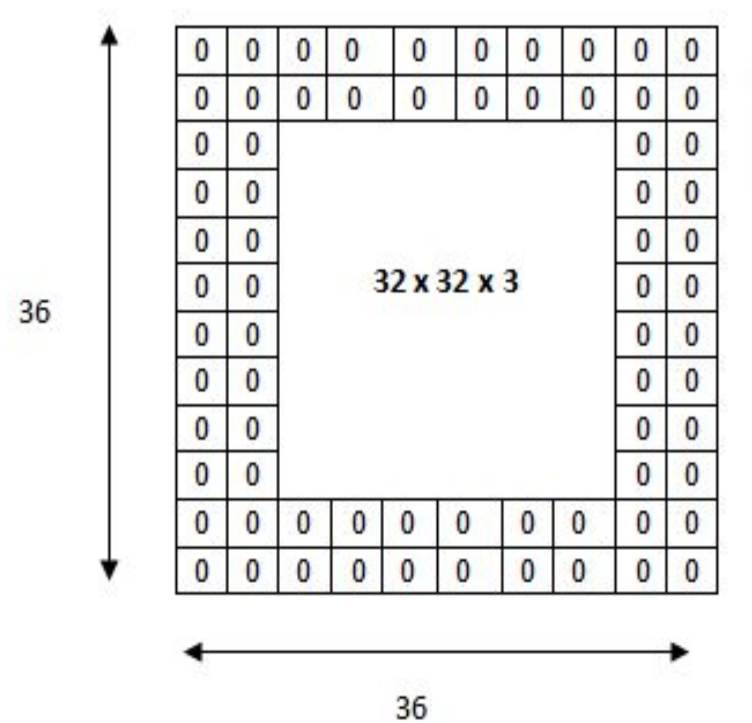
➔



These filter weights are shared across all the hidden neurons.

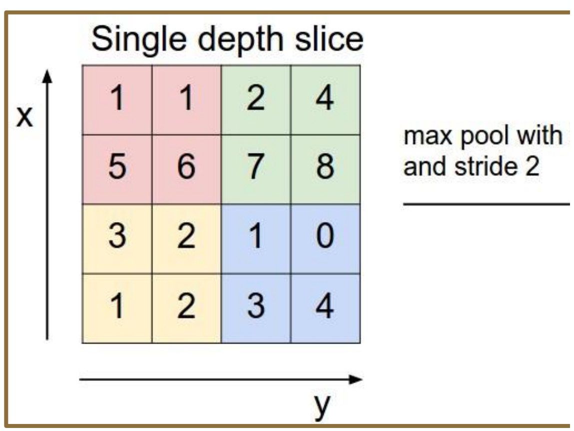
CNN Elements : Padding

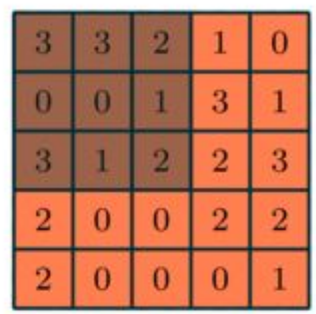
Let's say we want to apply the same conv layer but we want the output volume to remain 32 x 32 x 3. To do this, we can apply a zero padding of size 2 to that layer. Zero padding pads the input volume with zeros around the border.

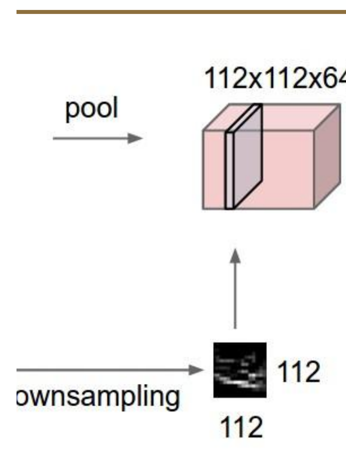


CNN Elements : Pooling

- **Pooling layer** is referred to as a downsampling layer, for two main purposes:
  - The first is that the amount of parameters or weights is reduced by 75%, thus **lessening the computation cost**.
  - The second is that it will control **overfitting**.
  - max pooling, average pooling and L2-norm pooling



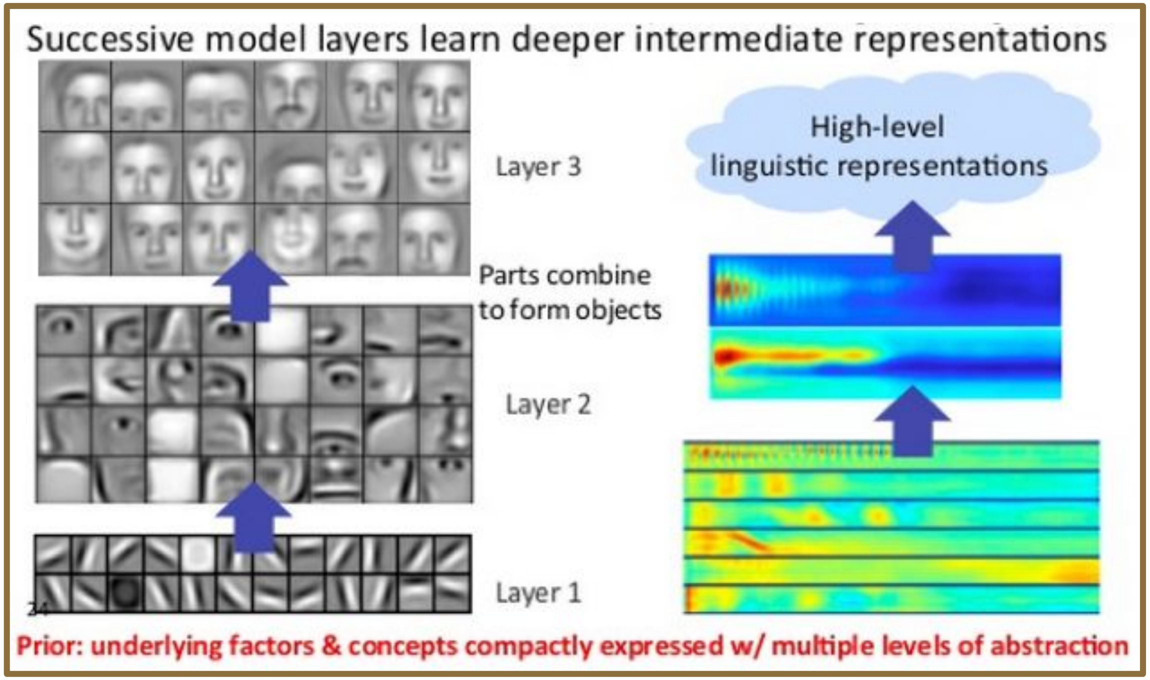




Some Key aspects of CNN

1. **Location Invariance**: We don't care where an elephant is in the image (striding everywhere)
2. **Local similarity**: Each filter extracts information in a neighborhood
3. **Compositionality**: Each filter composes a lower level feature into a higher level representation
4. **Weight Sharing = Fewer Parameters:** One of the reasons CNNs are efficient is **weight sharing** — the same filter (set of weights) slides across the image.
   a. This massively reduces the number of parameters compared to fully connected layers.

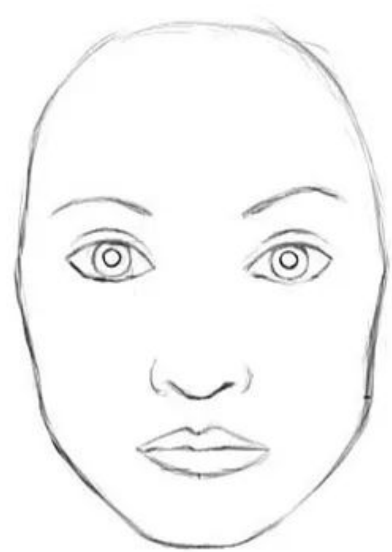
Some Key aspects of CNN

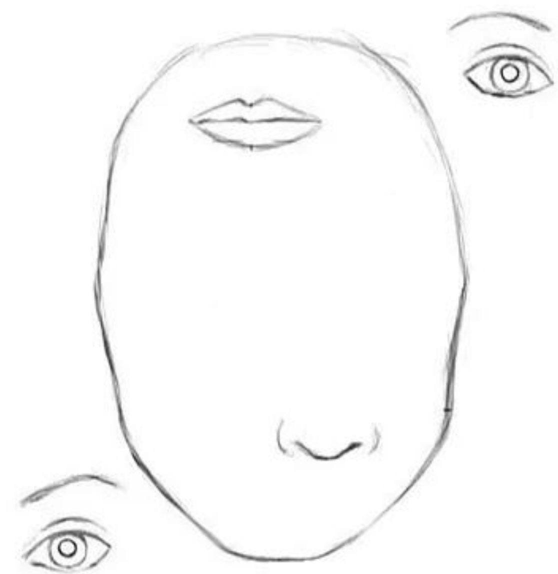


- **Low-Texture Images → Focus on Shape Features**
  - Use **larger kernels, deeper networks, edge detection augmentations.**
- **High-Texture Images → Preserve Local Textures**
  - Use **smaller kernels, more filters, texture-aware augmentations.**

CNN Challenges

- For a CNN, both of these pictures are almost same.
  - CNN does not encode the **relative position** of different features.
  - **CNNS Don’t See the Whole Picture :** Each filter in a CNN only "sees" a small part of the input at a time. But as you go deeper, they **combine local features** to understand complex global patterns(like assembling the parts of a face).
- **Large filters are required** to encode the combination of these features.
  - For examples: to encode the information “eyes above nose and mouth” require large filters.

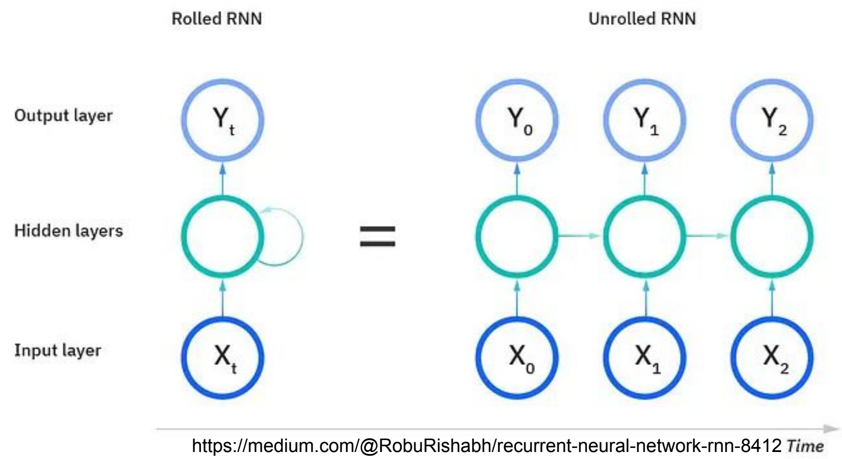




PURDUE UNIVERSITY

Recurrent Neural Network (RNN, 1980s)

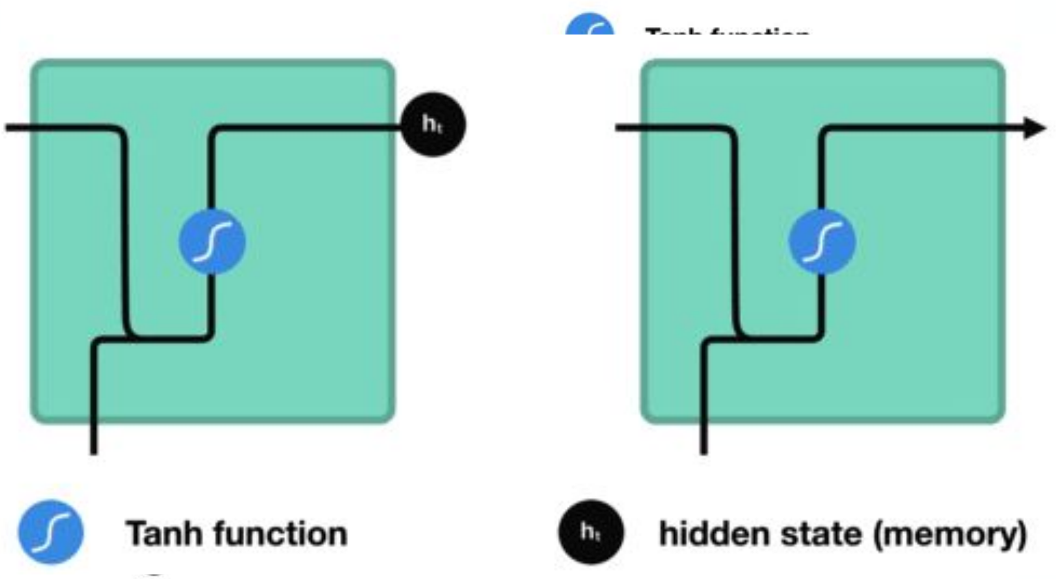
- RNNs were designed to **mimic the way humans process sequences**(**language, sound, or time-series data**), where **past context influences the present**.
- **Parameter Efficiency:** RNNs have a **compact parameter space** because they **share weights across all time steps** — making them efficient and elegant for sequential modeling.
  - It also limits the model's capacity to **learn position-specific behavior**.



https://medium.com/@RobuRishabh/recurrent-neural-network-rnn-8412b9abd755

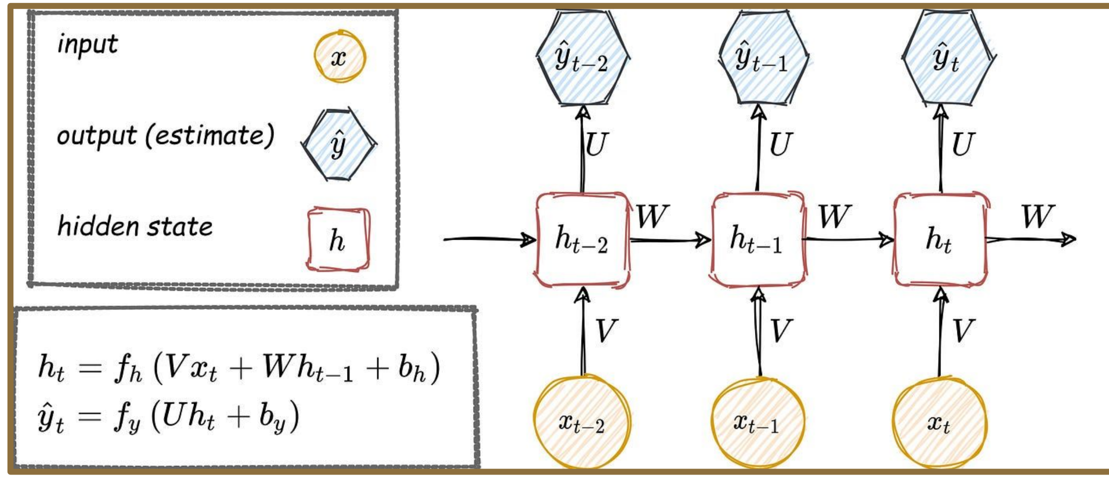
RNN Cell structure

- Hidden state is working memory of the RNN Cells to capture context. It's a vector that stores **information learned from previous time steps**.
- **Hard to parallelize** (due to sequential nature), and slow in training due to autoregressive nature.



https://penseeartificielle.fr/comprendre-lstm-gru-fonctionnement-schema/

Vanishing Gradient in RNN



https://medium.com/metaor-artificial-intelligence/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22

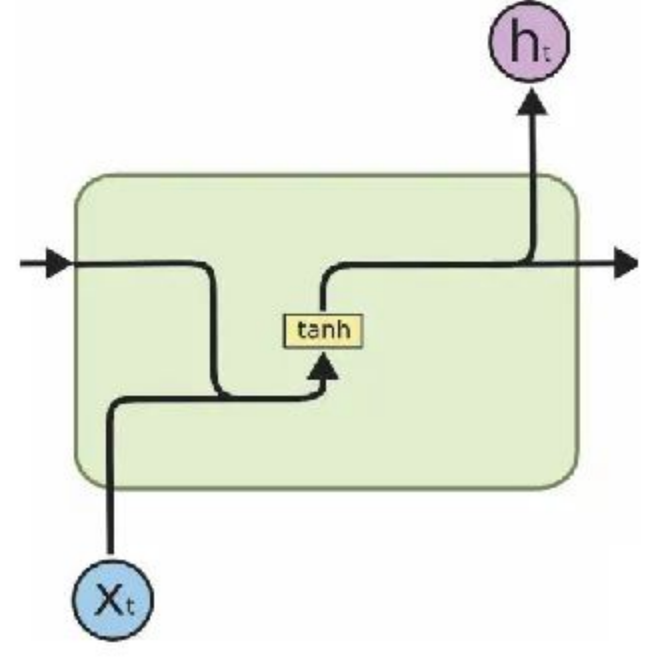
$$E(y,\hat{y}) = \sum_t E_t(y_t,\hat{y_t}) = -\sum_t y_t log\hat{y}_t$$

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

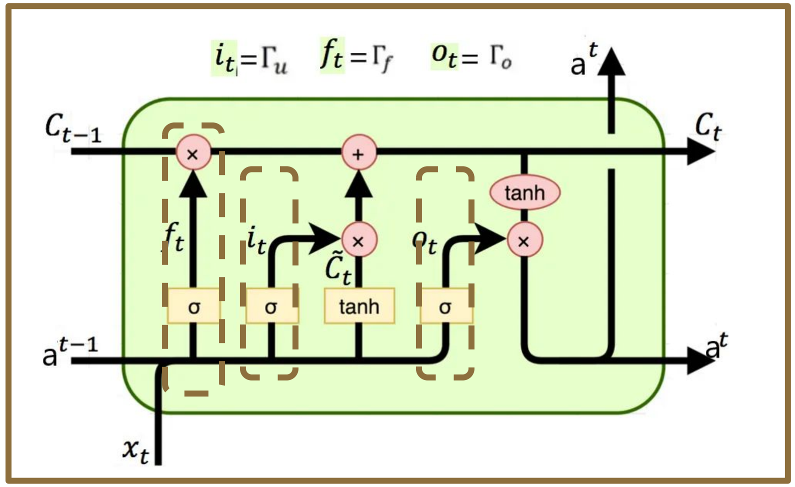
$$\frac{\partial E_T}{\partial W} = \frac{\partial E_T}{\partial \hat{y_T}}\frac{\partial \hat{y_T}}{\partial h_T}\frac{\partial h_T}{\partial W} = \frac{\partial E_T}{\partial \hat{y_T}}\frac{\partial \hat{y_T}}{\partial h_T}\frac{\partial h_T}{\partial h_{T-1}}\cdots\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial w}$$

1. ***Vanishing gradient*** $\left\|\frac{\partial h_i}{\partial h_{i-1}}\right\|_2 < 1$
2. ***Exploding gradient*** $\left\|\frac{\partial h_i}{\partial h_{i-1}}\right\|_2 > 1$

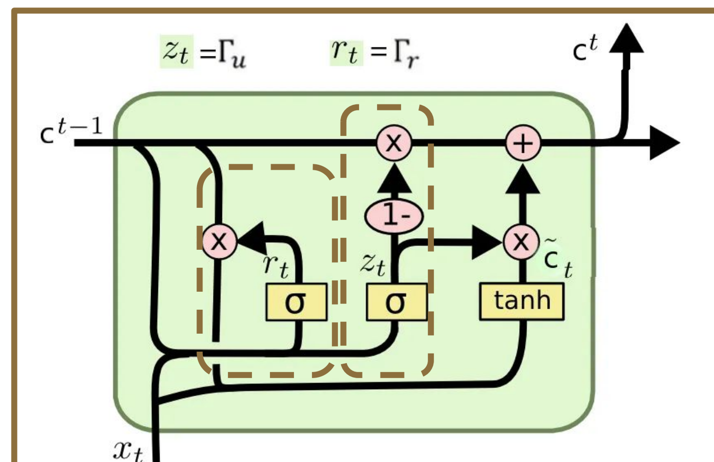
- **RNN** Forget important info from earlier in long sequences and overemphasizes new input:
    - **GRU, LSTM** were invented to decide what to forget and what to keep in memory,



RNN (1982): backpropagation through time

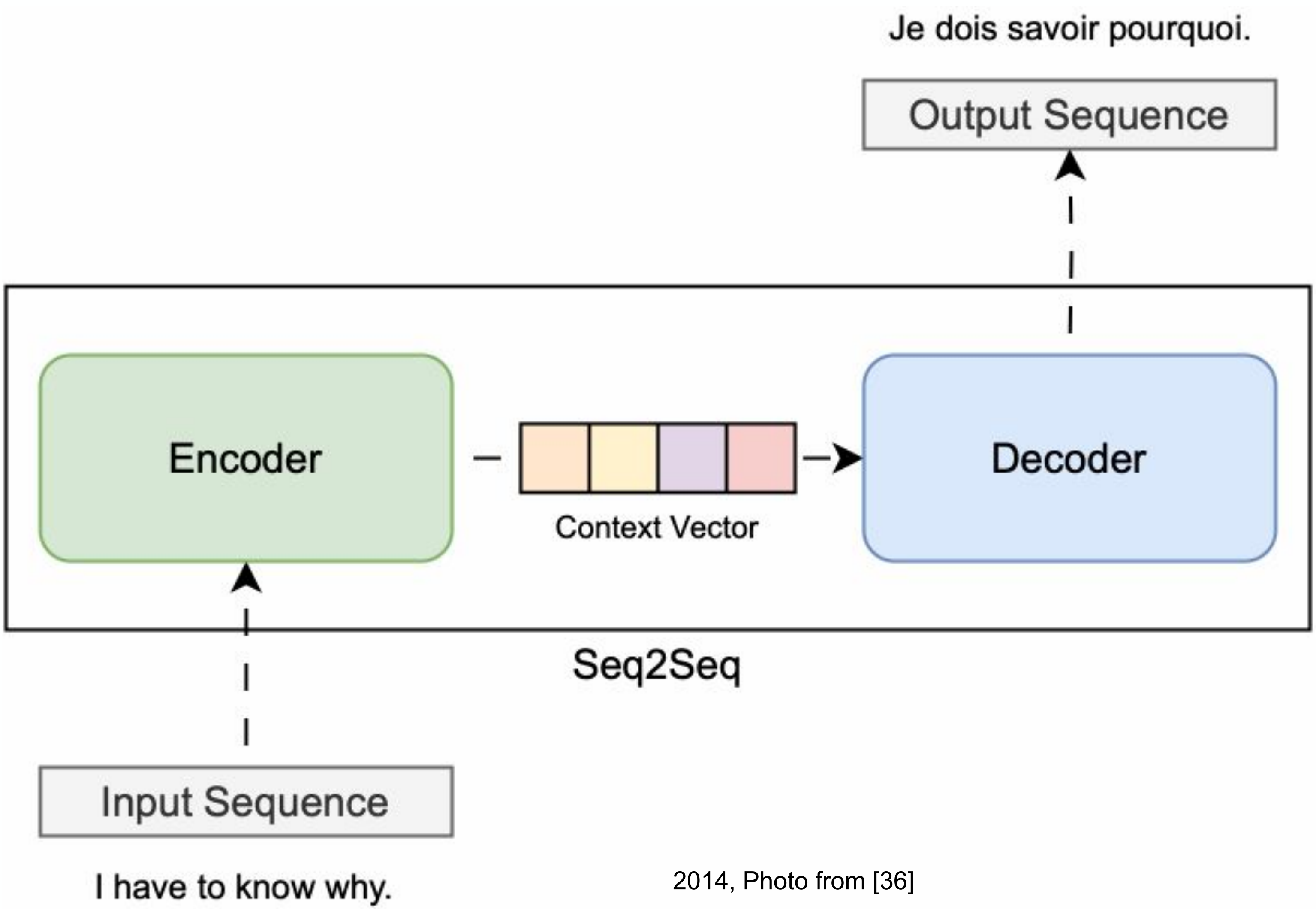


LSTM(1997): Reset and forget gate to solve the **vanishing gradient problem** in standard RNNs,



GRU(2014) :
Input, output and forget gate,
simpler alternative to LSTMs, with fewer parameters to avoid **overfitting**.

RNN to capture order in sequences



2014, Photo from [36]

RNN to capture order in sequences

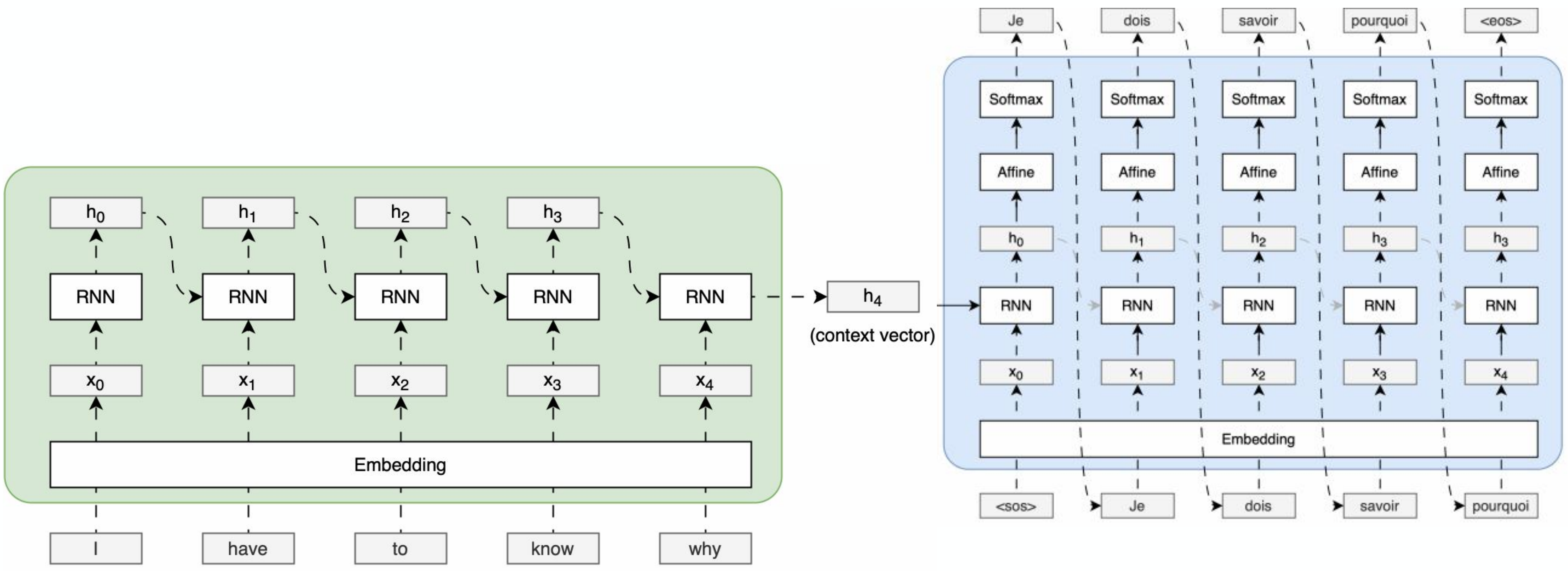
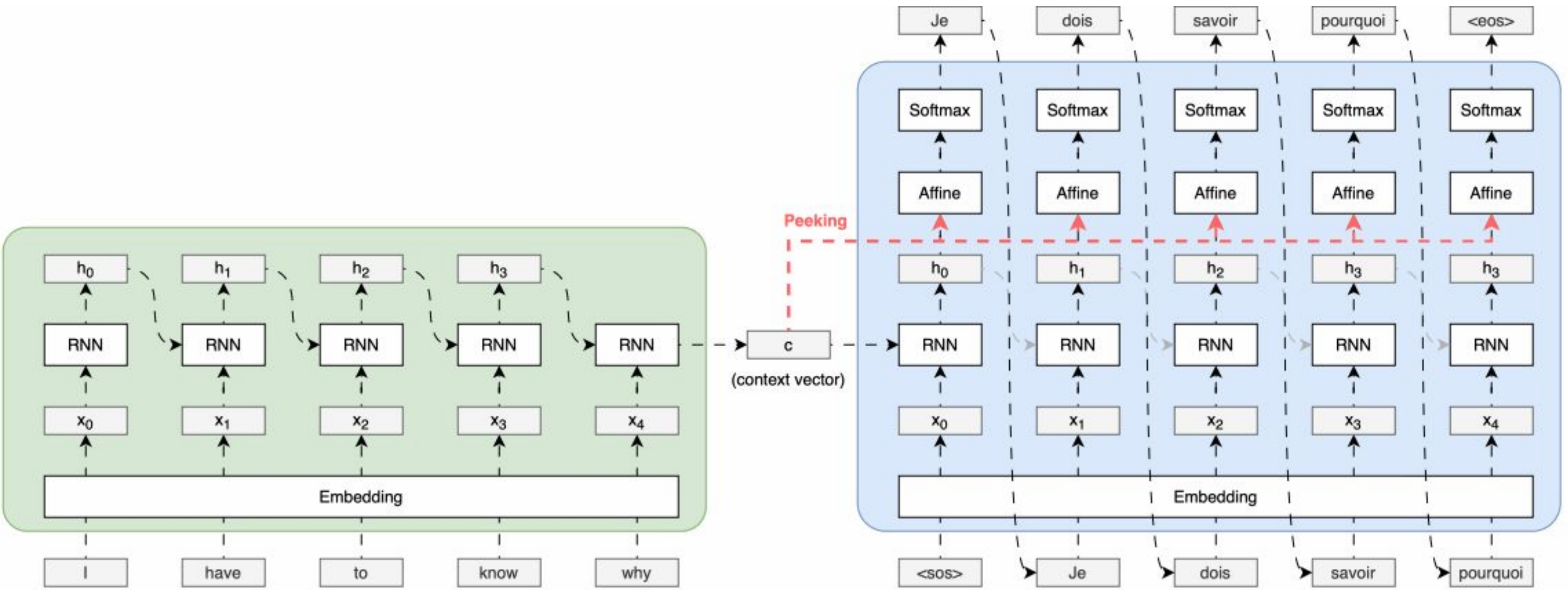


Photo from [36]

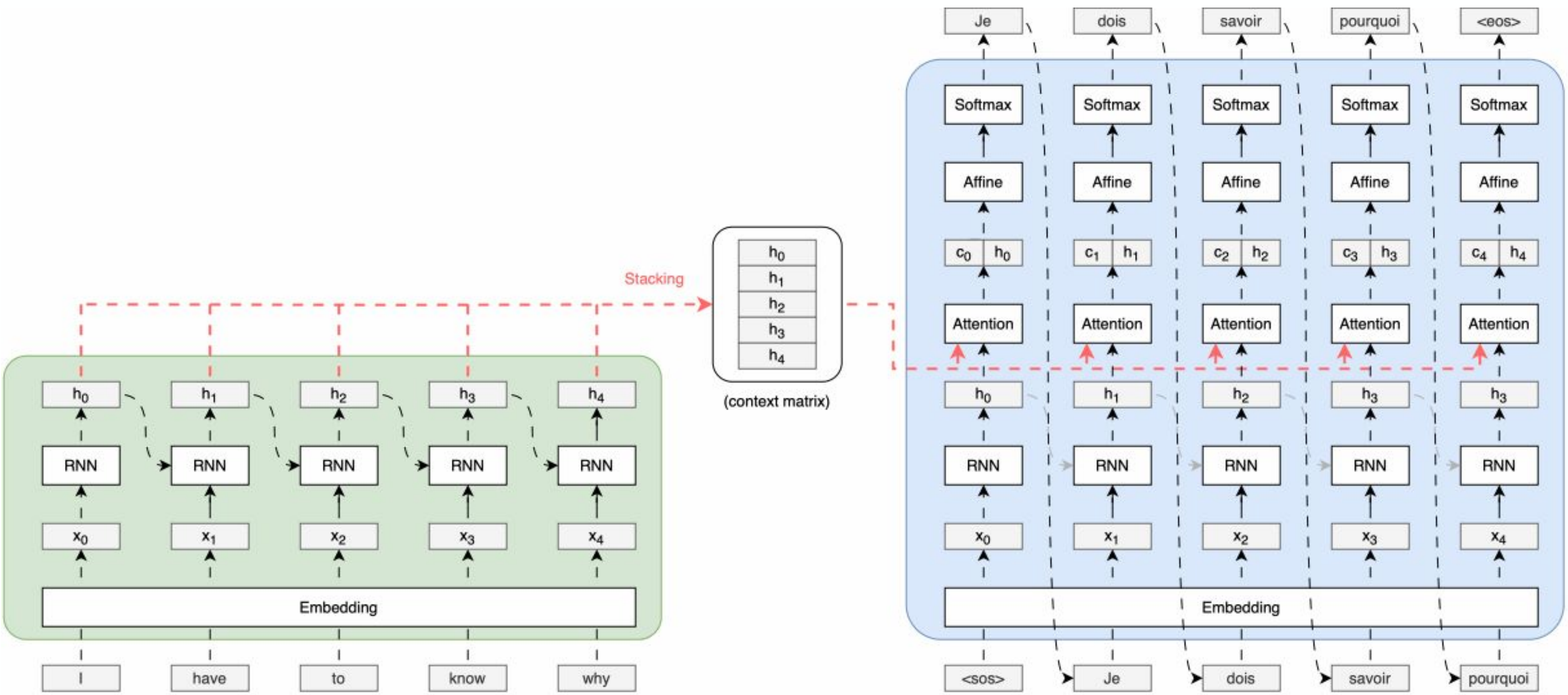
RNN to capture order in sequences



Attention for Machine Translation

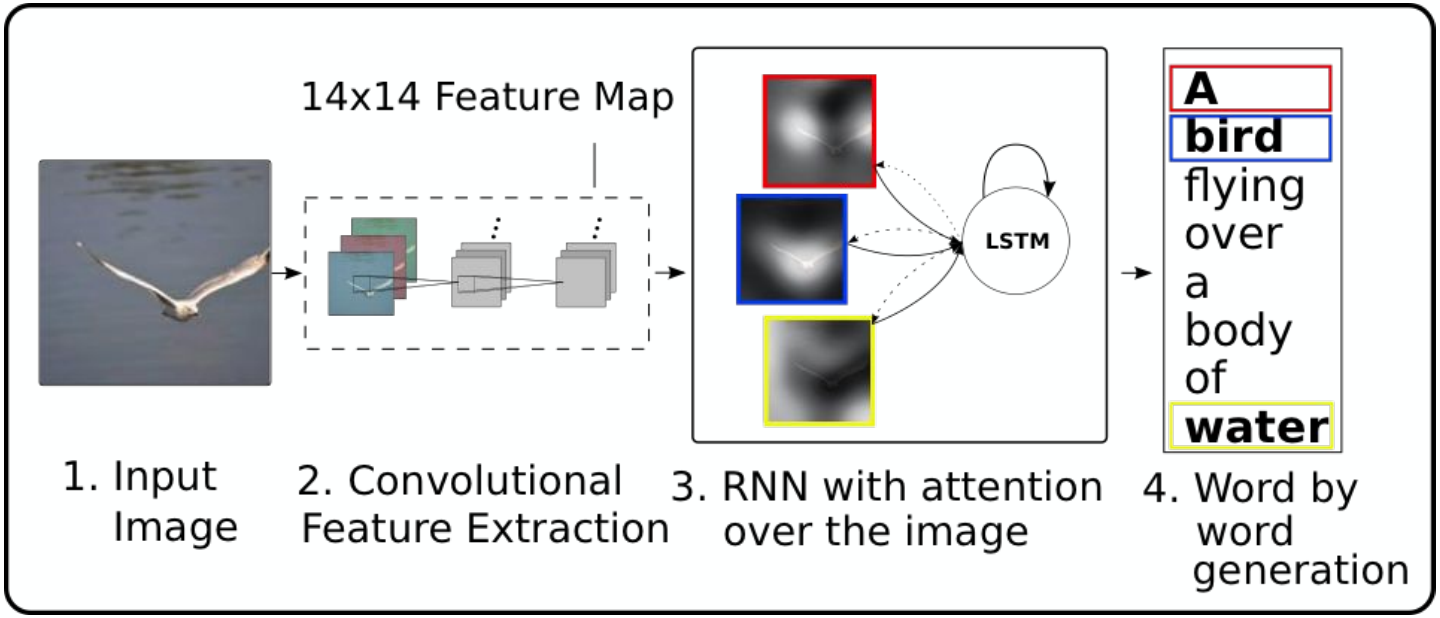
The limitations of RNNs led to attention mechanisms ⇒**transformers**⇒GPT, BERT, and the AI we have now.

➥ RNNs are the grandparent of modern AI chatbots.

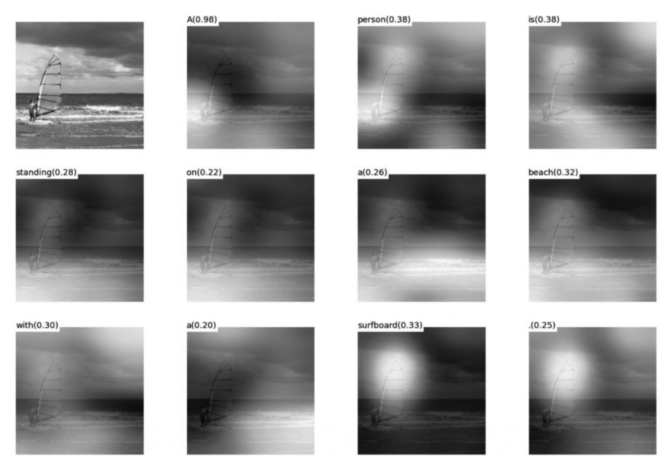


NEURAL MACHINE TRANSLATION, Bengio et al 2015

Image-to-Text: Attention for Caption Generation



Xu, Kelvin, et al. Arxiv'15



(b) A person is standing on a beach with a surfboard.

RNNs before Transformers

1. **They’ve Written Music and Poetry**

RNNs were once state-of-the-art for generating text, including **poetry, music lyrics, even Shakespearean-style sonnets**.

1. **They’ve used to Write Code**

Before transformers took over, people trained RNNs to generate code, for example, predicting the next line in Python functions.

1. **RNNs are Not Totally abandoned**

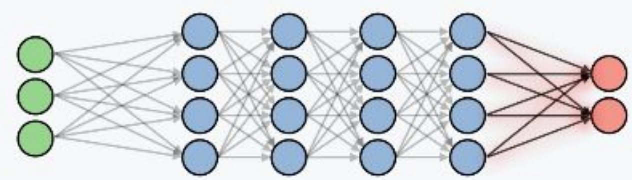
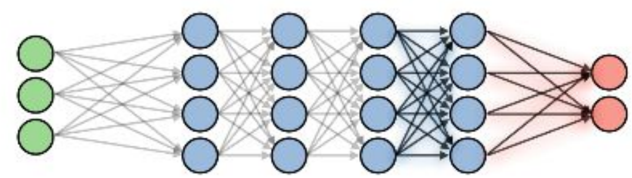
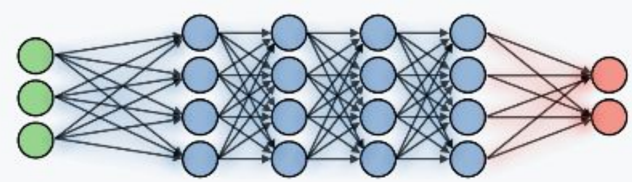
While transformers dominate NLP today, RNNs are still used in **low-latency** or **resource-constrained environments**, where smaller models are preferred.

Pretraining: Where the Story Begins (~2010)

PURDUE UNIVERSITY

Definition

- **Pretraining** is the process of training a model on a **large, general-purpose dataset** to learn **general knowledge**, that can be transferred to other tasks.
- **Fine-Tuning** is the process of adapting a pretrained model to a specific task or dataset by updating some or all of its parameters.
- **Fine-Tuning** adapts that knowledge to a **specific task** with **smaller datasets**.

Illustration	Explanation
	Freezes all layers, trains weights on softmax
	Freezes most layers, trains weights on last layers and softmax
	Trains weights on layers and softmax by initializing weights on pre-trained ones

https://www.labellerr.com/blog/fasten-up-your-data-annotation-process-with-pre-trained-models/

Pre-Training Revolution

1. Reduced Data Requirements

- Before pretraining, training deep models required **large labeled datasets** for every task (supervised learning).
- With pre training, models learn **general features** first, so they can be fine-tuned with **much less task-specific data**.
- Helps in low-resource settings (e.g., rare languages, medical images)

2. Learns Broad, Reusable Representations

- During pre training (on massive, diverse datasets), models learn **general patterns, then** The model starts with *knowledge of the world for each new task*, not from scratch.:
  - In NLP: grammar, syntax, semantics
  - In vision: edges, textures, object shapes
  - In audio: pitch, phonemes, speaker traits

3. Better initialization Enabled Larger and Deeper Models

- Without pretraining, deep models struggled with overfitting or vanishing gradients.
- Pretraining gives models a **stable foundation** to build deeper and more powerful architectures.

Layer-by-Layer Pretraining

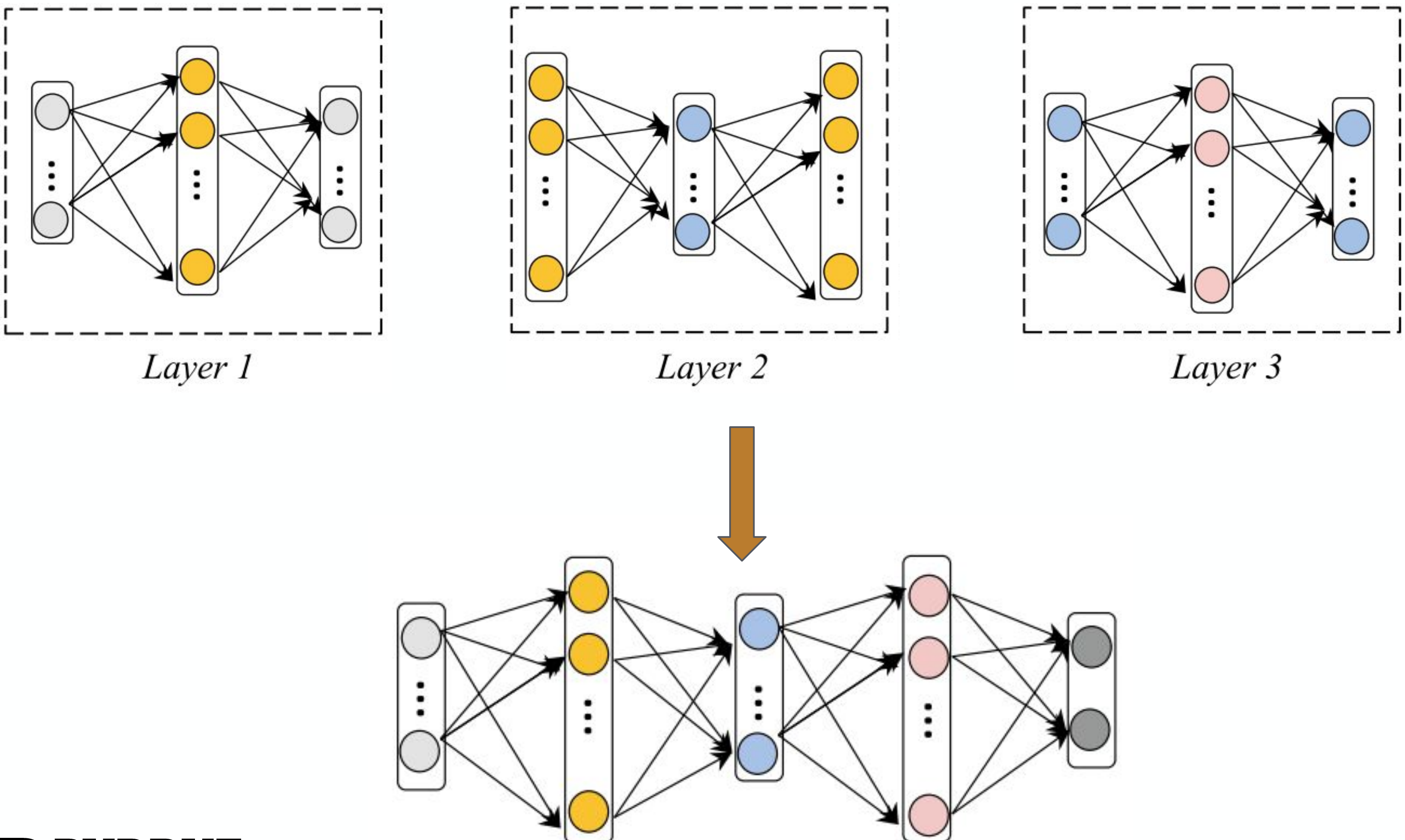
Geoffrey Hinton et al. (2006) and Yoshua Bengio (2007) introduced Layer-wise pretraining is an **unsupervised, greedy approach** where:

1. **Each layer is trained separately in an unsupervised manner** before fine-tuning the full model.
2. **Pretrained layers are stacked** progressively to build deeper networks.
3. A final **supervised fine-tuning step** is applied using labeled data.

Before this, feature extraction relied on domain knowledge (e.g., edge detectors in computer vision). deep models proved **deep models could learn useful features without manual engineering**.

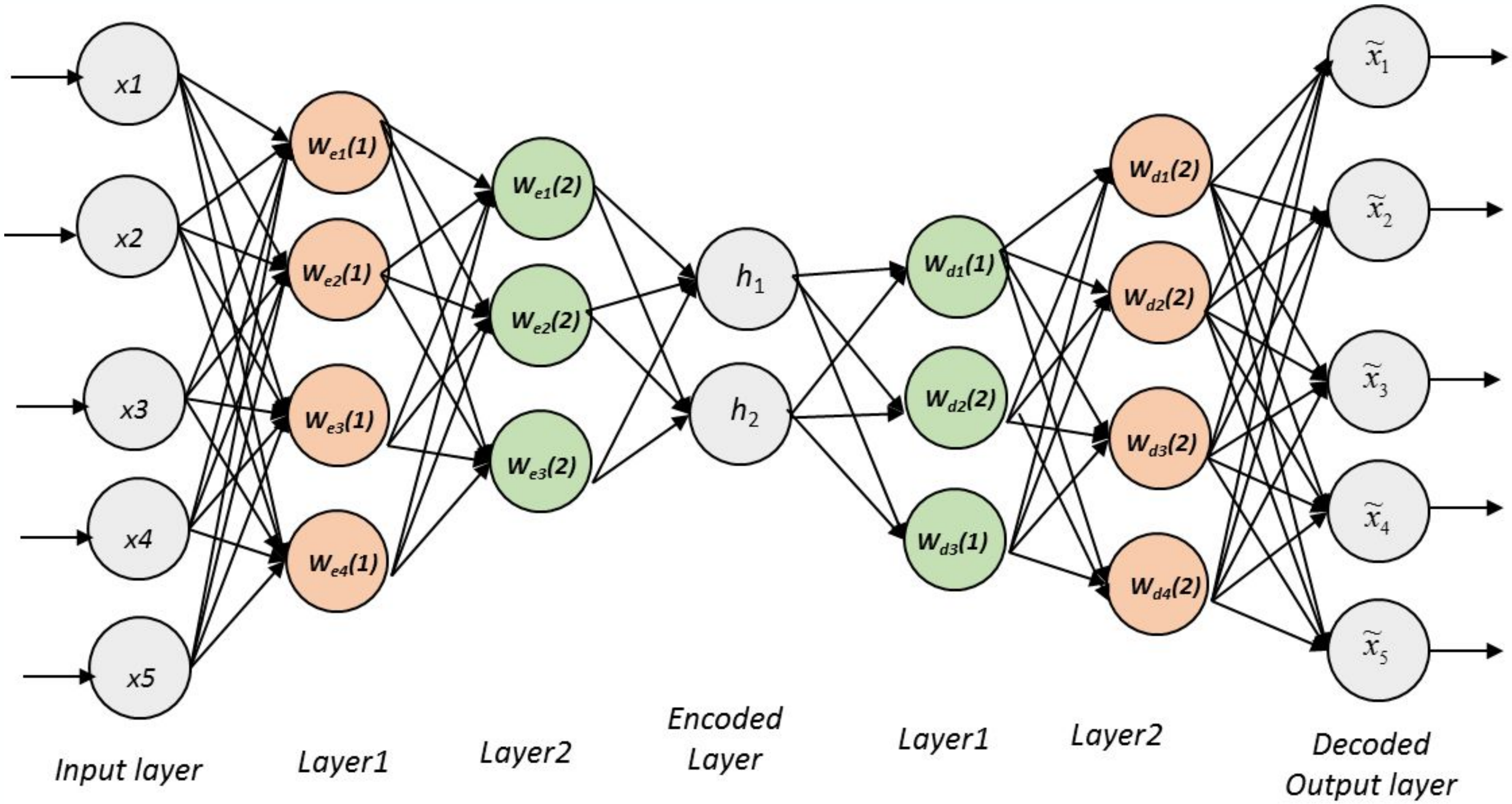
Stacked Autoencoders

Unsupervised layerwise pretraining



Stacked AutoEncoder

Stacking encoders, and stacking decoders

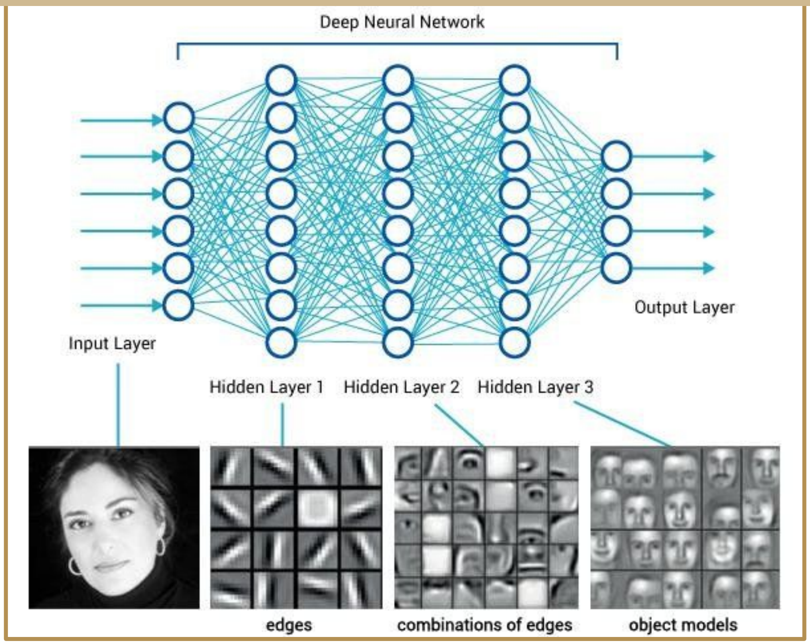


Representation Learning

Prior to Deep Learning: Traditional machine learning relied heavily on **hand-engineered features**.

- Domain experts would manually craft features from raw data (e.g. TF-IDF for test, SIFT and HOG, and Wavelet for image) ,
- Then they fed into shallow models like linear regression, SVMs, or simple feedforward neural networks.

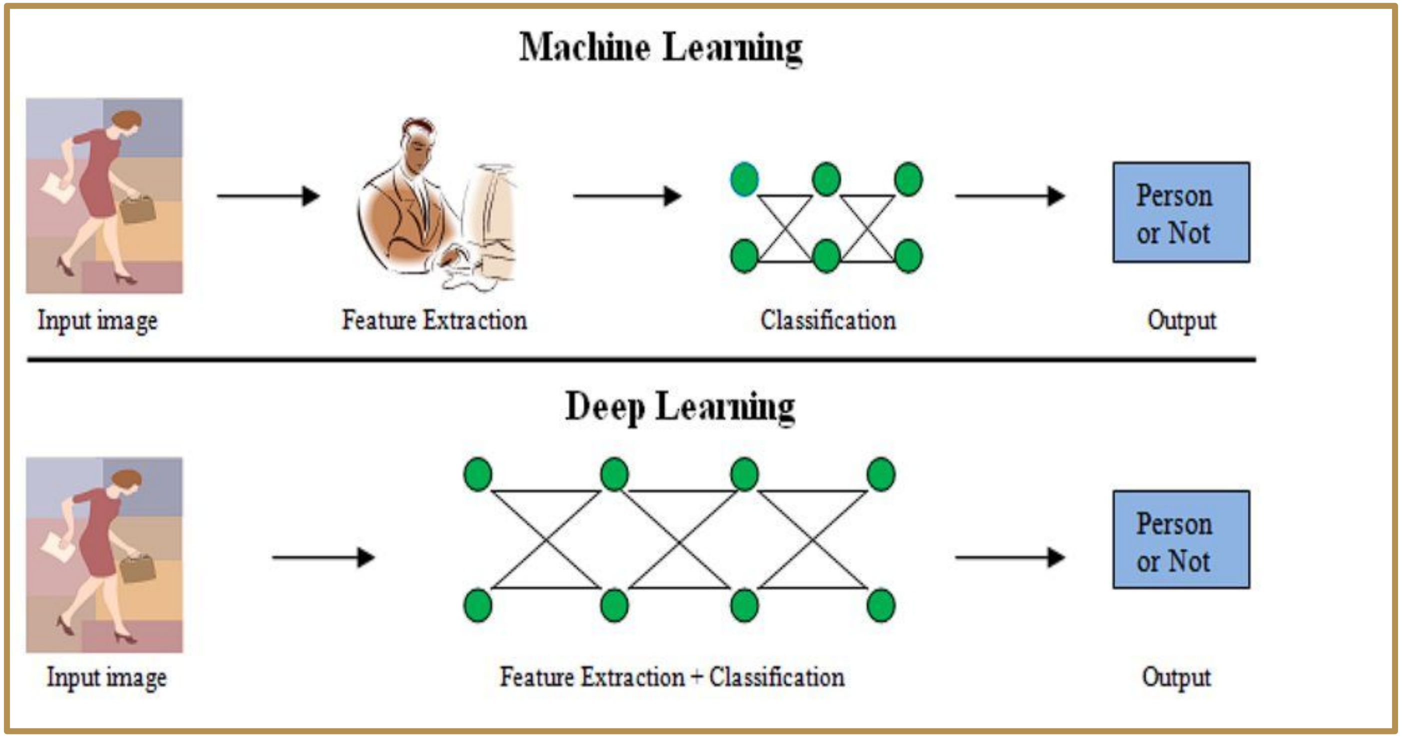
Deep learning models inherently perform hierarchical feature learning, automatically extracting **low-level features** in initial layers (e.g., edges in images)**, intermediate features** in middle layers (e.g., shapes or textures)**,** and **high-level abstractions** in deeper layers (e.g., objects or faces).



Example of feature hierarchy learned by a deep learning model on faces from Nie et al. (2019).

End-to-end Learning

- Deep learning allows models to learn end-to-end from raw data to the final prediction, **by integrating feature extraction into the model training process**.
- The entire pipeline, from raw pixels or tokens to final predictions, became a **single computational graph** where gradients could flow **end-to-end**.
- This allowed for **joint optimization**, where **features and tasks are learned together**,



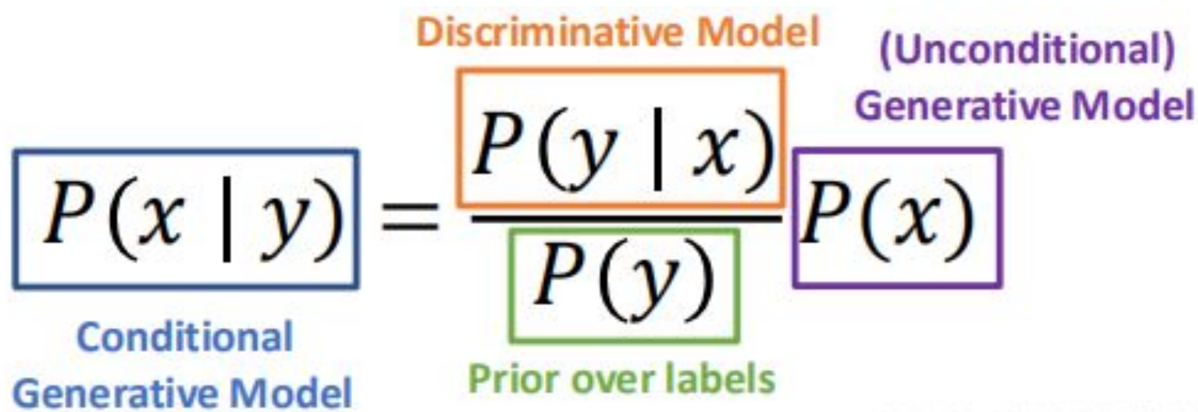
Machine Learning vs. Deep Learning, [Krishna et al, 2019]

Generative AI: How it started…

PURDUE UNIVERSITY

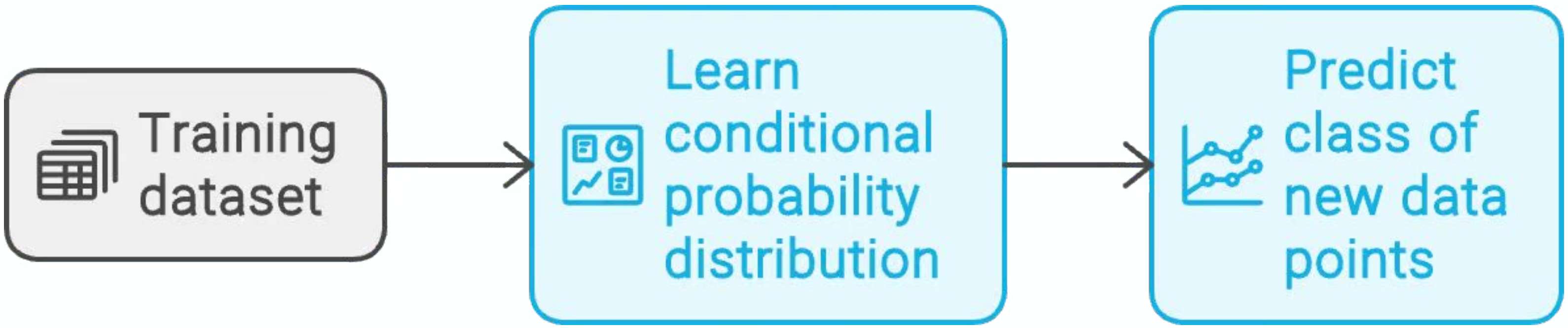
Bayes Rule

Given a training dataset consisting of data points (x), and their associated labels (y):



Discriminative and generative model workflow

A discriminative model learns the conditional probability distribution P(y|x).



A generative model learns the joint probability distribution P(x,y). It then uses this underlying distribution to generate new data similar to the training examples **or** address classification problems.

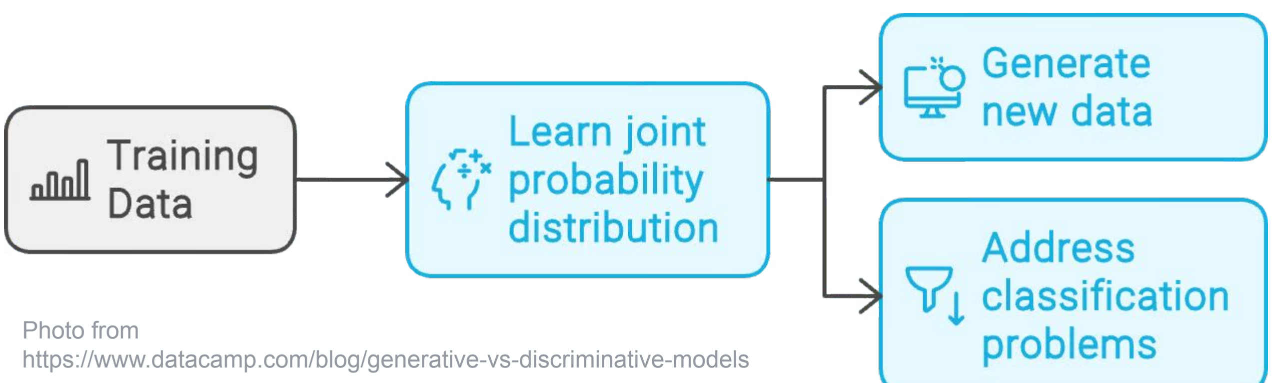
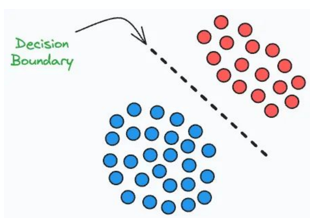
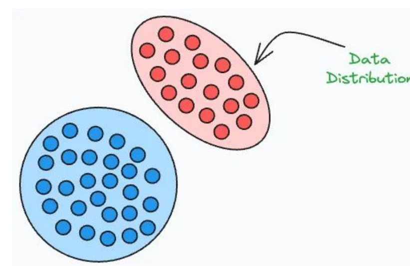
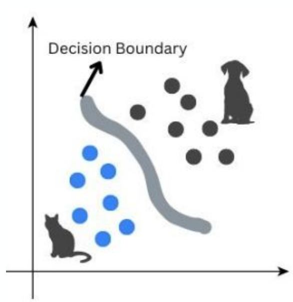
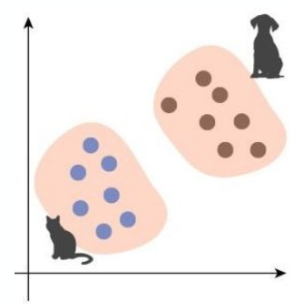


Photo from https://www.datacamp.com/blog/generative-vs-discriminative-models

Discriminative vs Generative Models

Discriminative Models	Generative Models
Directly estimate decision boundary between classes P(y|x)	Learns input distribution P(x|y) to deduce P(y|x) using bayes rule.
regression, SVM, older models like **RNN**, **CNN** and transformer models like **BERT** family are mainly discriminative.	Bayes, GDA, **GPT** Family, **Diffusion** models
	
 Classify or Label data point as cat or dog	 Produce a new data point that looks like cats or dogs

Generative vs Discriminative Learning

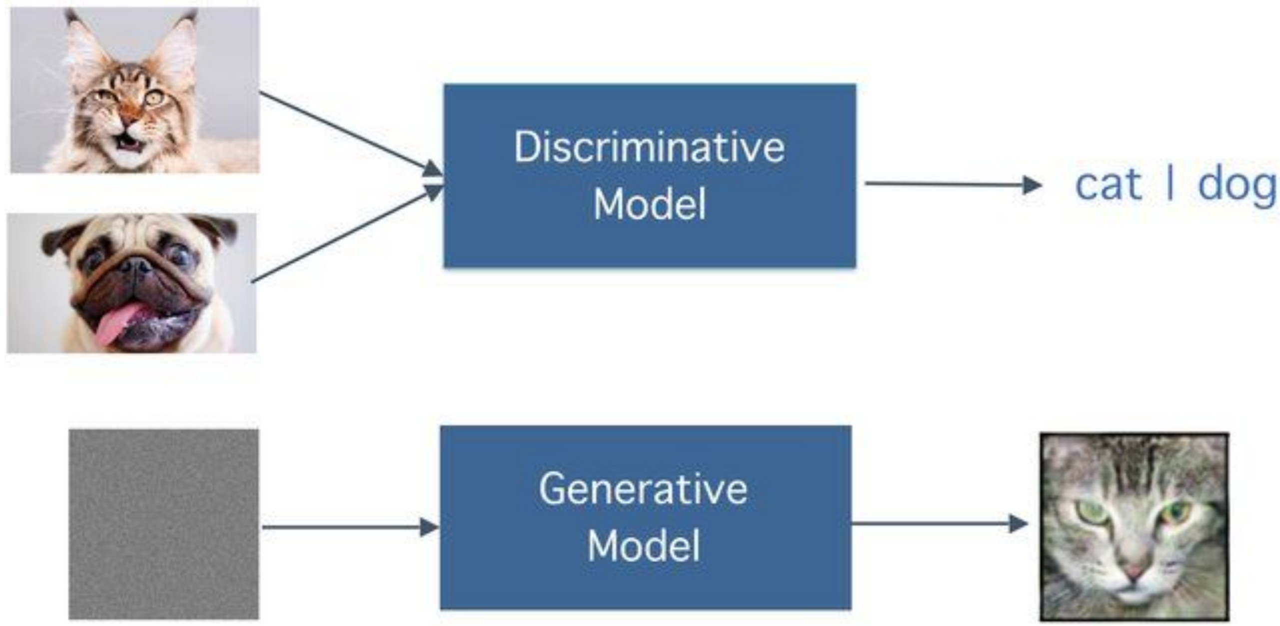
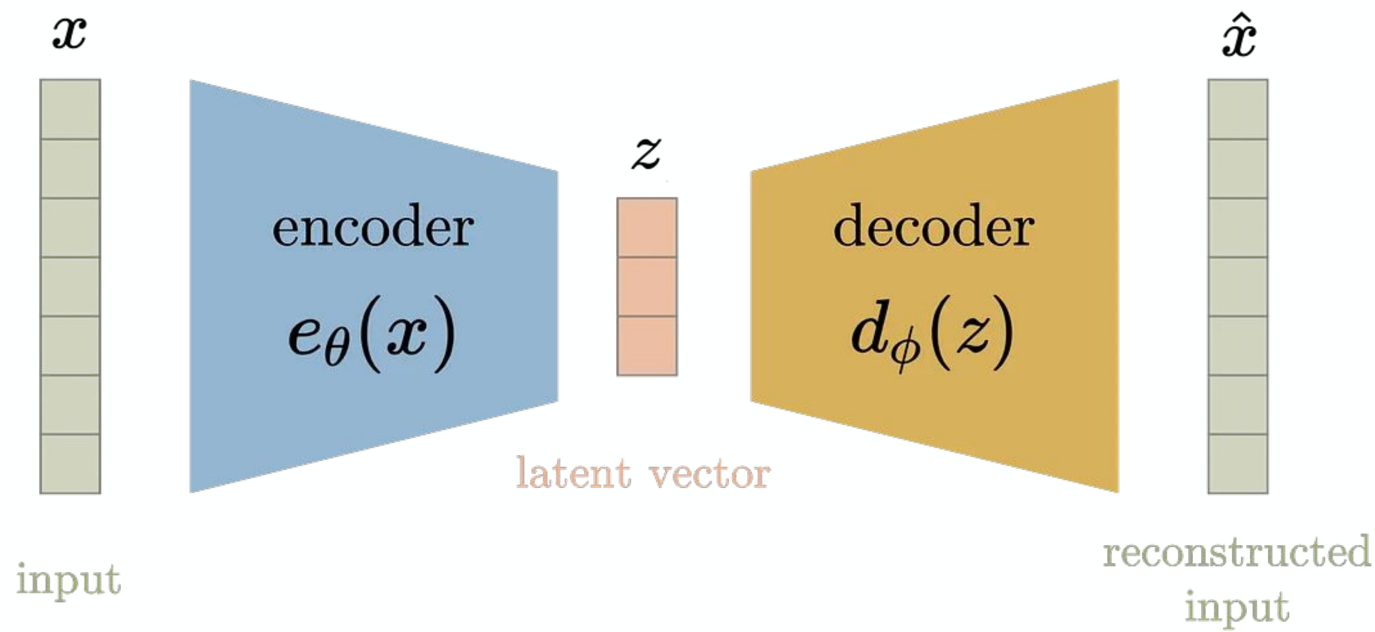


Photo from: A Critical Overview of Privacy in Machine Learning, 2021

AutoEncoders for Data Compression

- The AE is able to compress data to fewer bits essentially getting rid of the redundancy (Encoder).
- But due to non-regularized latent space AE, the decoder can not be used to generate valid input data from vectors sampled from the latent space.



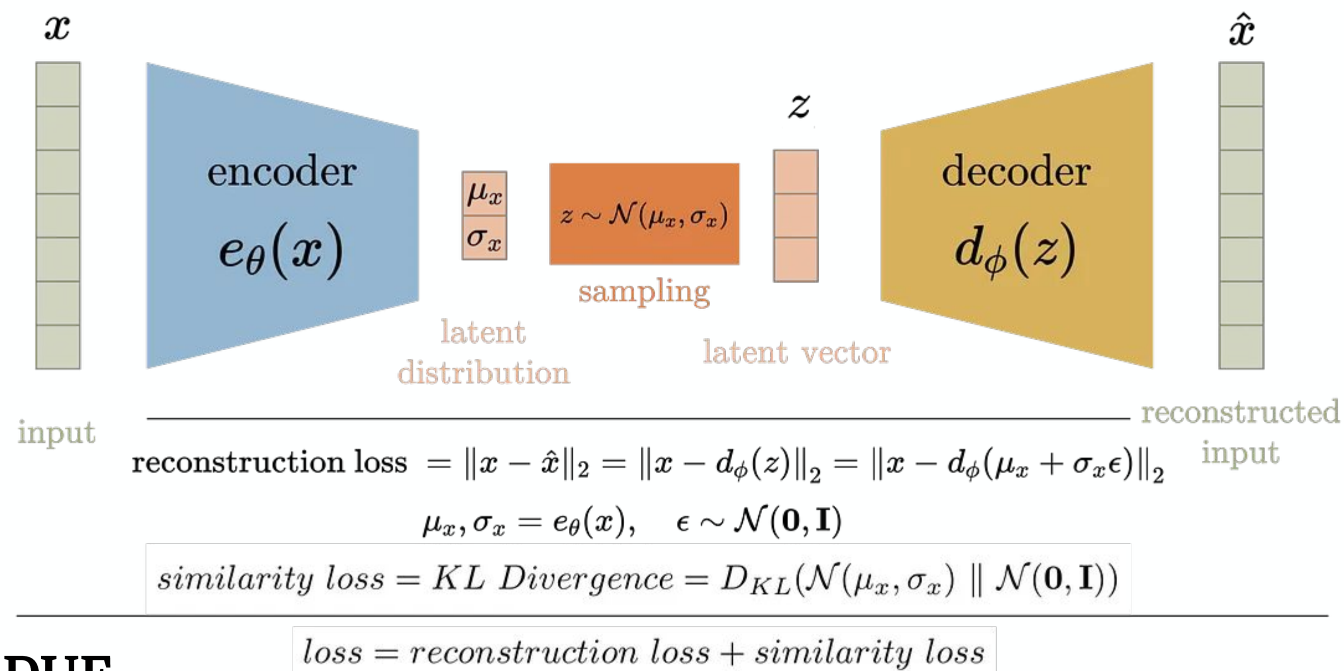
$$loss = \|x - \hat{x}\|_2 = \|x - d_\phi(z)\|_2 = \|x - d_\phi(e_\theta(x))\|_2$$

Autoencoders [2]

Variational Autoencoders, 2013

- With AutoEncoders:
  - You can't sample random points in latent space and expect valid outputs.
  - No guarantee that nearby points produce similar reconstructions.

- The encoder of VAE outputs parameters of a predefined distribution in the latent space for every input.



Rocca, J. Understanding variational autoencoders (VAES), 2021.

Variational AutoEncoder

ELBO (evidence lower bound) is a key concept in Variational Bayesian Methods. It transforms inference problems, which are always *intractable*, into optimization problems that can be solved with, for example, gradient-based methods.

From the perspective of auto-encoder, the neural network with parameters ϕ is called *encoder* because it maps from the observation space to the latent space, while the network with parameters θ is called *decoder* because it maps from the latent to the observation space.

$$
\begin{aligned}
\log p_\theta(x) &= \log \int_z p_\theta(x,z)dz \\
&= \log \int_z p_\theta(x,z)\frac{q_\phi(z|x)}{q_\phi(z|x)}dz \\
&= \log \mathbb{E}_{z\sim q_\phi(z|x)}\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right] \\
&\geq \mathbb{E}_z\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] \text{ by Jensen's inequality}
\end{aligned}
$$

$$
\begin{aligned}
&= \int_z q_\phi(z|x)\log\frac{p_\theta(x,z)}{q_\phi(z|x)}dz \\
&= \int_z q_\phi(z|x)\log\frac{p_\theta(x|z)\,p(z)}{q_\phi(z|x)}dz \\
&= \mathbb{E}_{z\sim q_\phi(z|x)}\left[p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x)\,\|\,p(z)\right)
\end{aligned}
$$

- In a well-balanced scenario, the KL divergence should be neither too high nor too low. A higher KL value indicates that the model is learning significant information about the data that deviates from the prior, while a value approaching zero means less information is being encoded.

The KL divergence term can be interpreted as a measure of the additional information required to express the *posterior* relative to the *prior*. As it approaches zero, the posterior is fully obtainable from the *prior*.

PURDUE UNIVERSITY

Variational AutoEncoder

Theory

ELBO (evidence lower bound) is a key concept in Variational Bayesian Methods. It transforms inference problems, which are always *intractable*, into optimization problems that can be solved with, for example, gradient-based methods.

$$
\begin{aligned}
&\ln p(x) \\
&= \ln \int_z p(x,z) \\
&= \ln \int_z p(x,z) \frac{q(z|x)}{q(z|x)} \\
&\geq \mathbb{E}_{q(z|x)}[\ln \frac{p(x,z)}{q(z|x)}] \\
&= \mathbb{E}_{q(z|x)}[\ln \frac{p(x|z)p(z)}{q(z|x)}] \\
&= \mathbb{E}_{q(z|x)}[\ln p(x|z)] + \mathbb{E}_{q(z|x)}[\ln \frac{p(z)}{q(z|x)}] \\
&= \mathbb{E}_{q(z|x)}[\ln p(x|z)] + \int_z q(z|x) \ln \frac{p(z)}{q(z|x)} \\
&= \mathbb{E}_{q(z|x)}[\ln p(x|z)] - D_{KL}[q(z|x)||p(z)] \\
&= likelihood - KL
\end{aligned}
$$

VAE example

When decoding from the latent state, we'll randomly sample from each latent state distribution to generate a vector as input for our decoder model.

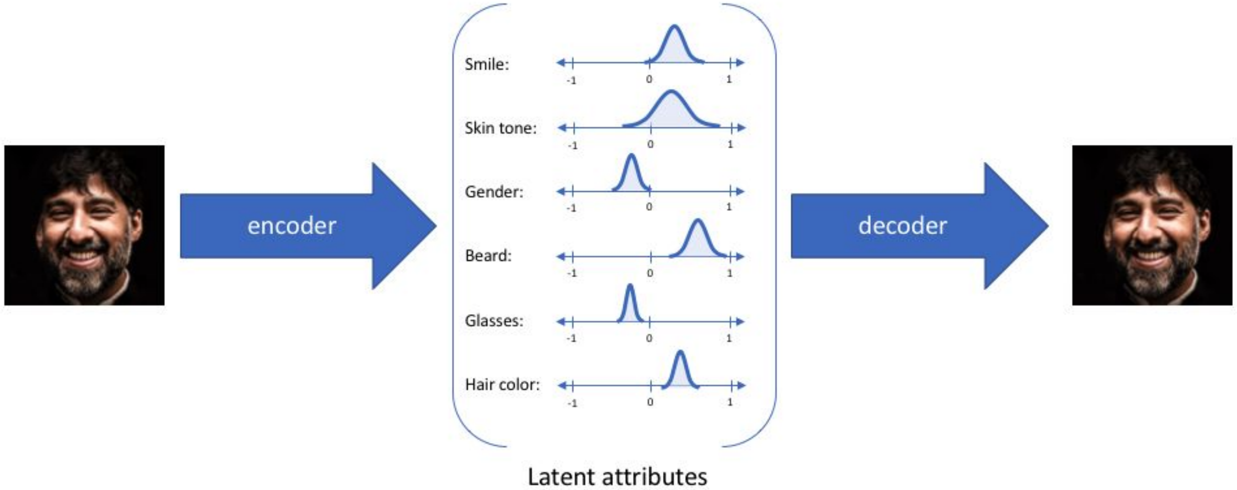
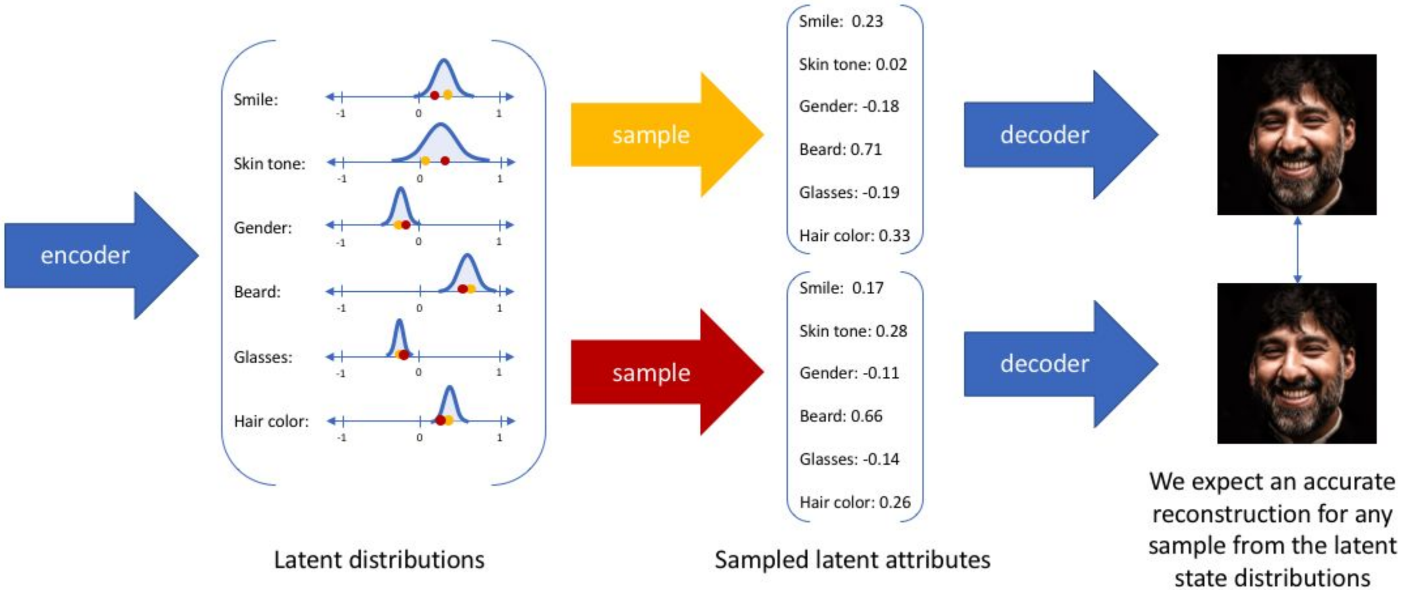


Photo from [6]

VAE example

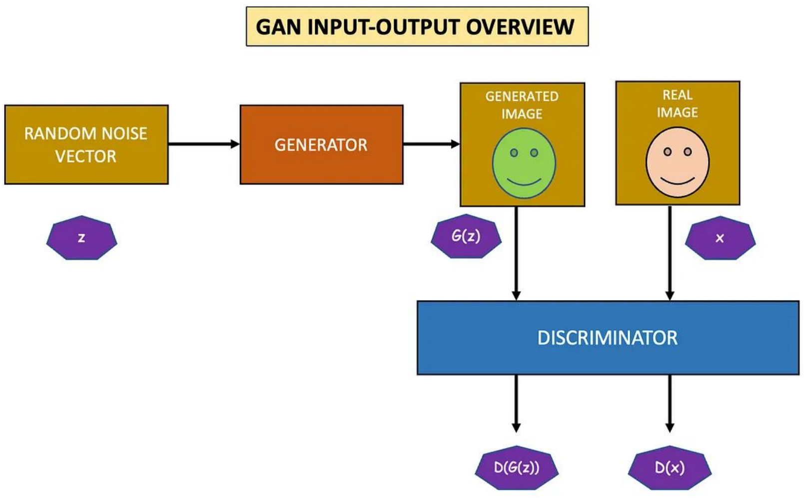
Values which are nearby to one another in latent space should correspond with very similar reconstructions.



PURDUE UNIVERSITY

Generative Adversarial Networks (GANs, 2014)

- VAE suffers from Blurry Image Generation and **Mode Collapse (Overly Averaged Samples)** due to Gaussian assumption**.**
- GANs pit a **generating neural network** that creates realistic content against a **discriminating neural network** for detecting fake content.



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Photo from:
https://towardsdatascience.com/decoding-the-basic-math-in-gan-simplified-version-6fb6b079793/

PURDUE UNIVERSITY

Diffusion Models: Dall-E, Stable Diffusion

- One issue with GANs is that they can suffer from mode collapse in which the generator produces **limited and repetitive outputs**, making them difficult to train.
- GANs are also **hard to optimize and stabilize**, and there is no explicit control over the generated samples.

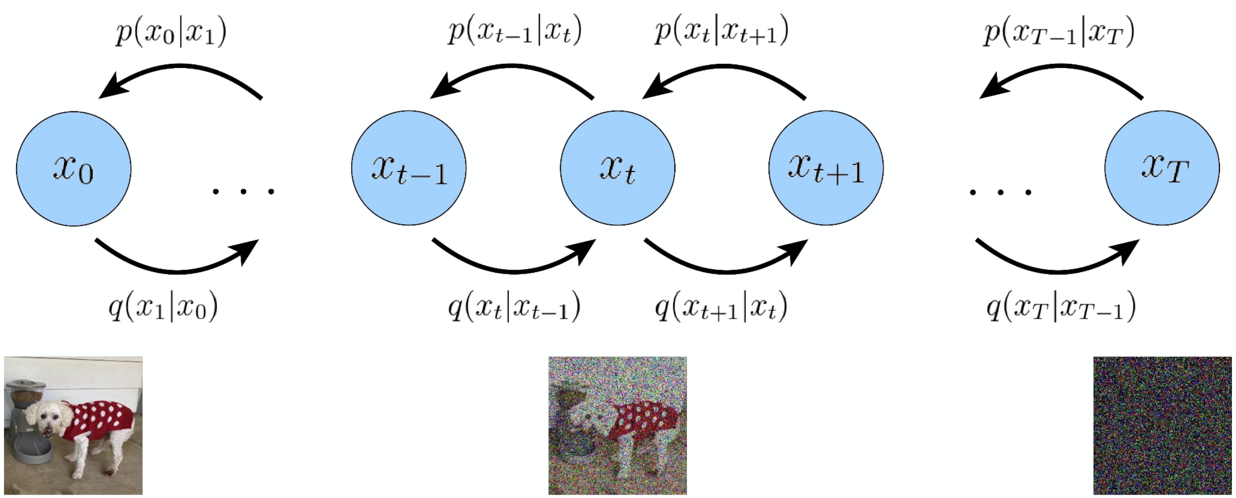
PURDUE UNIVERSITY

Diffusion Models

Given a data point sampled from a real data distribution $x_0 \sim q(x)$, let us define a *forward diffusion process* in which we add small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples
$x_1,\dots,x_T$. The step sizes are controlled by a variance schedule $\beta_t \in (0,1)$

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

The data sample x_0 gradually loses its distinguishable features as the step t becomes larger. Eventually when $T \to \infty$, x_T is equivalent to an isotropic Gaussian distribution.



The Markov chain of forward (reverse) diffusion process of generating a sample by slowly adding (removing) noise, https://calvinyluo.com/2022/08/26/diffusion-tutorial.html

References

1. Krishna, S.T. and Kalluri, H.K., 2019. Deep learning and transfer learning approaches for image classification. International Journal of Recent Technology and Engineering (IJRTE), 7(5S4), pp.427-432.
2. https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2
3. https://medium.com/@rushikesh.shende/autoencoders-variational-autoencoders-vae-and-%CE%B2-vae-ceba9998773d
4. Saha, M., Mitra, P. and Nanjundiah, R.S., 2017. Deep learning for predicting the monsoon over the homogeneous regions of India. Journal of earth system science, 126, pp.1-18.
5. https://yunfanj.com/blog/2021/01/11/ELBO.html
6. https://www.jeremyjordan.me/variational-autoencoders/
7. https://www.microsoft.com/en-us/research/blog/how-can-generative-adversarial-networks-learn-real-life-distributions-easily/
8. https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
9. https://pub.towardsai.net/diffusion-models-vs-gans-vs-vaes-comparison-of-deep-generative-models-67ab93e0d9ae
10. https://magazine.sebastianraschka.com/p/understanding-encoder-and-decoder
11. https://cameronrwolfe.substack.com/p/language-model-training-and-inference
12. https://x.com/cwolferesearch/status/1689388468911132672
13. https://twitter.com/cwolferesearch/status/1671628210180698112?s=20
14. https://twitter.com/cwolferesearch/status/1659608476455256078?s=20
15. https://twitter.com/cwolferesearch/status/1692617211205022064?s=20
16. https://docs.cohere.com/docs/controlling-generation-with-top-k-top-p
17. https://x.com/cwolferesearch/status/1766180825173803516
18. Wang, W., Chen, W., Luo, Y., Long, Y., Lin, Z., Zhang, L., Lin, B., Cai, D. and He, X., 2024. Model compression and efficient inference for large language models: A survey. *arXiv preprint arXiv:2402.09748*.
19. https://medium.com/@eugene-s/unleashing-the-potential-of-large-language-models-llms-with-chatgpt-8210f0cb063d
20. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), pp.1-67.

Thank You

PURDUE
UNIVERSITY®