Generative AI Applied to Telecommunications

Aldebaro Klautau, Pedro Sousa, Cláudio Modesto, João Borges, Felipe Bastos and Cleverson Nahum

SBrT24 – Minicurso - 01/out/2024 - https://sbrt2024.sbrt.org.br/

Code available at https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

LASSE 5G/6G & IoT Research Group

www.lasse.ufpa.br

Universidade Federal do Pará (UFPA)

Generative AI Applied to Telecommunications

QR code to download the slides:

Feel free to download and go over the **slides**! You may also check the associated tutorial **software**, made available at: https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

Extra!

*Jupyter shortcuts, magic and shell commands

Two different keyboard input modes:

- **Edit** mode: type code or text into a cell.
  Green cell border
- **Command** mode: notebook level commands. Gray cell border with a blue left margin

List of shortcuts:

Go to menu: Help □ Keyboard Shortcuts

Shortcuts that work in both edit and command modes:

Shift + Enter - run the current cell, select below
Ctrl + Enter - run selected cells
Alt + Enter - run the current cell, insert below
Ctrl + S - save and checkpoint

Magic commands:

%matplotlib inline - **Display matplotlib graphs in notebook**
%run ‹file name› - **Run a file**
%%time - **Get an execution time**
%who - **List all variables**
%pinfo ‹variable› - **Get detailed information about variable**
%env – **List all environment variables**
%load_ext autoreload – **Reload modules**
%pip – **Install in current kernel (instead initially launched)**
%conda – **Install in current kernel (instead initial)**

Shell commands in IPython / Jupyter:

Use the ! character as prefix to the command. For instance:
!ls (on Linux)
!dir (on Windows)

[1] https://towardsdatascience.com/jypyter-notebook-shortcuts-bf0101a98330
[2] https://towardsdatascience.com/top-8-magic-commands-in-jupyter-notebook-c1582e813560
[3] https://jakevdp.github.io/PythonDataScienceHandbook/01.05-ipython-and-shell-commands.html

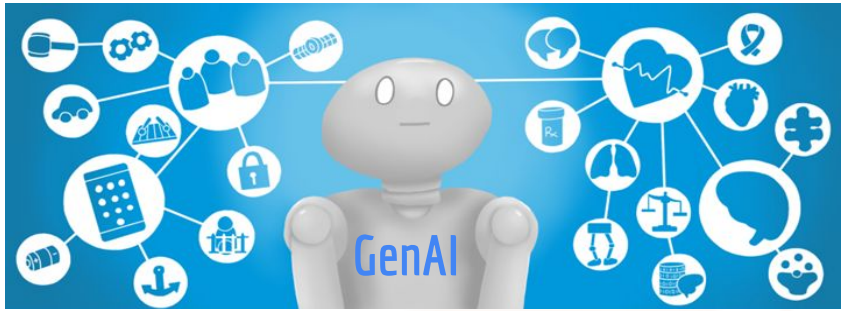
Scope and guidelines

- We will discuss the **intuition** behind the main GenAI methods, applied to **diverse telecom applications** presented in papers from a **variety of authors**, not only ours
- We describe the main **modeling techniques** and **mathematical tools**, but we have to omit details due to the limited time. **Jupyter** notebooks with **Python** code and **extra slides** are shared to help interested people get started and dig deeper

- Telecom is a vast area, and GenAI is flourishing in many systems. For instance: *OpticGAI: Generative AI-aided Deep Reinforcement Learning for Optical Networks Optimization*, Siyuan Li et al, Jun 2024, https://arxiv.org/abs/2406.15906. But this tutorial focuses on **wireless communications** and, more specifically, 3GPP-related technology
- Until Oct. 11, 2024, feel free to discuss and ask questions using lasse.courses@gmail.com

The maximum entropy team

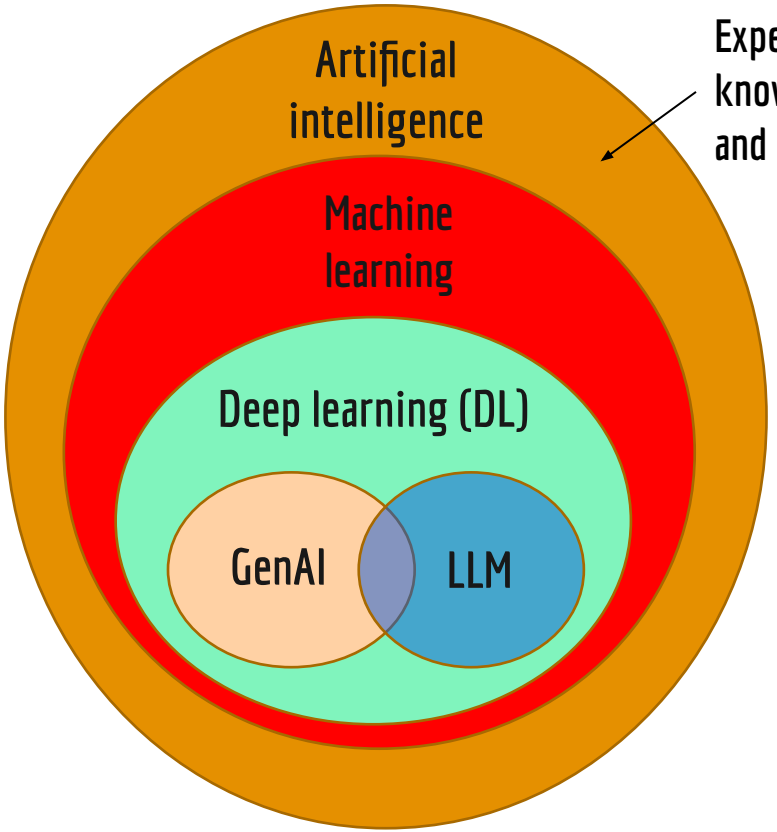
Agenda



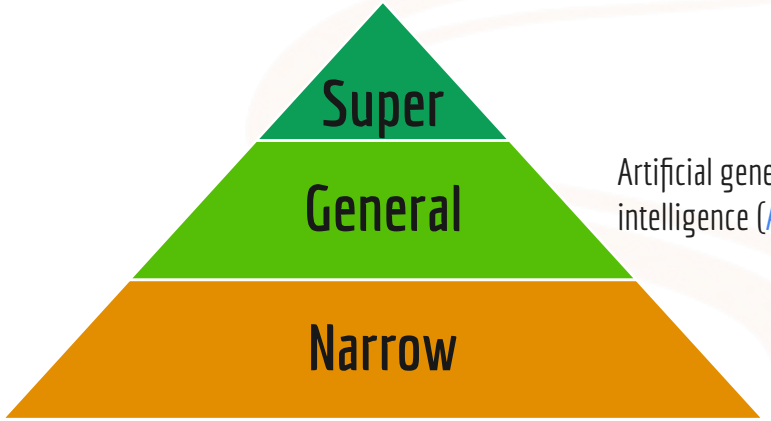
- Part I - Introduction
  - Motivation
  - Definitions related to GenAI
  - Overview of deep learning concepts
  - Overview of wireless communications concepts
- Part II - GenAI Applied to RAN (Use Cases Chosen by 3GPP)
  - Generative Adversarial Network (GAN)
  - Variational Autoencoder (VAE)
  - Diffusion Model (DM)
- Part III - Foundations of Transformers and LLMs
  - Recurrent networks (RNNs)
  - Embeddings
  - Attention
  - Transformers and an application in telecommunications
  - Large Language Models (LLMs)
- Part IV - Transformers and LLMs Applied to Telecommunications
  - Transformers and LLMs applied to RAN (Use Cases Chosen by 3GPP)
  - LLMs applied to the telecom domain
- Part V - Conclusions

Part I - Introduction

Generative artificial intelligence (GenAI) context



Expert systems, symbolic AI, knowledge representation and reasoning, planning, etc.



Artificial general intelligence (AGI)

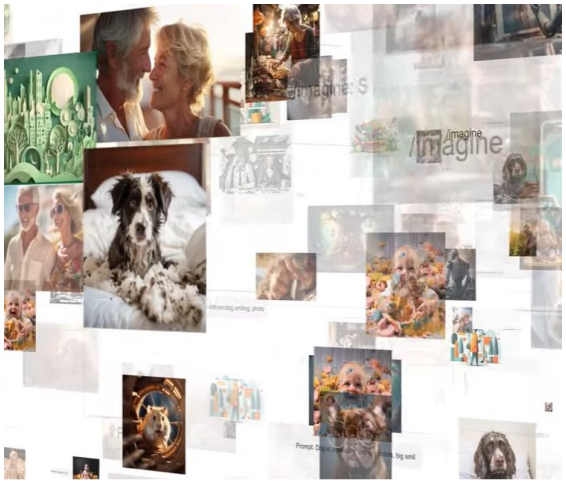
GenAI and LLMs are bringing us closer to General AI

Few examples of generative AI

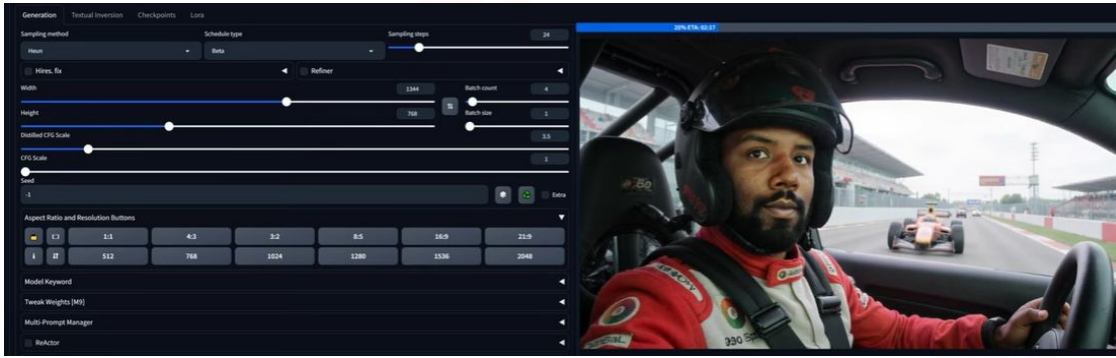
Extra!

Examples of generative AI
Multimodal large language models

Analyze images with open source NVIDA's 'Eagle Eye'



Generate a photo with your own face: Flux + LoRA (Flux Gym)



LoRA: Low-Rank Adaptation - https://arxiv.org/pdf/2106.09685

https://www.youtube.com/watch?v=0-md0S9GDJA https://www.youtube.com/watch?v=1q20FmYR-7k
https://www.youtube.com/watch?v=wxG9ZDp7qpw
https://huggingface.co/XLabs-AI/flux-lora-collection
https://flux-ai.io/
https://github.com/cocktailpeanut/fluxgym

Examples of generative AI

Google's "*GameNGen*" (VALEVSKI, Dani et al. Diffusion Models Are Real-Time Game Engines, 2024)

All scenarios were generated by AI

GameNGen is a Generative Diffusion Model

It is an augmented version of the open source Stable Diffusion v1.4

The model learns to generate frames of the game based on several runs executed by a Reinforcement Learning (RL) agent that plays it

With this data, **GameNGen** learns to model, for example, interactions sent (e.g. player actions) and received (e.g. take damage), and also persist states (e.g. ammo, health, etc.) for some time

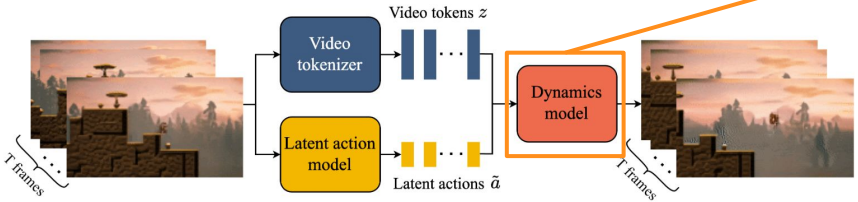
Website: https://gamengen.github.io/
Paper: https://arxiv.org/abs/2408.14837
Video: https://www.youtube.com/watch?v=O3616ZFGpqw

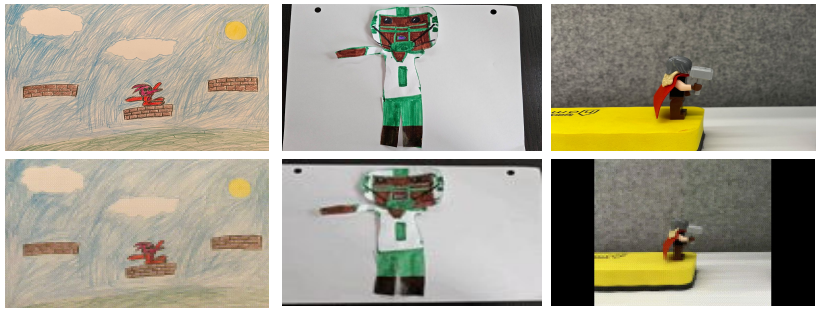
Examples of generative AI
Google's DeepMind "*Genie*" (BRUCE, Jake et al. Genie: Generative interactive environments, 2024)

Genie (Generative Interactive Environments) generates action-controllable worlds from a variety of inputs (e.g. artificially generated images, photos or sketches)

It trains on videos obtained from the internet, tokenizing them and attributing inferred actions, before sending to a **dynamics model**, which is a "decoder-only MaskGIT" transformer (for MaskGIT see CHANG, 2022)

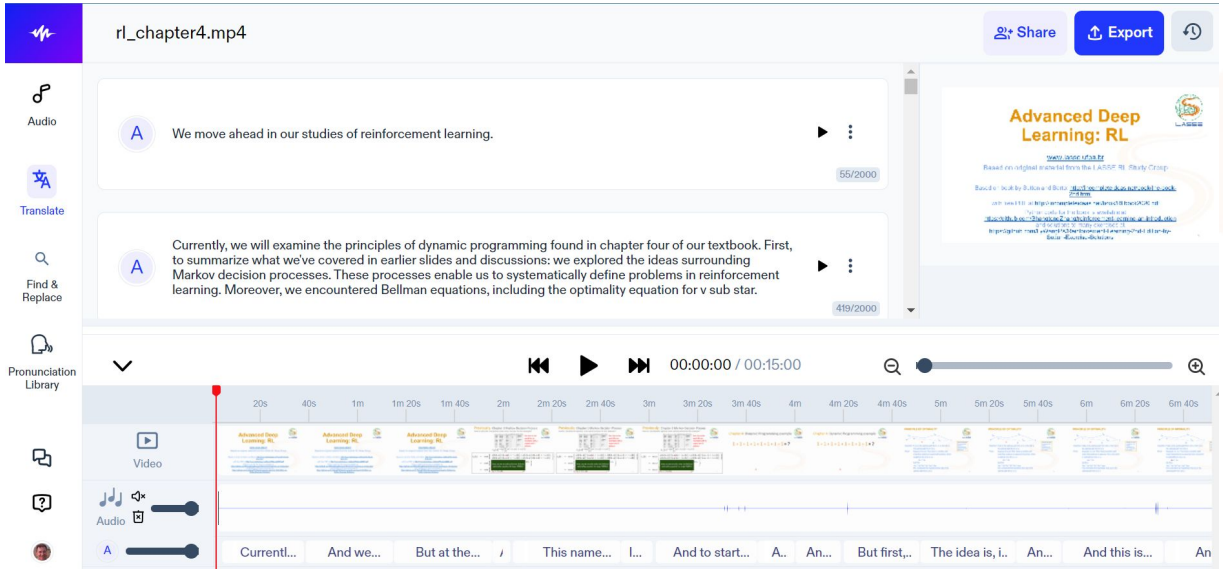
(CHANG, 2022) CHANG, Huiwen et al. MaskGIT: Masked Generative Image Transformer, 2022





Website: https://sites.google.com/view/genie-2024/
Paper: https://arxiv.org/abs/2402.15391

Examples of generative AI
Audio processing (translation, dubbing, etc.) by Speechfy



From a recording in a given language, generate another recording using a different language (Chinese, English, etc.) and voice (your own voice, other gender, etc.)

Go to: https://nextcloud.lasseufpa.org/s/3eF9mXnHa6jRH6x and watch the original lecture (original_rl_chapter4_aldebaro.mp4) and its new version (female_rl_chapter4_speechfy_english.mp4) that used a female voice

https://speechify.com/blog/turn-audio-into-different-languages/

Extra!

Examples of generative AI

Video to video generation

Runway's Video to Video Gen-3 Alpha

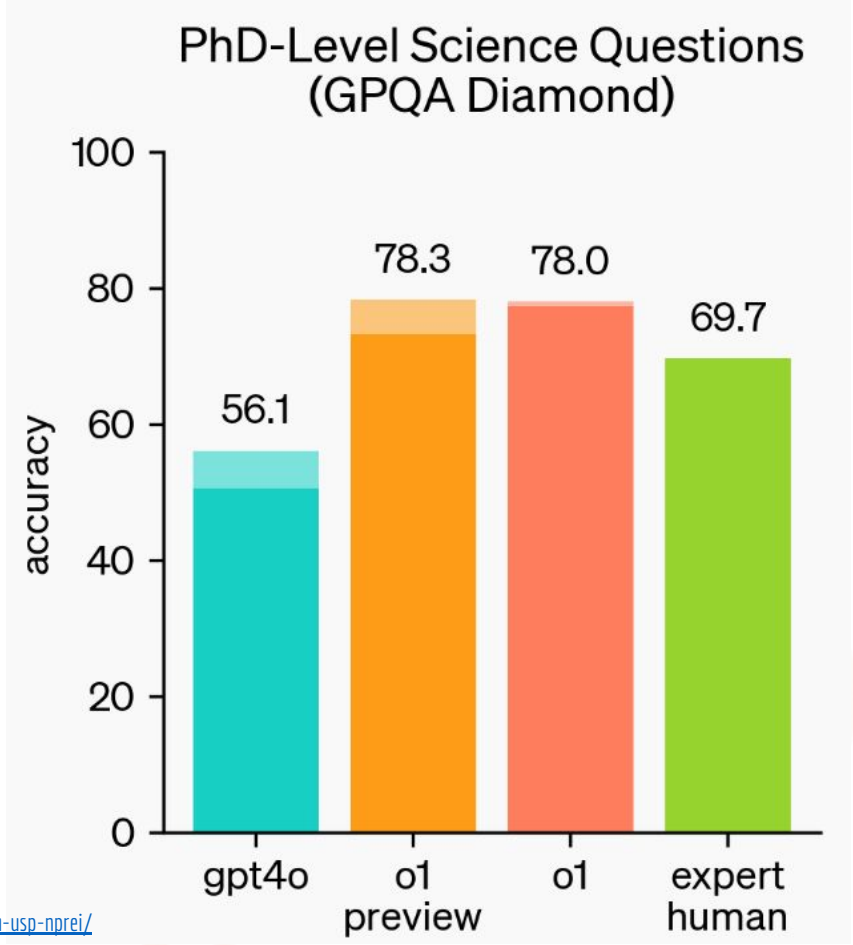
original video

screenshots of generated videos

https://www.youtube.com/watch?v=B_RfJ1B5wME&ab_channel=Runway
https://www.youtube.com/watch?v=gyCg0yv3Njw&ab_channel=CuriousRefuge

OpenAI o1 model: trained with reinforcement learning to perform reasoning

OpenAI o1 correctly solved all problems of ITA's 2024 math exam and would be approved at the "residência médica" USP-SP 2024 exam with 82% of accuracy

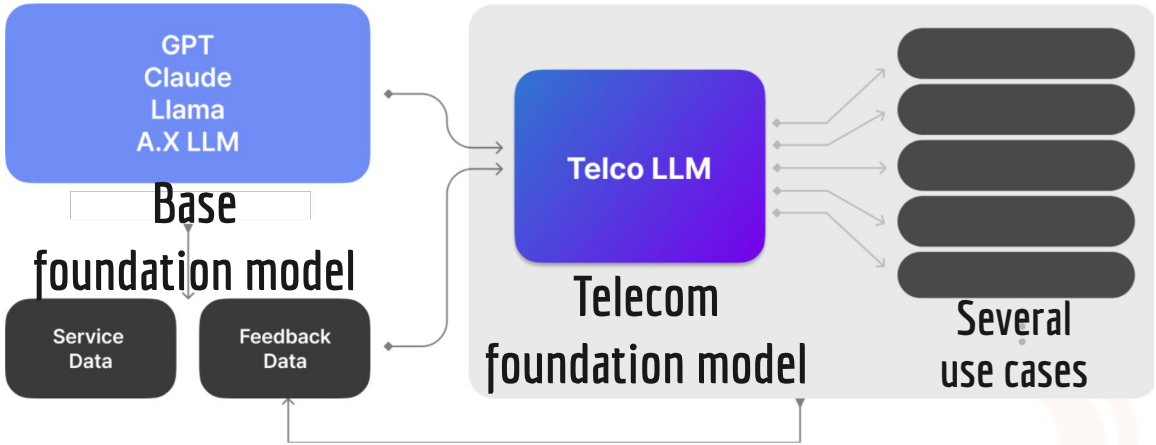


https://www.estadao.com.br/link/inovacao/nova-ia-da-openai-tira-10-em-prova-do-ita-e-passa-em-residencia-medica-da-usp-nprei/
https://arxiv.org/pdf/2303.08774
https://openai.com/index/learning-to-reason-with-llms

Generative AI (and AI) in mobile communications

GenAI on 3GPP Mobile Communications

1) Most direct application of GenAI: LLMs applied to telecom

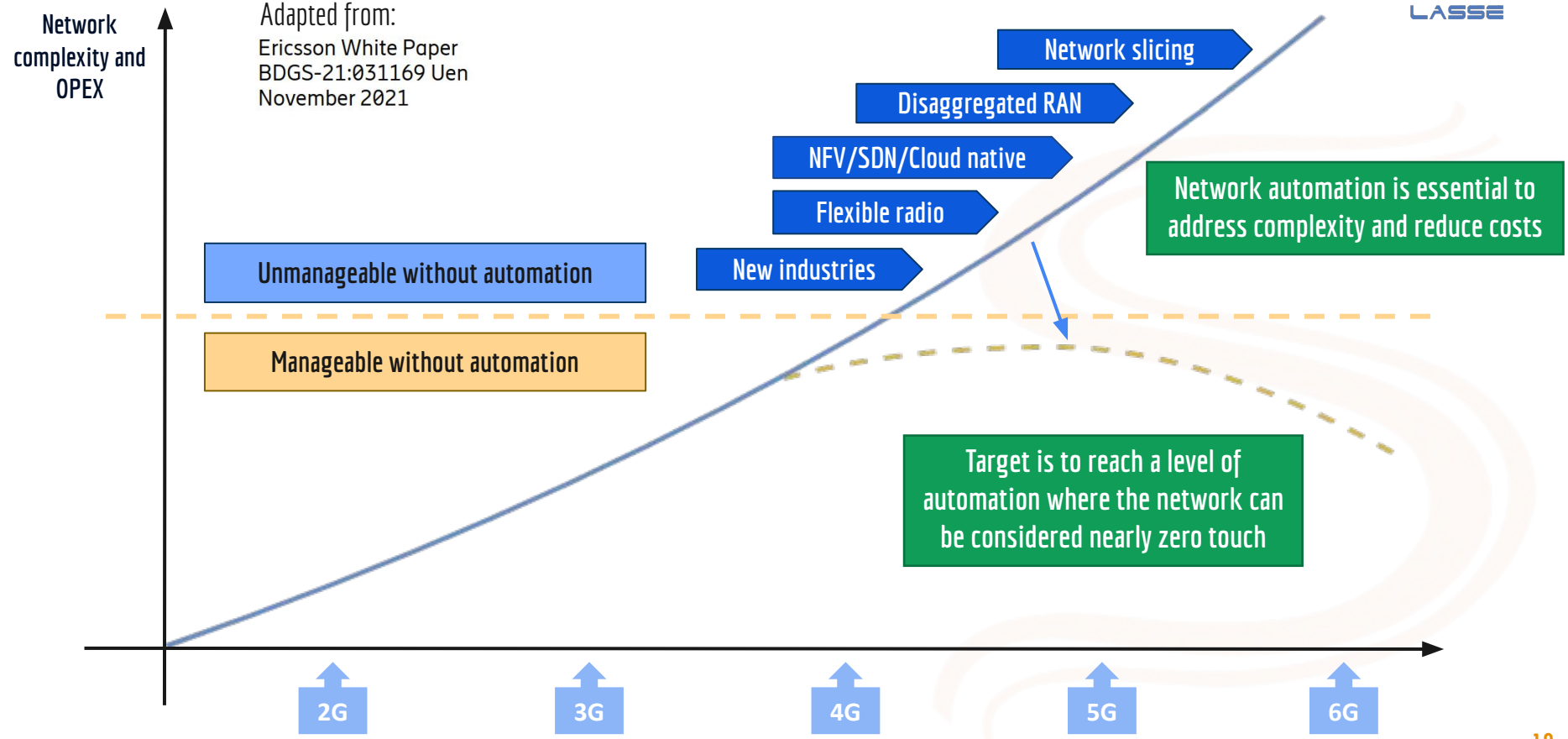


2) GenAI is also discussed in this tutorial within specific radio access network (RAN) applications that require domain knowledge

To understand GenAI in 3GPP we need to put AI in context:
from **proprietary solutions in Release 17** [1] to **native-AI in 5G Advanced**.
Study item in Release 18 identified three use cases [2]: **CSI feedback**, **beam management** and positioning

[1 Xingqin Lin et al,]Fueling the Next Quantum Leap in Cellular Networks: Embracing AI in 5G Evolution towards 6G, 2021
[2] 3GPP RP-213599 - New SI: Study on Artificial Intelligence (AI)/Machine Learning (ML) for NR Air Interface, 2021

The need for AI in the whole 5G / 6G network

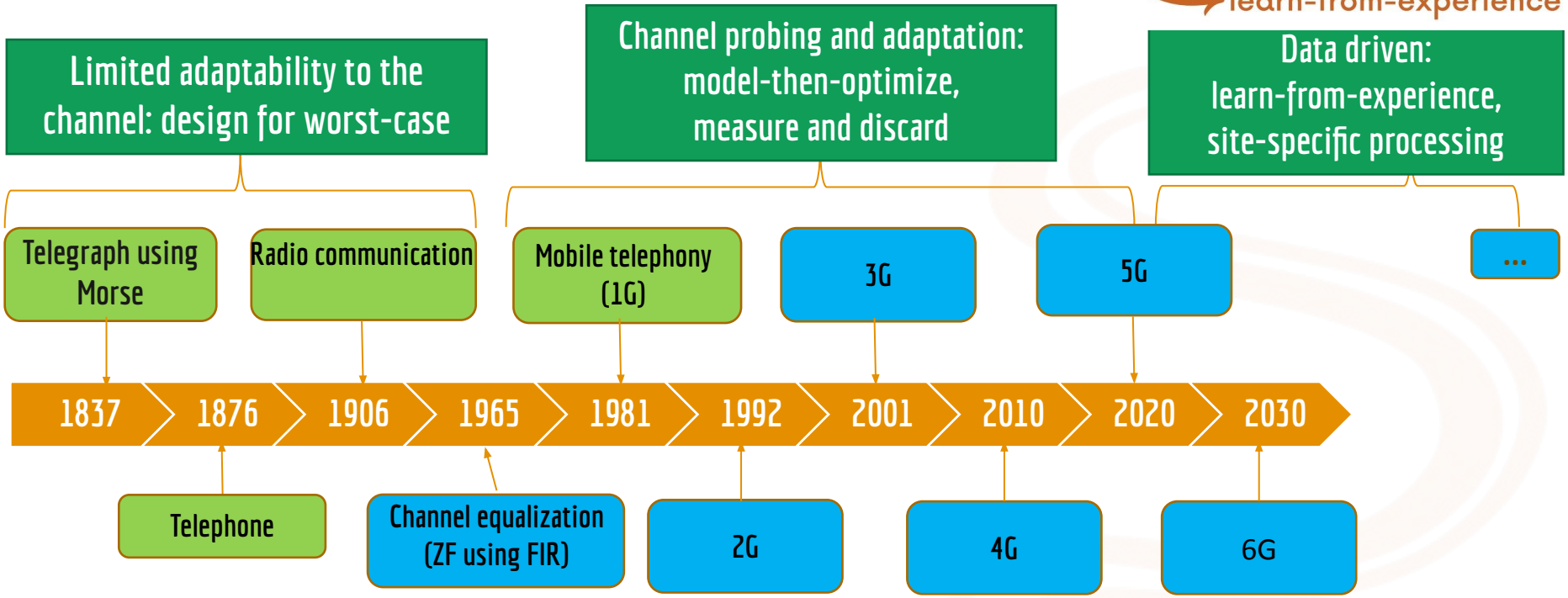


https://www.ericsson.com/en/reports-and-papers/white-papers/smo-enabling-intelligent-ran-operations

Extra!

Right moment to work in AI / ML / GenAI for telecommunications

model-then-optimize
data driven
measure-and-discard
learn-from-experience



AI is hot: Standardization bodies and consortia discussing AI/ ML

O-RAN Software Community (SC): 3.1M lines of code

ITU

- Architectural Framework for ML Rec. Y.3172
- Network automation and resource adaptation Rec. Y.3177

3GPP

- Network Data Analytics Function (NWDAF) TR 23.791
- Analytics in 5G Core TS 23.288

ETSI

- Experiential Networked Intelligence (ENI) and Open Source MANO (OSM)
- Zero-Touch Network and Service Management (ZSM)

Linux Foundation

- AI in Open Network Automation Platform (**ONAP**)
- ML and open data platforms (ACUMOS, etc)

O-RAN Alliance

- RAN Intelligent contr oller (RIC): Near-RT and Non-RT RICs
- Data-driven workflows for closed-control loops

ONF

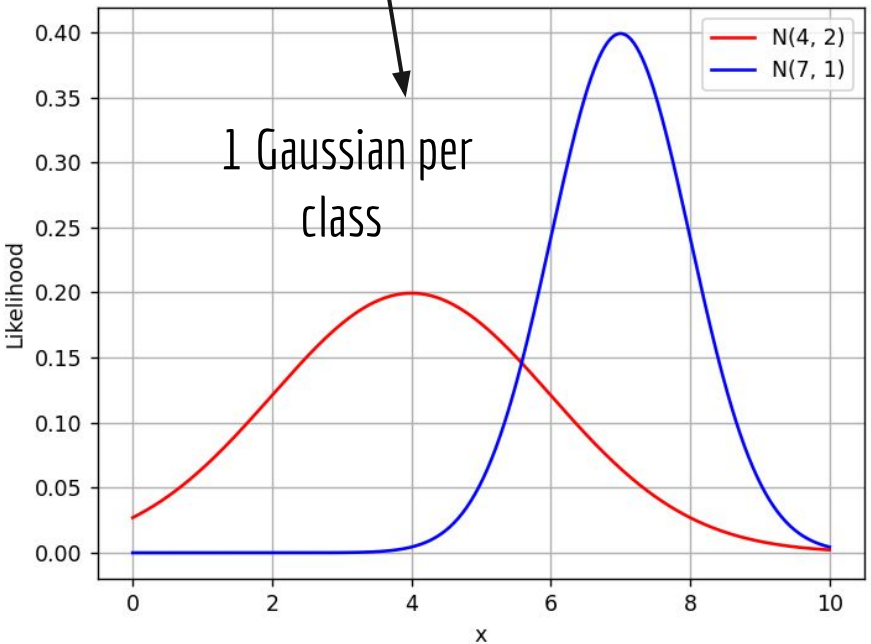
- SD RAN open source near RT-RIC and exemplar xApps (handover, load balancing)
- Aether (5G connected edge platform) and ONOS (for SDN)

And others: https://ai-ran.org, etc.

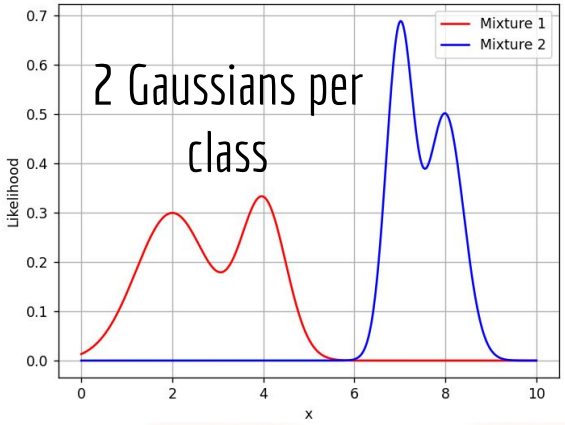
Basic GenAI Definitions

Gaussian mixture model (GMM): our first generative model

Split world into two categories of people: "bad" with average "grade" 4 and standard deviation 2, and "good" with average 7 and std. dev. 1. These distributions are denoted as **N**(4, 2) and **N**(7, 1), respect.

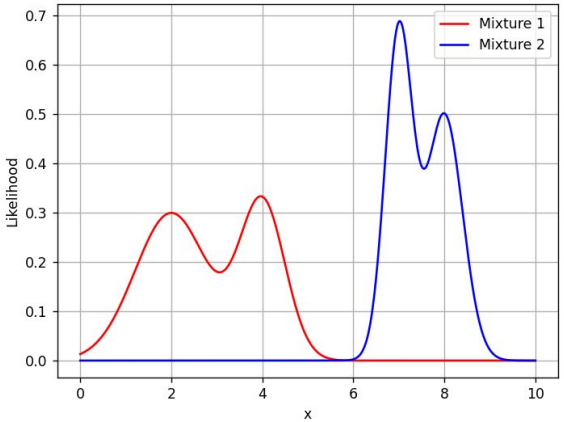


More advanced model:




```
# Python to generate "samples" from N-d Gaussian:
samples = np.random.multivariate_normal(mean,
    cov, num_samples)
```

One can generate several statistics from a distribution and, similarly, from a generative model



Generative model definition: can generate new data points from learned probability distribution

samples = np.random.multivariate_normal(mean, cov, N)

One can also obtain several statistics:

For example, the average of *X* ⇒ $\mathbb{E}[X] = \int_{-\infty}^{\infty} x \cdot p(x)\, dx$

That is applied in [1] to estimate the wireless channel *h* from received signal *y* ⇒ $\mathbb{E}[\boldsymbol{h}|\boldsymbol{y}] = \int \boldsymbol{h} p(\boldsymbol{h}|\boldsymbol{y}) \mathrm{d}\boldsymbol{h}$

[1] FESL, Benedikt et al. Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation, 2024.

Bayes classifier

Posterior = Likelihood × Prior ÷ Evidence

$$P(C_k|\mathbf{x}) = \frac{P(\mathbf{x}|C_k)P(C_k)}{P(\mathbf{x})}$$

$P(C_k|\mathbf{x})$ = Posterior probability of class C_k given the data point $\mathbf{x}$

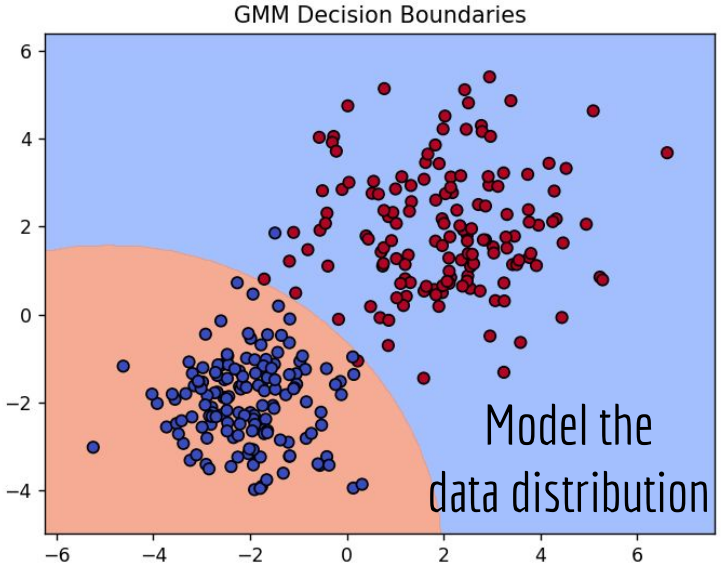
$P(\mathbf{x}|C_k)$ = Likelihood of the data given class C_k

$P(C_k)$ = Prior probability of class C_k

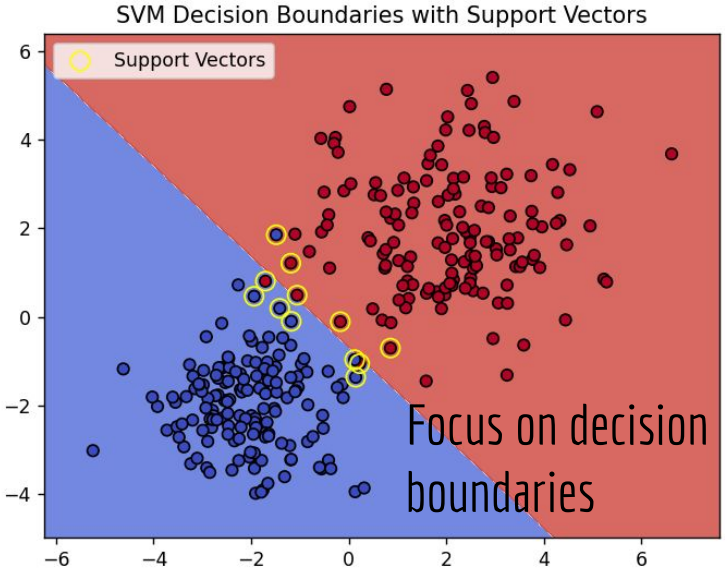
$P(\mathbf{x})$ = Marginal likelihood (normalizing constant) ⇒ Evidence

Discriminative (support Vector Machines, SVMs) versus generative (Gaussian Mixture Models, GMMs)

Generating examples may not be the main job of a generative model. It can be used, for instance, as a **generative classifier**



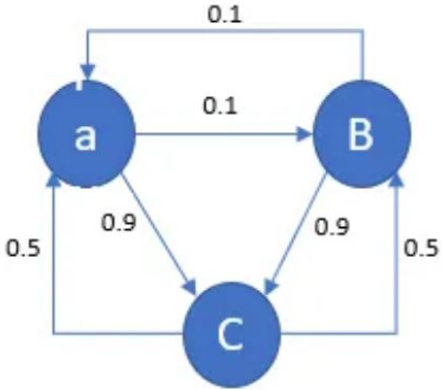
When using the correct distributions, the **Bayes generative classifier is optimum**



Discriminative are better classifiers: why learning distributions if the task is discriminate classes?

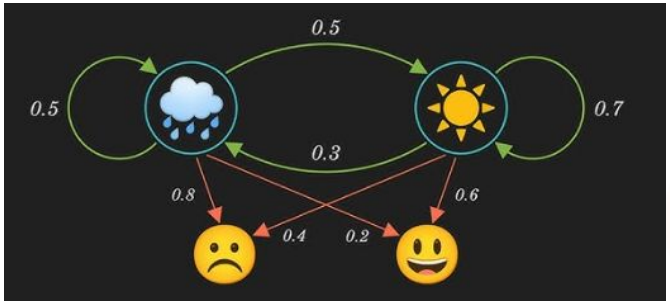
Many classic probabilistic and non-DL generative models

Markov Chains

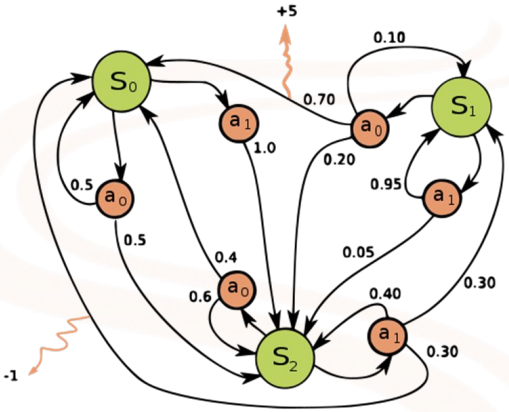


$$P = \begin{bmatrix} 0 & 0.1 & 0.9 \\ 0.1 & 0.0 & 0.9 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}$$

Hidden Markov Models (HMMs)

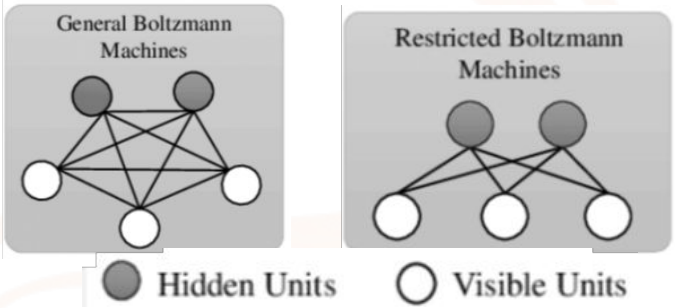


Markov Decision Processes (MDPs)



Neural network-based:

- Boltzmann Machines (BMs)
- Restricted Boltzmann Machines (RBMs)

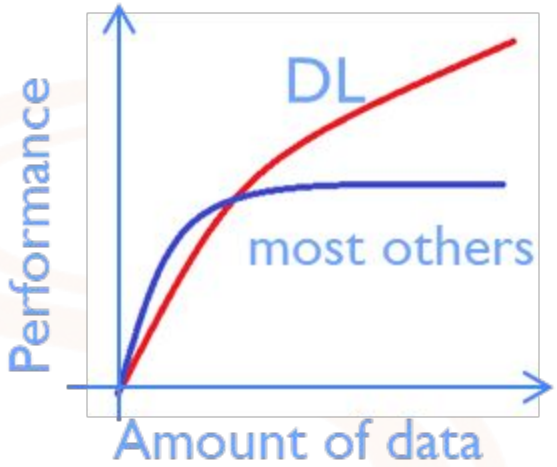


https://medium.com/@soumallya160/a-complete-guide-to-boltzmann-machine-deep-learning-7f7ce29b9e09
HMM: https://www.youtube.com/watch?v=RWkHJnFj5rY
https://towardsdatascience.com/markov-chain-analysis-and-simulation-using-python-4507cee0b06e

Rapid progress on generative models: many DL-based

TABLE 6. GMs along with potential complementary roles.

Comp. Role	Generative Model Examples
Data Augmentation	DCGAN, CycleGAN, VAEs, RBMs, PixelCNN/RNN, FGMs
Data Imputation	VAEs, RBMs, DBNs, PixelCNN/RNN, Normalizing FGMs
Disentanglement	β-VAEs, InfoGANs, FactorVAEs, FGMs
Regularization	VAEs (as Bayesian prior), RBMs (as pre-training)
Dim. Reduction	VAEs, RBMs, DBNs, FGMs
Feature Learning	BiGAN, InfoGAN, VAEs, RBMs, DBNs, PixelCNN/RNN, FGMs
Transfer Learning	StarGAN, CycleGAN, conditional VAEs, PixelCNN/RNN
Semi-SL	SGAN, CatGAN, VAEs, PixelCNN/RNN
Multi-Task Learning	MT-GAN, MT-VAEs, PixelCNN/RNN
Balancing Exp.	VAEs, GANs , PixelCNN/RNN (all with model-based RL)
Imitation Learning	GAIL, VAEs, PixelCNN/RNN (all with behavioral cloning)

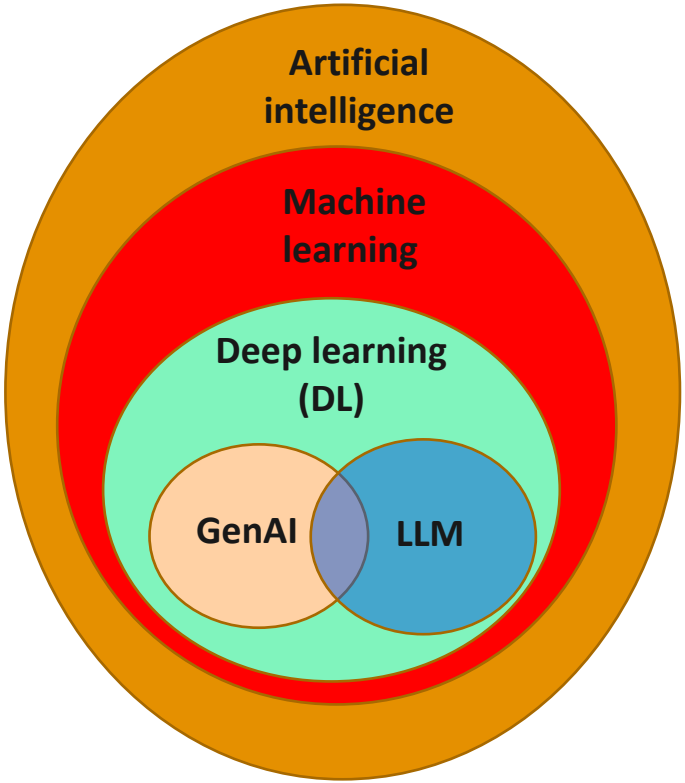


In this tutorial:

CNN, RNN, VAEs, GANs…

Celik, Abdulkadir, and Ahmed M. Eltawil. "At the Dawn of Generative AI Era: A tutorial-cum-survey on new frontiers in 6G wireless intelligence." *IEEE Open Journal of the Communications Society* (2024)

What is GenAI?
It is DL-based generative modeling



Model type	Category	Applications
Transformers	Self-attention- based Models	NLP (text completion, translation, summarization, etc.)
GANs	Implicit Generative Models	Image, audio and video generation
VAEs	Explicit Generative Models	Data generation/augmentation, anomaly detection
Diffusion Models	Stochastic Generative Models	Computer Vision (image and video generation, image processing, semantic segmentation)

Karapantelakis, Athanasios, et al. "A Survey on the Integration of Generative AI for Critical Thinking in Mobile Networks." *arXiv preprint arXiv:2404.06946* (2024).

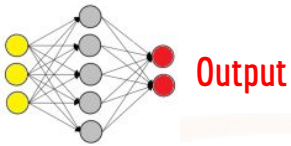
Extra!

Some milestones on artificial neural networks

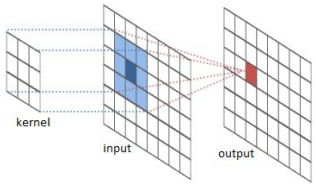
[Rosenblatt] "Perceptron"

[Rumelhart et al] "Multilayer Perceptron" (using backpropagation)

MLP "dense" architecture:



1957 | 1980 | 1986 | 1989 | 1994 | 2006 | 2011 | 2012



[Fukushima] "Convolutional" layers

[LeCun et al] Zip code (MNIST) with LeNet5

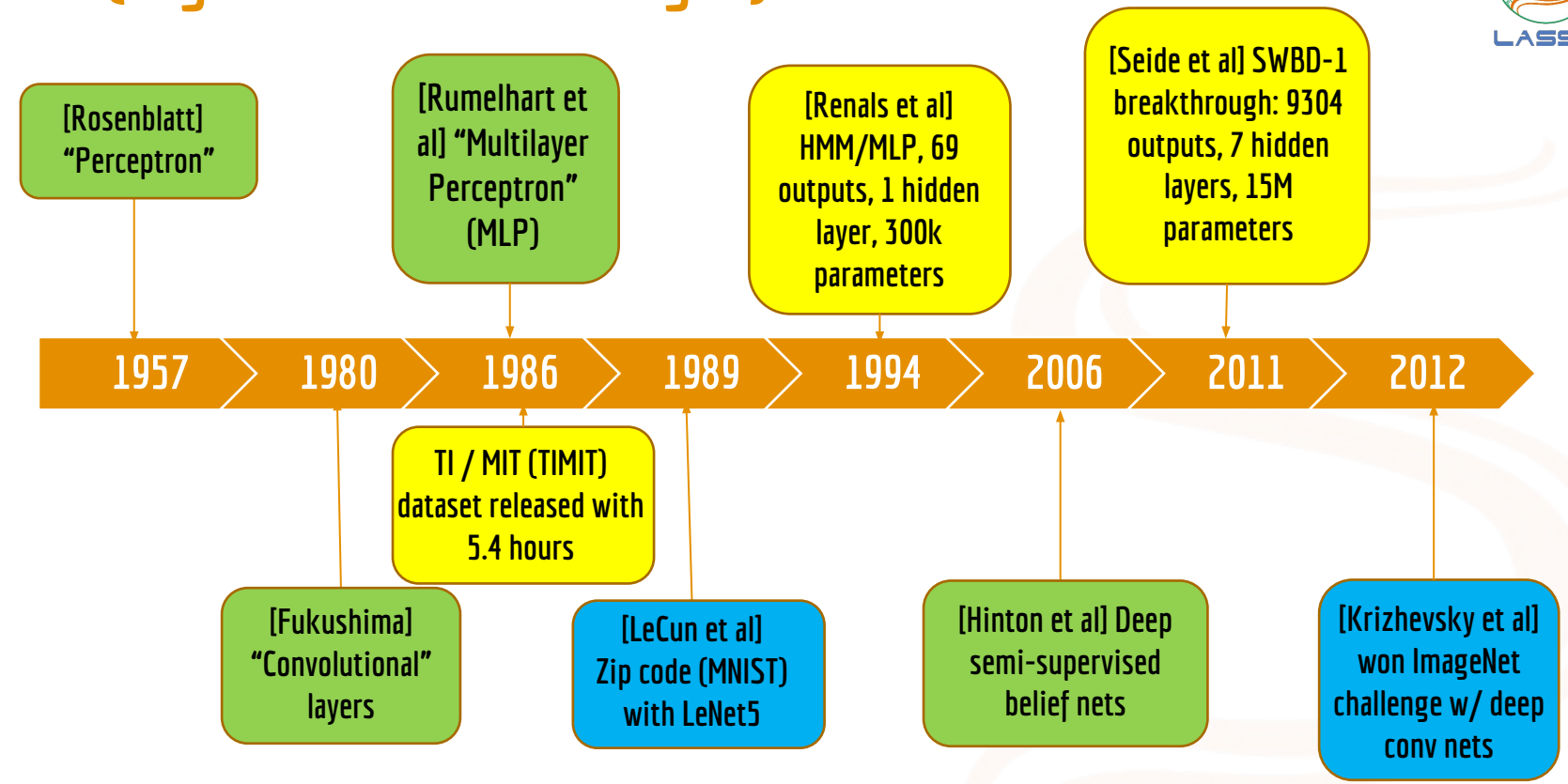
[Hinton et al] Deep semi-supervised belief nets

[Krizhevsky et al] won ImageNet challenge w/ deep conv nets

Algorithms

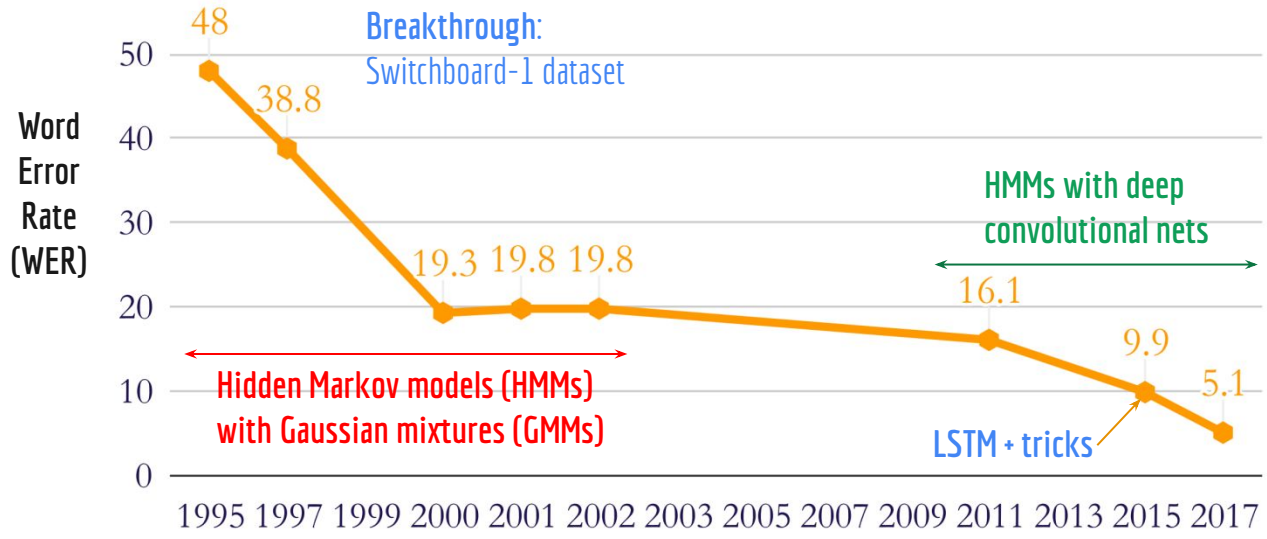
Image processing

DL ("ignorance modeling"?) milestones



Extra!

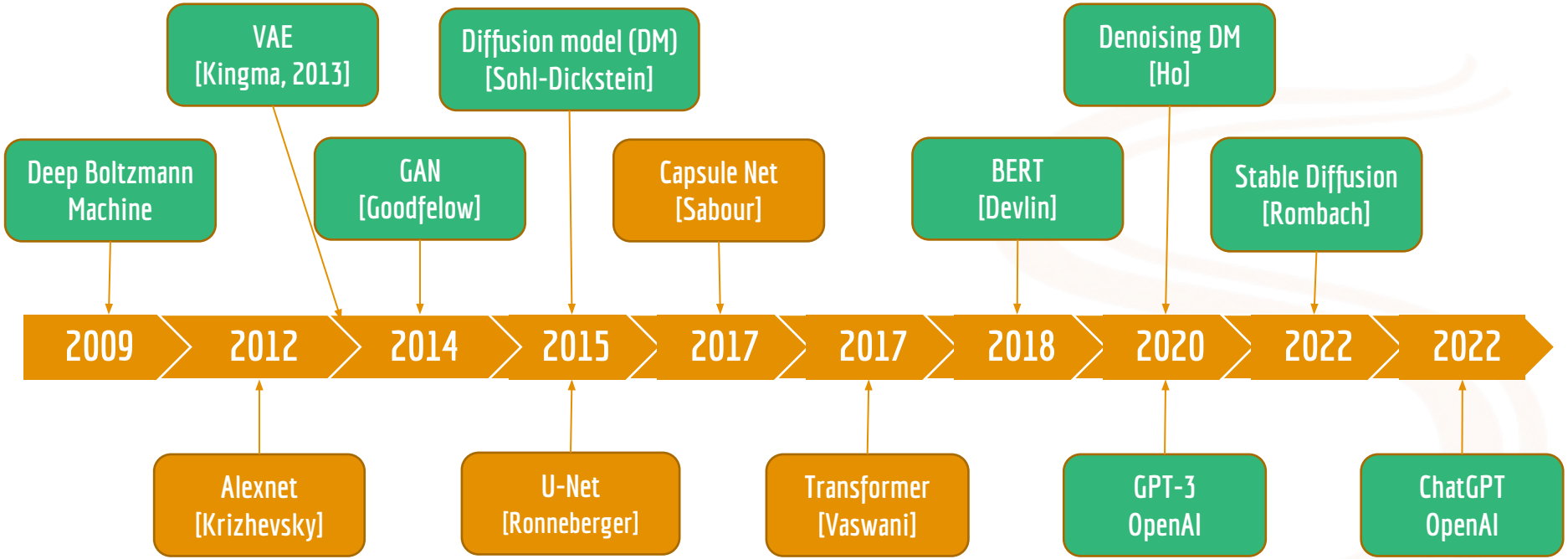
Deep learning improvement in speech recognition



- TIMIT dataset has detailed time-aligned orthographic and phonetic transcriptions
- Took 100 to 1000 hours of work to transcribe each hour of speech
- Project cost over 1 million dollars. Five phoneticians agreed on 75% to 80% of cases

Expert / knowledge-based advocates: against “ignorance” modeling

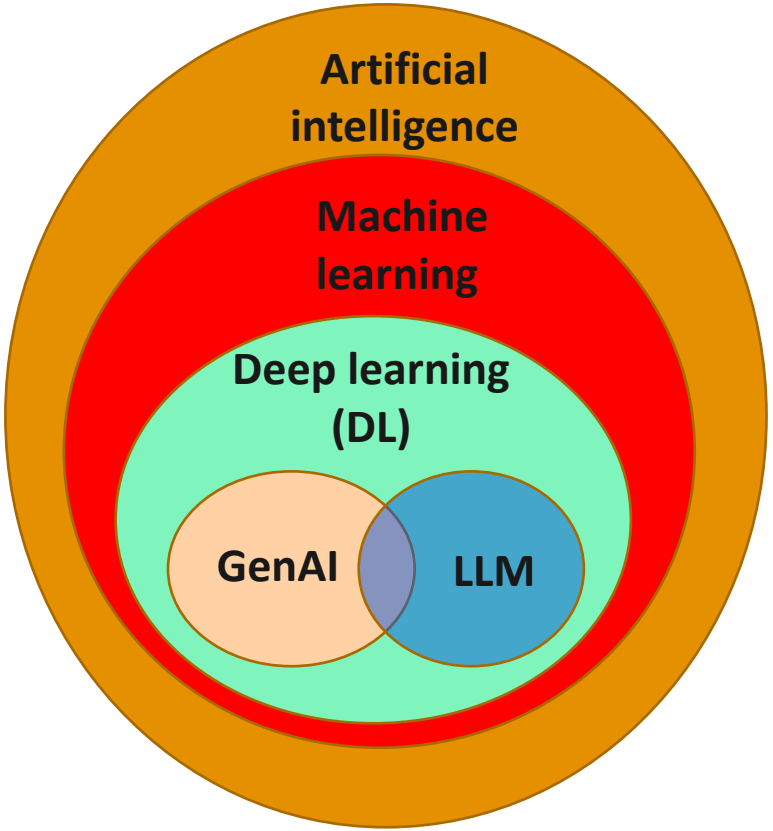
GenAI History



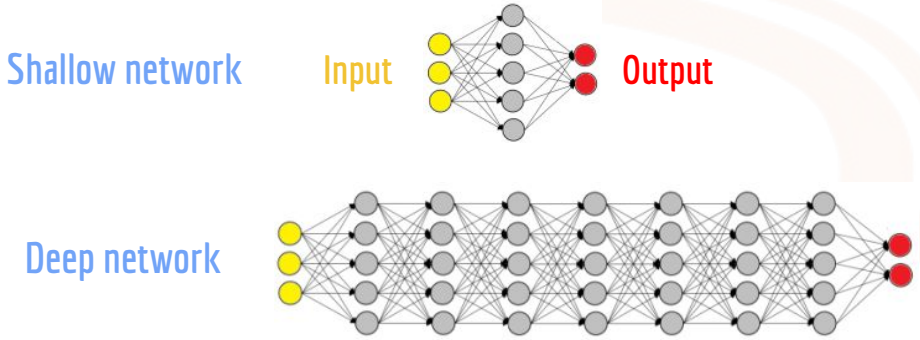
[1] J. Sohl-Dickstein et al, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics", 2015
[2] A. Celik and A. M. Eltawil, "At the Dawn of Generative AI Era: A Tutorial-cum-Survey on New Frontiers in 6G Wireless Intelligence," IEEE, 2024

Basic Concepts of DL

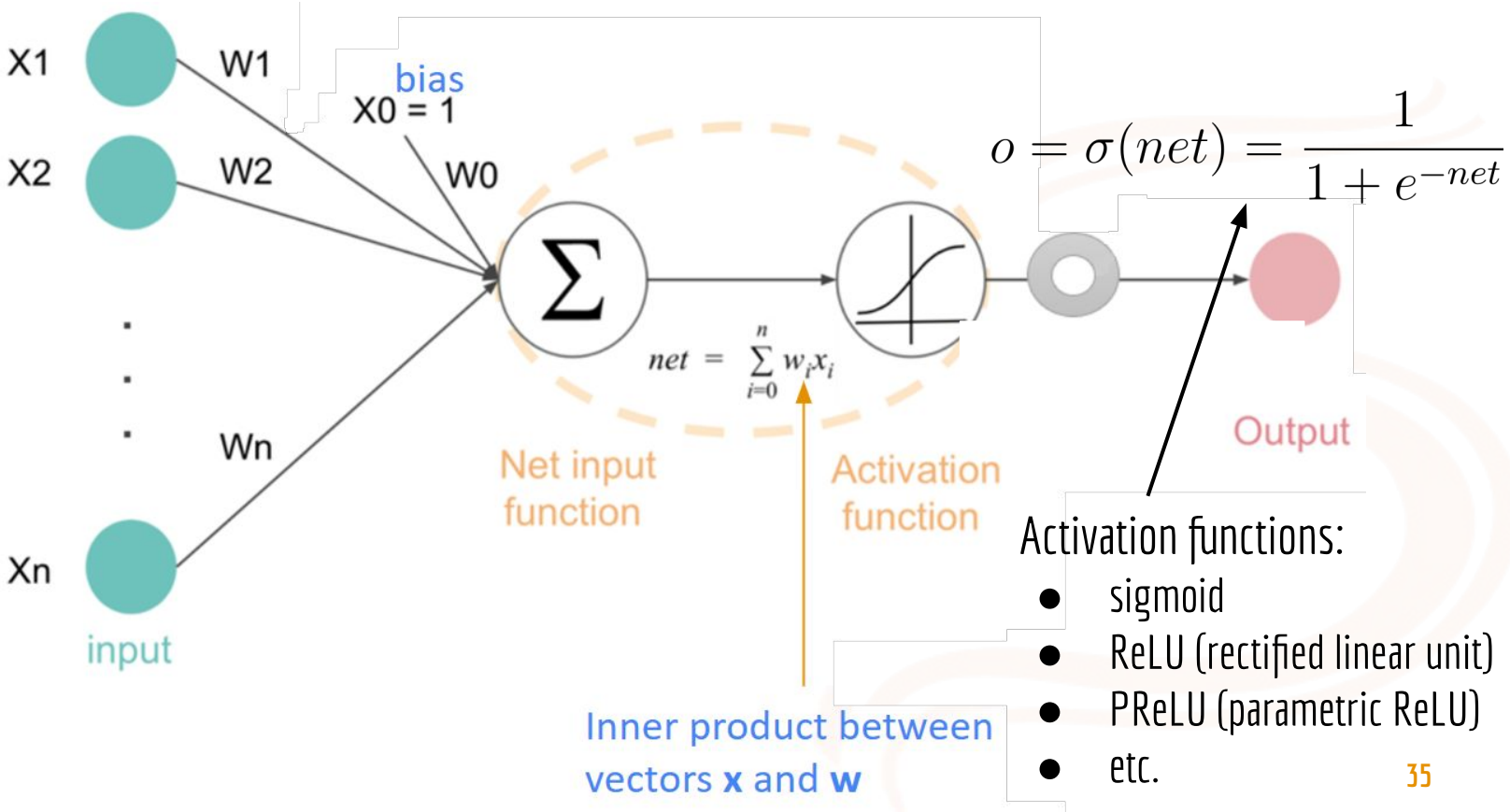
Deep learning



DL is a set of techniques to train and deploy neural networks with large number parameters



Perceptron: a single neuron



Activation functions:

- sigmoid
- ReLU (rectified linear unit)
- PReLU (parametric ReLU)
- etc.

Activation function softmax: operates on vectors, not scalars

Softmax: starting with unnormalized (natural) log probabilities or *logits* z

$$z_1 = \ln(q_1), z_2 = \ln(q_2), z_3 = \ln(q_3)$$

q values may be outside the range [0, 1]

Convert them back to *q* and normalize (*K*=3 in this example):

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Obtain a probability distribution with *p* in range [0, 1]

$$p_1 + p_2 + p_3 = 1$$

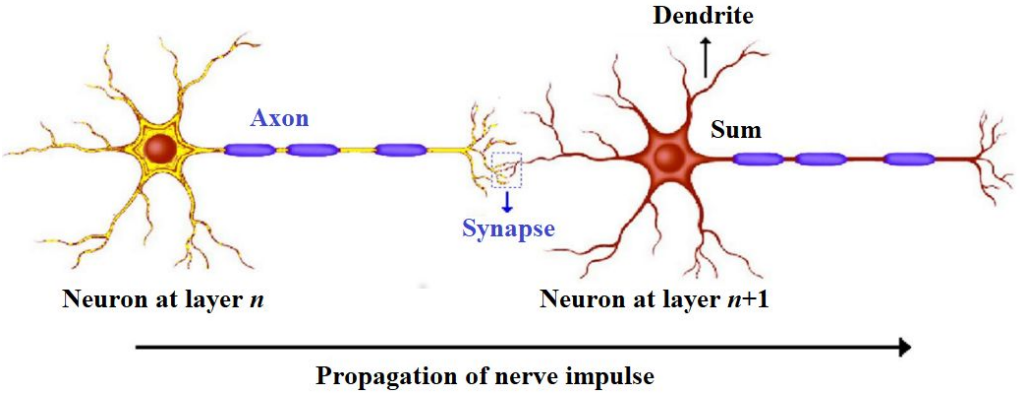
Example of logits:
z=[-2.4, 1.2, -0.6]
Softmax calculation:
q=exp(z)
p=q/sum(q)

Result:
q = [0.09, 3.32, 0.55]
p = [0.02 0.84 0.14]
sum(p) = 1

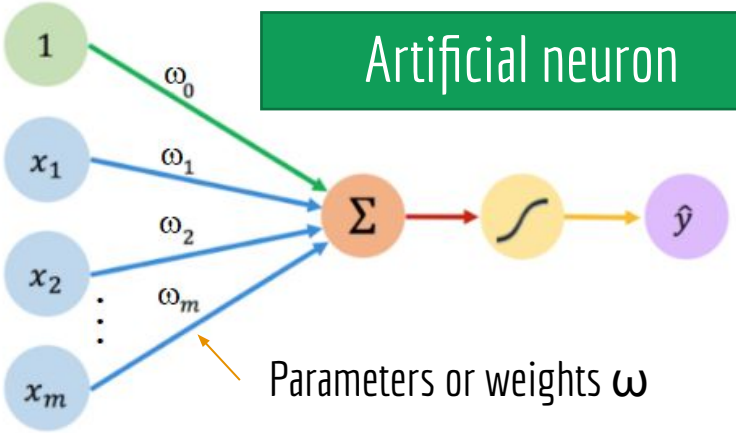
Extra!

Artificial neural network (ANN or NN): collection of neurons

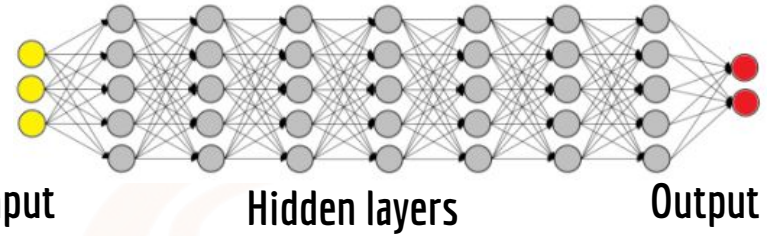
Biological neuron

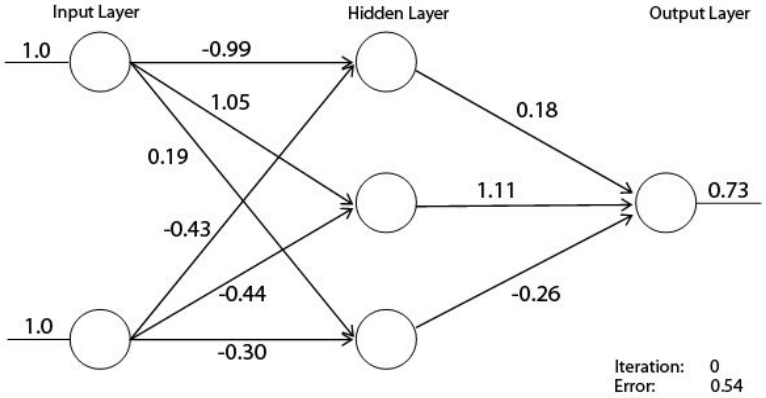


Artificial neuron



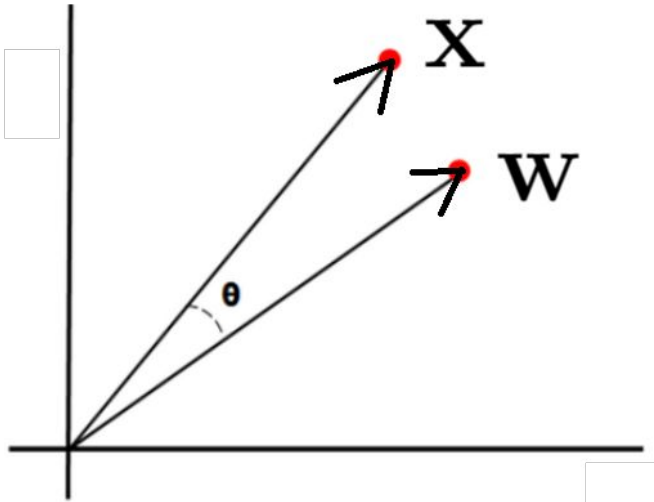
Parameters or weights ω





Many layers and architectures in DL: dense (fully-connected), convolutional, recurrent, etc.

Inner (or dot) product as a similarity measure



$$< \mathbf{x}, \mathbf{w} >= \|\mathbf{x}\| \ \|\mathbf{w}\| \cos(\theta) = \sum_{i=1}^{N} x_i w_i$$

Given $\mathbf{x} = [3, 2]$ and 3 examples:

$$\mathbf{w} = [4, 3], \quad < \mathbf{x}, \mathbf{w} >= 3 \times 4 + 2 \times 3 = 18$$

$$\mathbf{w} = [4, -3], < \mathbf{x}, \mathbf{w} >= 3 \times 4 - 2 \times 3 = 6$$

$$\mathbf{w} = [4, -6], < \mathbf{x}, \mathbf{w} >= 3 \times 4 - 2 \times 6 = 0$$

The **cosine similarity** $\cos(\Theta)$ indicates how closely vectors are aligned in direction

AI models (MLP, CNN, transformers, etc.) optimize weights **w** using **inner products** to obtain the desired result

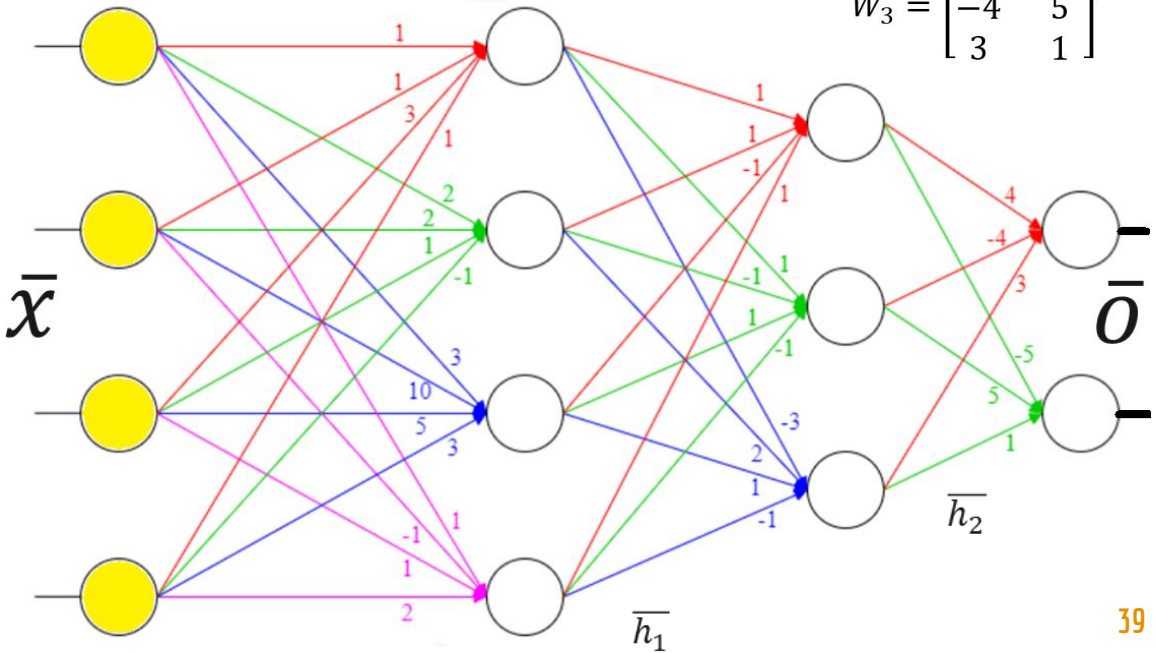
Matrix notation for fully connected (dense) layers

Example of **fully connected** network with *k*=2 hidden layers

$$W_1 = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 1 & 2 & 10 & -1 \\ 3 & 1 & 5 & 1 \\ 1 & -1 & 3 & 2 \end{bmatrix}$$

$$W_2^T = \begin{bmatrix} 1 & 1 & -3 \\ 1 & -1 & 2 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \end{bmatrix}$$

$$W_3^T = \begin{bmatrix} 4 & -5 \\ -4 & 5 \\ 3 & 1 \end{bmatrix}$$



Extra!

Matrix notation for fully connected (dense) layers

$$\overline{h}_1 = \Phi(W_1^T \overline{x}) \quad \text{Input processing}$$

$$\overline{h}_{p+1} = \Phi(W_{p+1}^T \overline{h}_p) \quad \forall p \in \{1 \dots k-1\}$$

$$\overline{o} = \Phi(W_{k+1}^T \overline{h}_k) \quad \text{Output processing}$$

$$W_1 = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 1 & 2 & 10 & -1 \\ 3 & 1 & 5 & 1 \\ 1 & -1 & 3 & 2 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 1 & 1 & -3 \\ 1 & -1 & 2 \\ -1 & 1 & 1 \\ 1 & -1 & -1 \end{bmatrix}$$

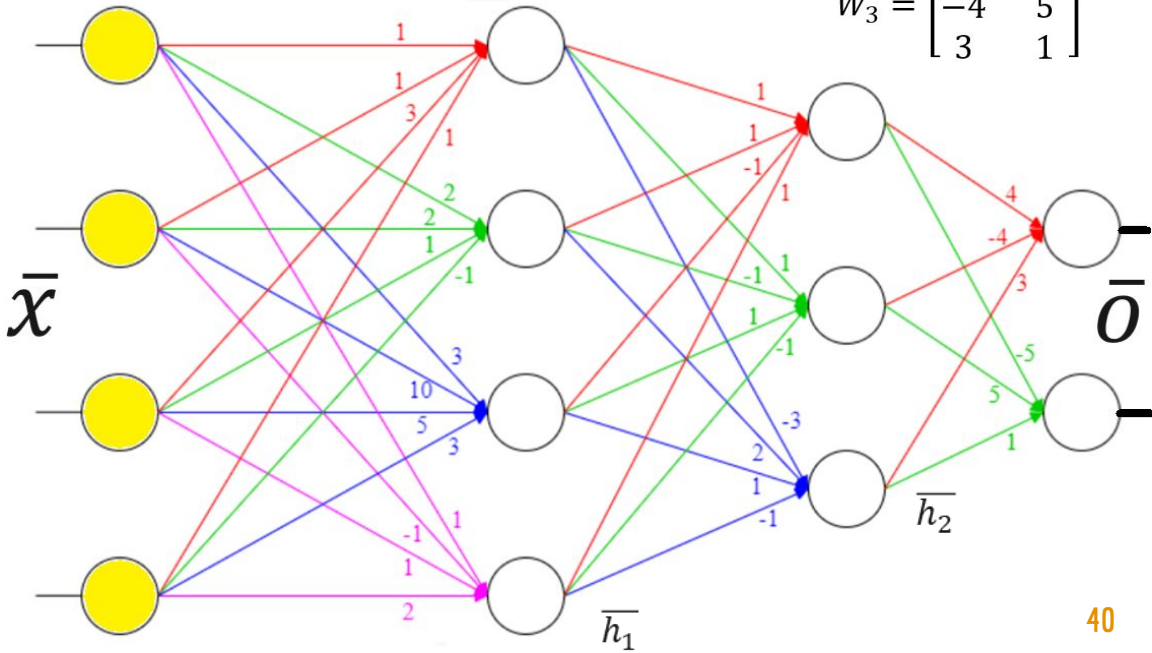
$$W_3 = \begin{bmatrix} 4 & -5 \\ -4 & 5 \\ 3 & 1 \end{bmatrix}$$

Example of dense (fully connected) network with k=2 hidden layers

Number N_p of weights (trainable parameters) for the p-th layer with M_p neurons:

- Without bias: $M_p \times M_{p-1}$
- With bias: $M_p (M_{p-1} + 1)$

34 weights in example (no bias): Dense 1: $M_1 = 4$ and $N_1 = 16$
Dense 2: $M_2 = 3$ and $N_2 = 12$
Dense 3: $M_{out} = 2$ and $N_{out} = 6$



Extra!

Describing the previous example in Keras

```
1  from tensorflow.keras import layers, models, utils
2  model = models.Sequential()
3  model.add(layers.Input(shape=(4,)))
4  model.add(layers.Dense(4, activation='relu',use_bias=False))
5  model.add(layers.Dense(3, activation='relu',use_bias=False))
6  model.add(layers.Dense(2, activation='relu',use_bias=False))
7  model.summary()
8  utils.plot_model(model, "model_graph.png", show_shapes=True)
```

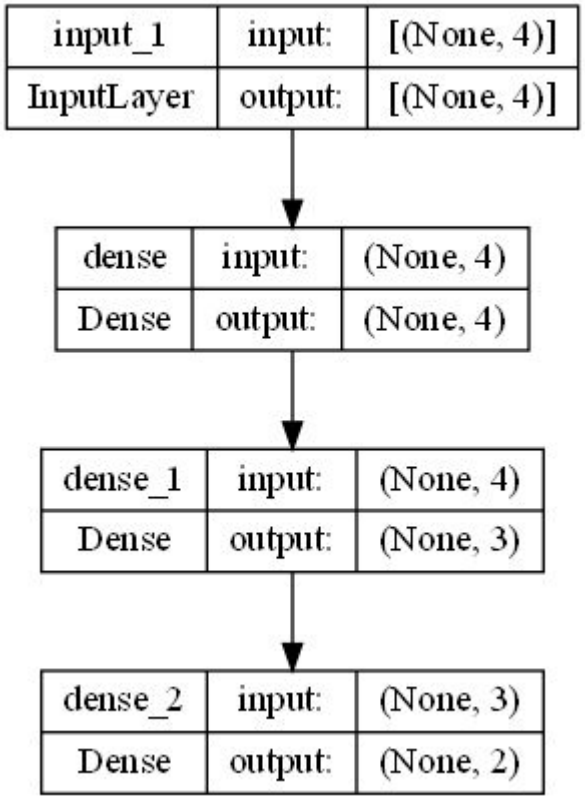
model.summary()

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 4)	16
dense_1 (Dense)	(None, 3)	12
dense_2 (Dense)	(None, 2)	6

Total params: 34
Trainable params: 34
Non-trainable params: 0

keras.util.plot_model:



Extra!

Dense layers: number of weights depend on input dimension

LASSE

Consider processing a 1920 x 1080 multispectral satellite image, with depth = 15 bands, using a single dense layer with 10 neurons (with bias) □ Number of weights is (1920x1080x15+1)x10 = 311 millions

```
1  from tensorflow.keras import layers, models, utils
2  model = models.Sequential()
3  model.add(layers.Input(shape=(1920,1080,15)))
4  model.add(layers.Flatten())
5  model.add(layers.Dense(10, activation='relu'))
6  model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 311040000)         0
dense (Dense)                (None, 10)                311040010
=================================================================
Total params: 311,040,010
Trainable params: 311,040,010
Non-trainable params: 0
```

Convolutional layers “share” weights and enable “localization”

Use the convolution (correlation, or "moving" inner product) to achieve location invariance

Neuron that "moves" and mimics visual cortex receptive fields

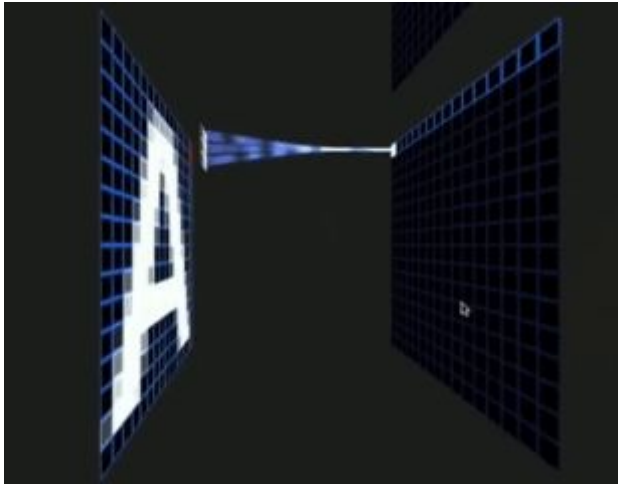
Making an analogy with a dense (fully connected) layer, a "neuron" in a **convolutional layer** is composed of a D-dimensional filter (with D distinct kernels, where D is the "depth") and a "**position**"

Source code: https://stackoverflow.com/questions/52067833/how-to-plot-an-animated-matrix-in-matplotlib

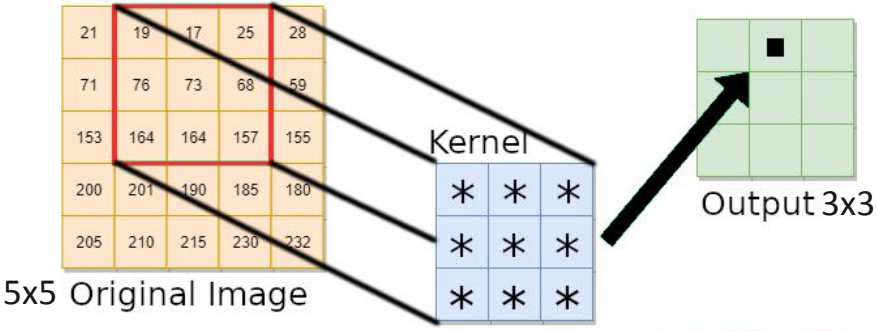
Extra!

Convolutional layers

Consider processing a 1920 x 1080 multispectral satellite image, with depth = 15 bands, using a dense layer. With only 10 neurons, the number of weights would be 1920x1080x15x10 = 311 milhões



Now, use convolution to train filters composed by one kernel per depth dimension



Number of trainable parameters for this 3D filter assuming one weight for the bias: N = 3 x 3 x 15 + 1 = 136

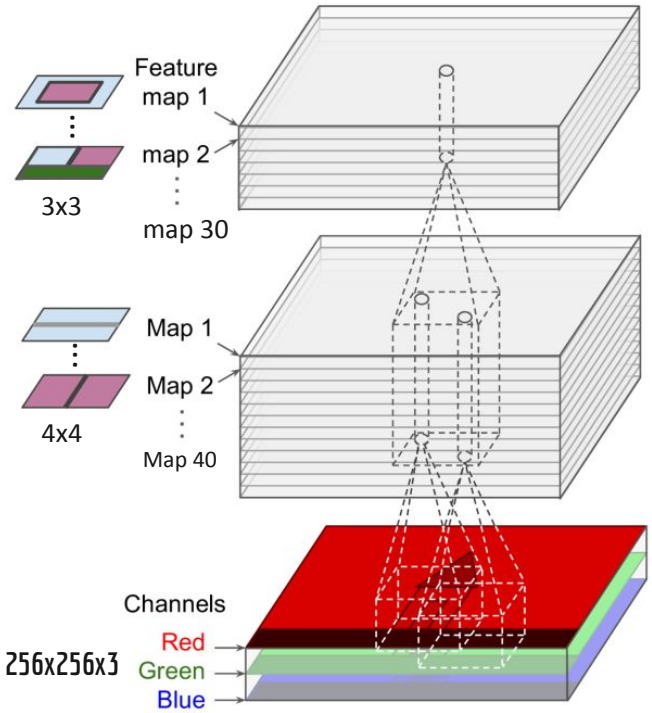
The combined effect of all kernels and bias is added to create a single dimension (feature map) of the filter

Dimension of each map can decrease if "*stride*" is larger than 1 or if there is no "*zero-padding*"

https://ulhpc-tutorials.readthedocs.io/en/latest/cuda/exercises/convolution/

Extra!

Convolutional layers (2)




```
1  #imports from Keras in TF2.0
2  from tensorflow.keras import layers, models, utils
3  #create convolutional network model
4  model = models.Sequential()
5  model.add(layers.Conv2D(40, (4, 4), activation='relu',
6                          input_shape=(256, 256, 3)))
7  model.add(layers.Conv2D(30, (3, 3), activation='relu'))
8  model.add(layers.Flatten())
9  #show information about the model and save PNG file
10 model.summary()
11 utils.plot_model(model,"model_graph.png",show_shapes=True)
```

conv2d_input	input:	[(None, 256, 256, 3)]
InputLayer	output:	[(None, 256, 256, 3)]

conv2d	input:	(None, 256, 256, 3)
Conv2D	output:	(None, 253, 253, 40)

conv2d_1	input:	(None, 253, 253, 40)
Conv2D	output:	(None, 251, 251, 30)

flatten	input:	(None, 251, 251, 30)
Flatten	output:	(None, 1890030)

Mini-batch dimension is not specified (None)

Number of filters impose the depth at each layer

251x251x30 = 1890030

Extra!

Convolutional layers (3)

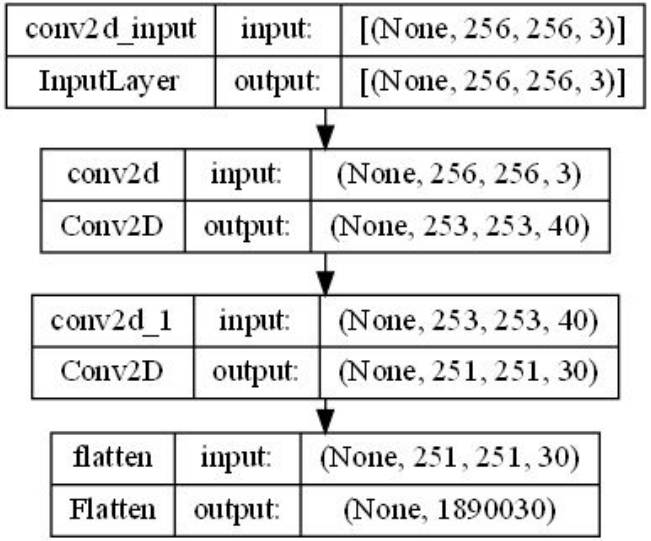
Number of parameters ("weights"):

40(4x4x3+1)=1960

30(3x3x40+1)=10830

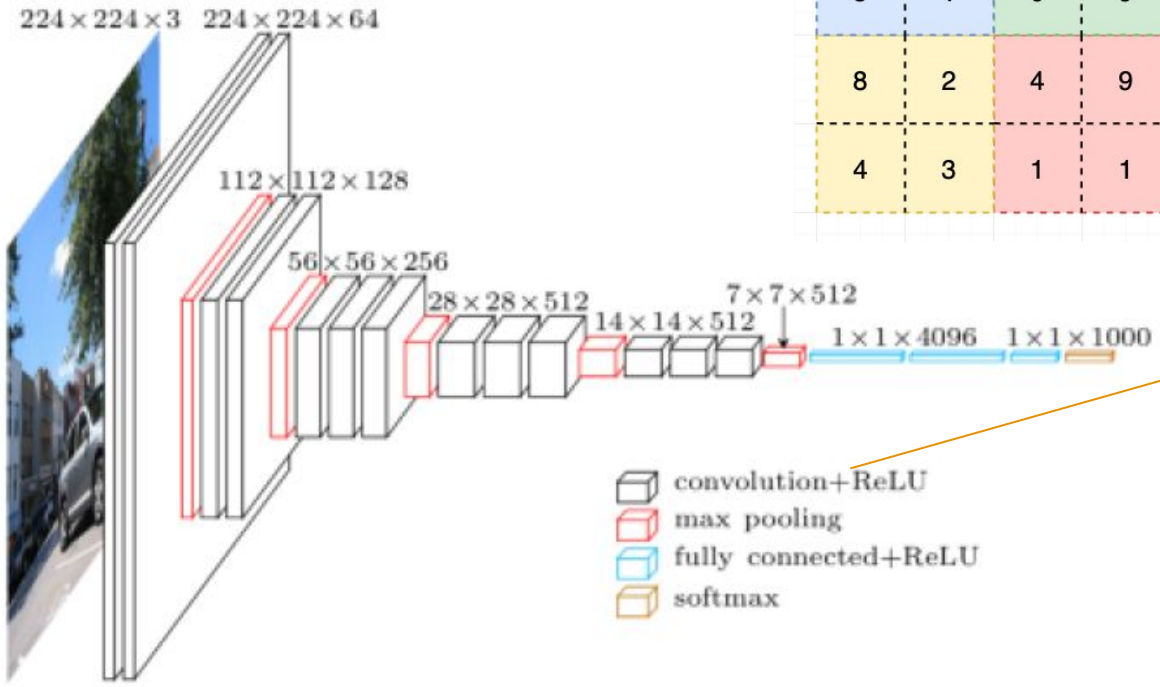
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 253, 253, 40)      1960
conv2d_1 (Conv2D)            (None, 251, 251, 30)      10830
flatten (Flatten)            (None, 1890030)           0
=================================================================
Total params: 12,790
Trainable params: 12,790
Non-trainable params: 0
_________________________________________________________________
```

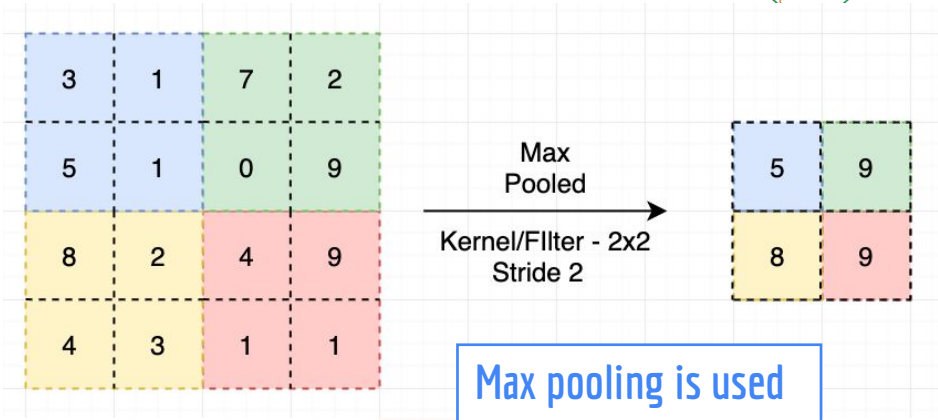
```
1  #imports from Keras in TF2.0
2  from tensorflow.keras import layers, models, utils
3  #create convolutional network model
4  model = models.Sequential()
5  model.add(layers.Conv2D(40, (4, 4), activation='relu',
6                          input_shape=(256, 256, 3)))
7  model.add(layers.Conv2D(30, (3, 3), activation='relu'))
8  model.add(layers.Flatten())
9  #show information about the model and save PNG file
10 model.summary()
11 utils.plot_model(model,"model_graph.png",show_shapes=True)
```



Extra!

Traditional convolutional neural network (CNN) for images





Max pooling is used to reduce dimension

Batch normalization (BN) is another popular layer

https://flyyufelix.github.io/2016/10/08/fine-tuning-in-keras-part2.html

Extra!

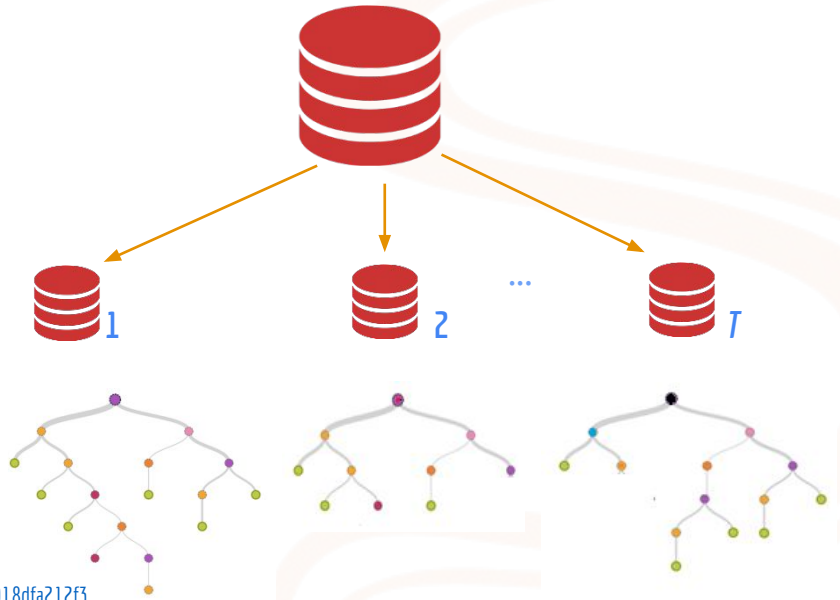
There is life outside the deep neural net world!

LASSE

Random forest: model composed of several **decision trees**

dmlc XGBoost

Random forests often provide better results than neural networks when datasets are obtained from tables (tabular data)



https://dmlc.github.io
https://xgboost.ai
https://towardsdatascience.com/xgboost-intro-step-by-step-implementation-and-performance-comparison-6018dfa212f3
https://towardsdatascience.com/xgboost-how-deep-learning-can-replace-gradient-boosting-and-decision-trees-291dc9365656
https://kaabar-sofien.medium.com/xgboost-2-0-is-here-to-improve-your-time-series-forecasts-98fd35aaca40

Extra!

Scikit-learn algorithms for classification

Relevant hyperparameters

Decision tree:	DecisionTreeClassifier(min_samples_leaf=5)
Decision stump:	DecisionTreeClassifier(max_depth=1)
Naïve Bayes	GaussianNB(priors=None, var_smoothing=1e-09)
K-nearest neighbors (KNN)	KNeighborsClassifier(n_neighbors=3, metric='euclidean')
Support vector machine (SVM)	SVC(C=1.0, kernel='rbf', decision_function_shape='ovo')
Linear (SVM)	LinearSVC(C=1.0, decision_function_shape='ovr')
Artificial neural network (ANN)	MLPClassifier(max_iter=500, hidden_layer_sizes=(100,))
Adaboost	AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1), n_estimators=50)
Random forest	RandomForestClassifier(n_estimators=30, max_depth=100)

Extra!

Classification with scikit-learn

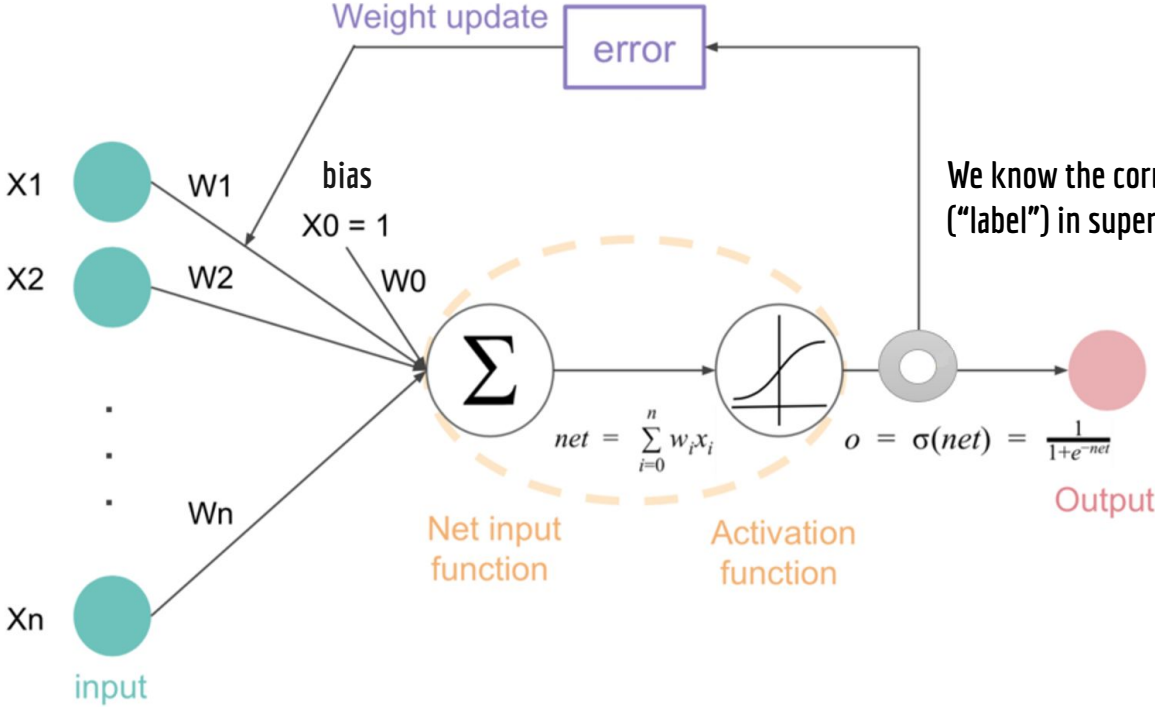
```
classifier = DecisionTreeClassifier(max_depth=20) #single tree
classifier = RandomForestClassifier(n_estimators=30,max_depth=10) #30 trees
classifier = SVC(gamma=1, C=1) #SVM with RBF kernel

classifier.fit(X_train, y_train) #training stage
y_predicted = classifier.predict(X_test) #test stage
print('Accuracy = ', accuracy_score(y_test, y_predicted))
```

Be aware that modern ML has sophisticated workflows that require time to set up and get familiar with

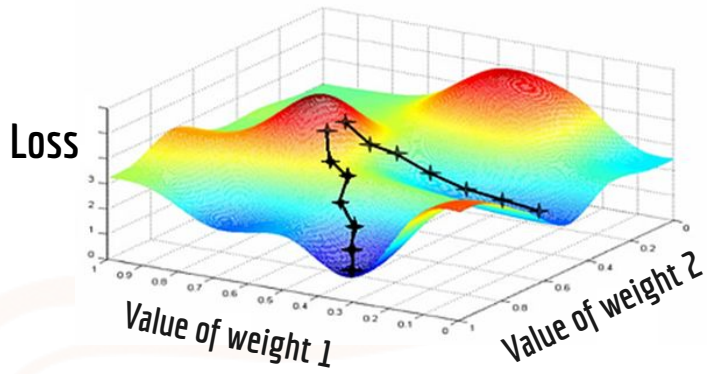
[1] https://scikit-learn.org/

Backpropagation via Stochastic Gradient Descent (SGD)



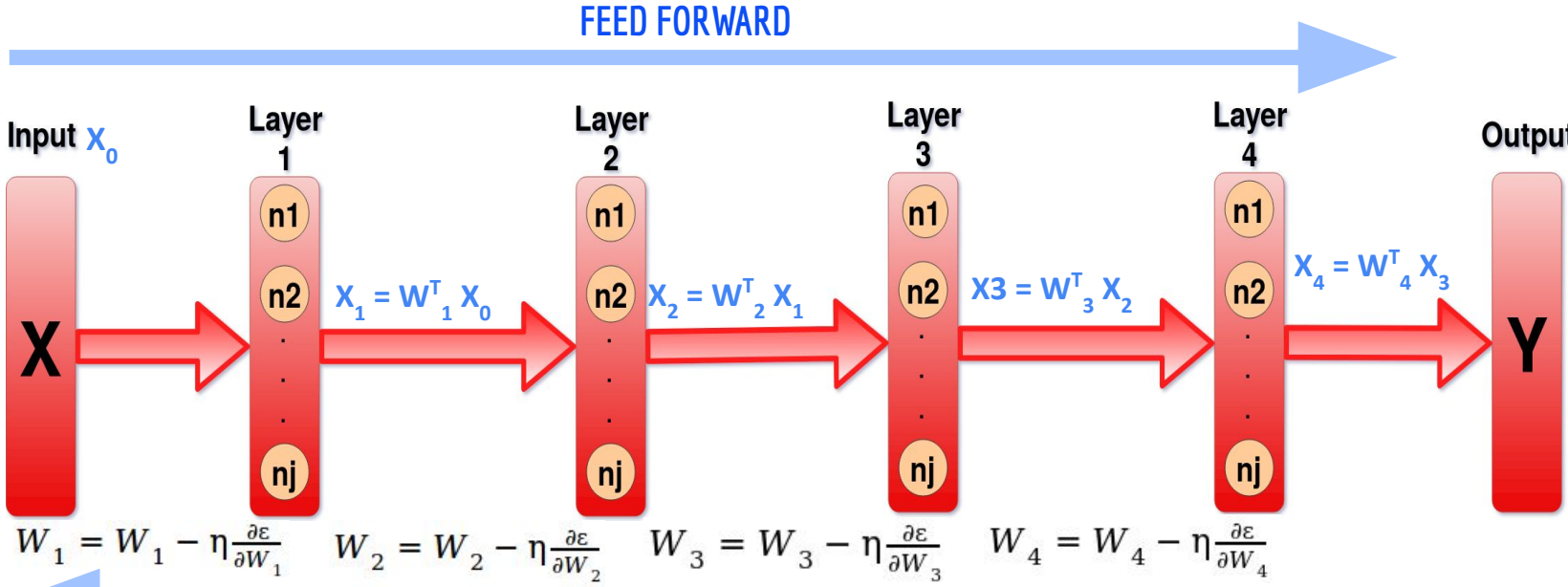
Assume NN with single neuron with 2 weights

We know the correct output ("label") in supervised learning



Extra!

*Backprop via ~~stochastic~~ gradient descent



Given the label, compute error **ε**

Update each weight using gradient descent with learning rate **η**

Increasing # multiplications in the chain rule may lead to over/underflow. Note for 1st layer:

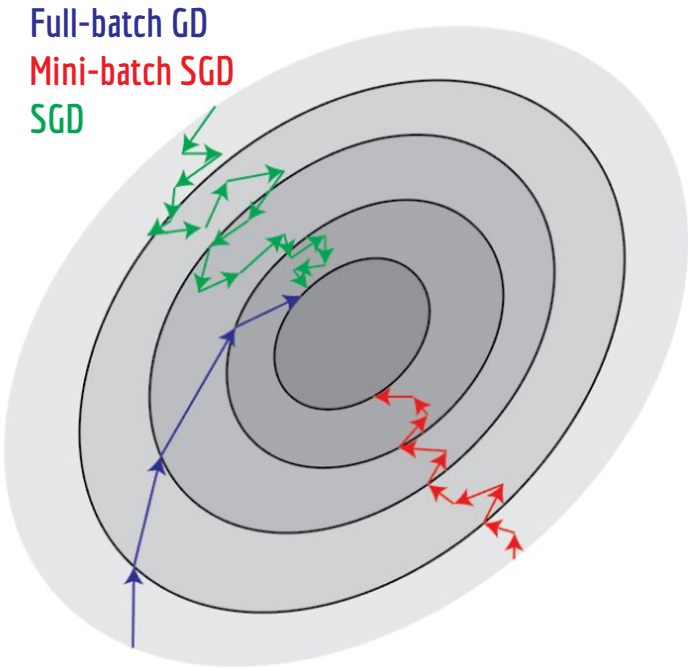
$$W_1 = W_1 - \eta \frac{\partial \varepsilon}{\partial W_4} \frac{\partial W_4}{\partial W_3} \frac{\partial W_3}{\partial W_2} \frac{\partial W_2}{\partial W_1}$$

https://web.stanford.edu/class/cs221/lectures/learningXX.pdf
https://www.3blue1brown.com/lessons/backpropagation-calculus

Extra!

Mini-batch SGD: gradient estimation with *B* examples

Gradient descent (GD) nomenclature assuming N training examples	Number *B* of examples in each gradient step	Number of weight updates per epoch
Full-batch GD or Batch GD or GD	$B = N$	1
Mini-batch SGD or Mini-batch GD	$N > B > 1$	N / B
Stochastic GD (**SGD**)	B = 1	N



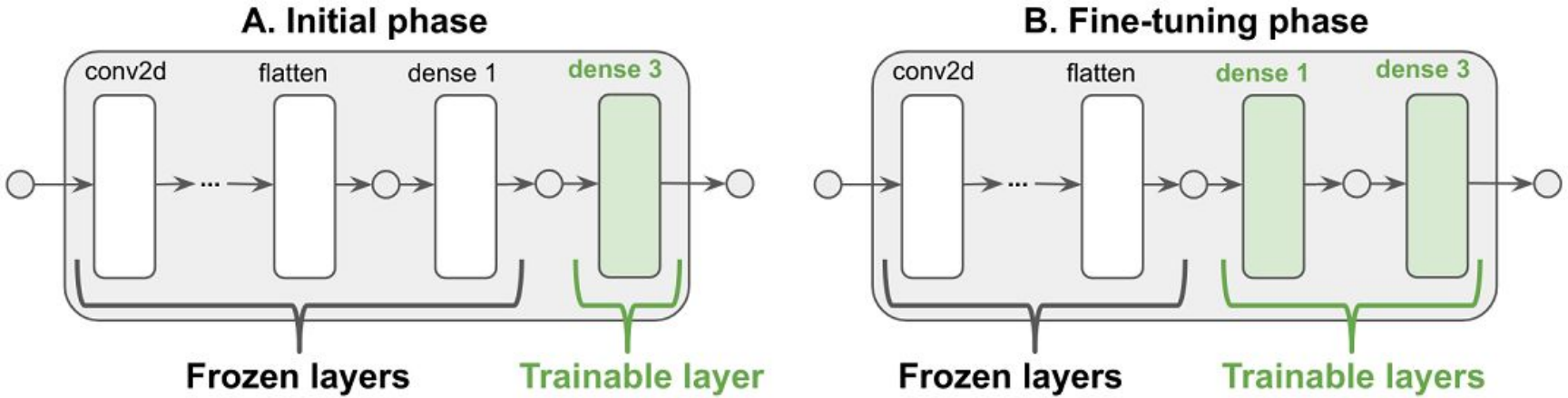
Mini-batch size *B* affects convergence time, training time per epoch and quality of the final model

Obs: the mini-batch needs to fit the GPU memory

[1] SGD: The Role of Implicit Regularization, Batch-size and Multiple Epochs, NeurIPS, 2021, S. Kale et al
[2] On large-batch training for deep learning: generalization gap and sharp minima, 2017, N. Keskar et al
[3] https://www.baeldung.com/cs/mini-batch-vs-single-batch-training-data
[4] https://www.baeldung.com/cs/learning-rate-batch-size

Extra!

Transfer learning and fine-tuning foundational models



Transfer learning:

1. Freeze layers from previously trained model
2. Add new, trainable layers on top of the frozen layers
3. Train the new layers on your dataset
4. Fine-tuning: unfreeze some layers and retrain

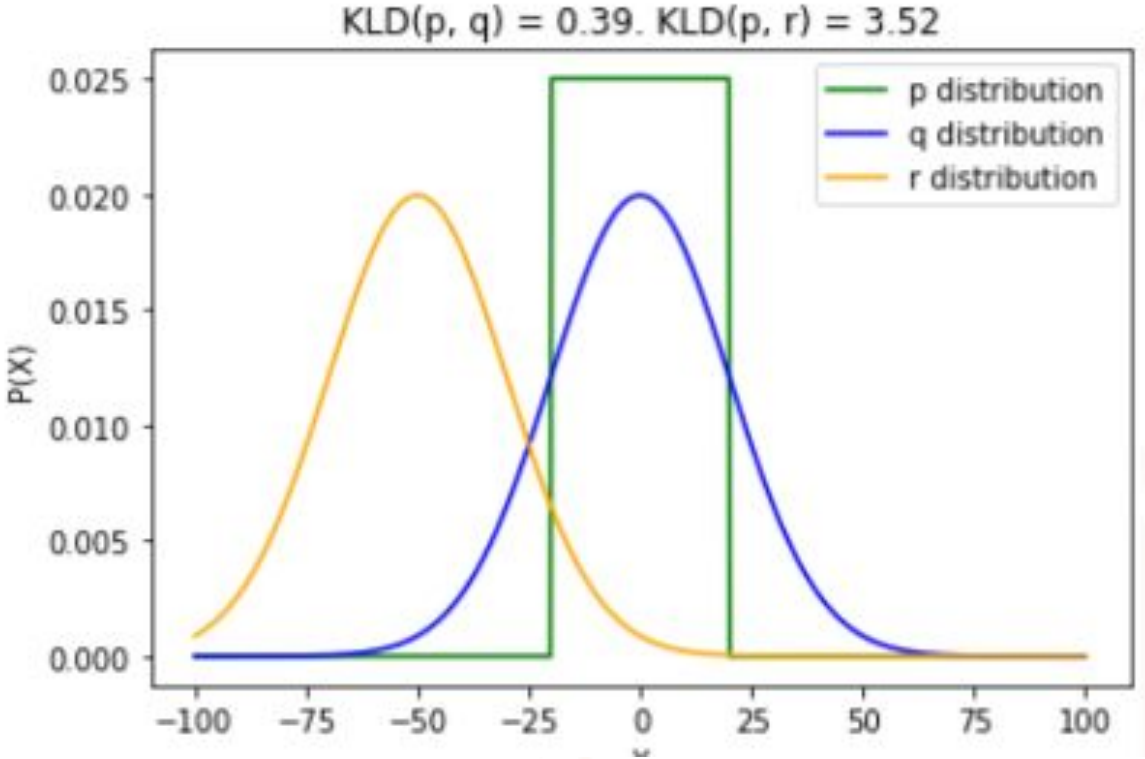
https://pyimagesearch.com/2019/06/03/fine-tuning-with-keras-and-deep-learning
https://www.tensorflow.org/guide/keras/transfer_learning
https://www.sciencedirect.com/science/article/abs/pii/S0378778819324843

Extra!

Cross entropy as loss function

Cross-entropy $L(p,q) = D_{KL}(p//q) + H(p)$ indicates how much two probability distributions differ and is often more effective for training models than the mean squared-error (MSE)

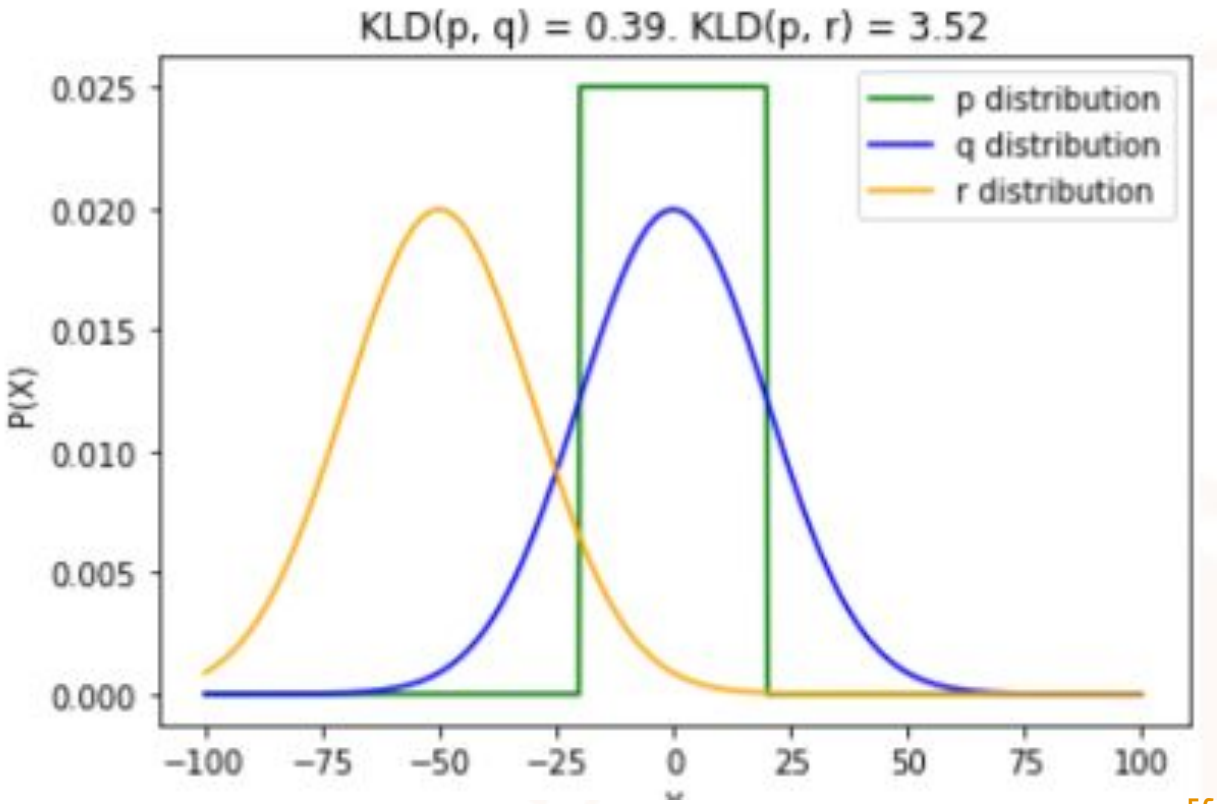
Because p is the fixed target distribution, H(p) is fixed. Hence, minimizing the cross-entropy L(p,q) is equivalent to minimizing the Kullback–Leibler divergence (relative entropy) $D_{KL}(p//q)$



https://livebook.manning.com/book/math-and-architectures-of-deep-learning/chapter-6/v-8/153

Using the Kullback-Leibler divergence to compare distributions

Kullback-Leibler divergence (KL divergence or KLD) indicates how much two probability distributions differ and is often more effective for training models than the mean squared-error (MSE)



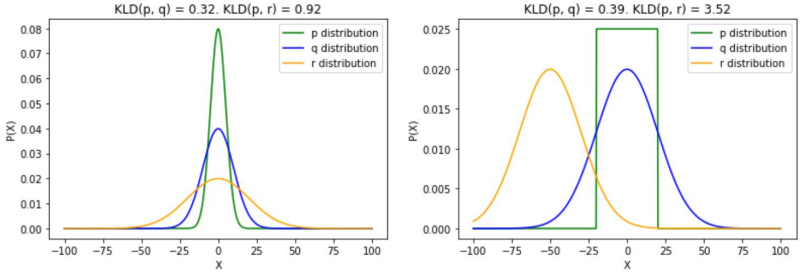
[1] https://livebook.manning.com/book/math-and-architectures-of-deep-learning/chapter-6/v-8/153

Extra!

Relating encoding, activation and loss functions

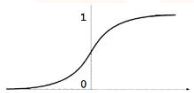
Cross-entropy $L(p,q) = D_{KL}(p//q) + H(p)$ can indicate how much two probability distributions differ and is more effective for training a classifier than the mean squared-error (MSE)

From [1]:



Because H(p) is fixed, minimizing the cross-entropy L(p,q) is equivalent to minimizing the Kullback–Leibler divergence (relative entropy) $D_{KL}(p//q)$

Output encoding	Target distribution(s) p	Output distribution(s) q	Loss	Output activation
Multilabel (bits)	[1, 0, 0, 1]	[0.9, 0.4, 0.7, 0.8]	Binary cross-entropy (BCE)	Sigmoid
One-hot	[0, 1, 0, 0]	[0.2, 0.4, 0.1, 0.3]	Categorical cross-entropy (CE)	Softmax



$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

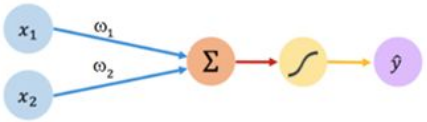
L(p,q) is applied for each of the *K* elements of the output in multilabel classification

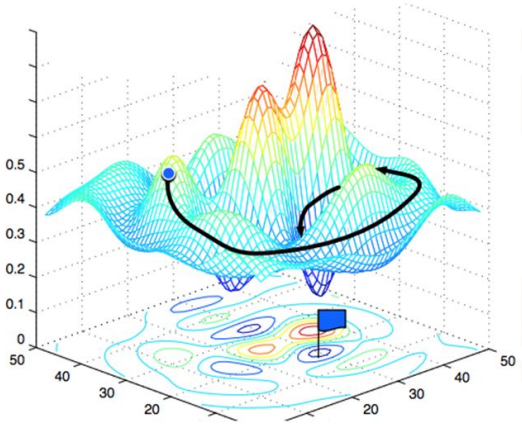
[1] https://livebook.manning.com/book/math-and-architectures-of-deep-learning/chapter-6/v-8/153
[2] https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
[3] https://stackoverflow.com/questions/58159154/how-to-calculate-categorical-cross-entropy-by-hand
[4] https://stats.stackexchange.com/questions/357963/what-is-the-difference-cross-entropy-and-kl-divergence

Extra!

Designing and training the neural network: summary

- Choose topology
  - Number of layers
  - Type (dense, convolutional, etc.) and activation function of each hidden layer
  - For output layer:
    - Type (typically dense)
    - Activation function (e.g., linear for regression and softmax for classification problems)
    - Encoding (e.g., one-hot)
- Choose training hyperparameters
  - Learning rate
  - Number of epochs (may use “early stopping” schemes)
  - Mini-batch size
  - Loss function (e.g., cross-entropy [1])
  - Dataset splitting in train, validation and test
  - Monitoring metrics
  - Optimizer (e.g. Adam, RMSprop, etc.)
  - Weights initialization
  - Regularization (L2, L1, Dropout layer, etc.)





[1] https://machinelearningmastery.com/cross-entropy-for-machine-learning

Applications of GenAI to wireless comm.

Summary of applications:

- ➔ Channel estimation with conditional GAN and diffusion model
- ➔ Channel generation with VAE
- ➔ Channel state information (CSI) compression with transformers
- ➔ Channel prediction with LLMs
- ➔ Beam management with LLMs
- ➔ LLM+RAG for question answering of 3GPP Release 18 documents

Wireless channels

[1] Va et al, Inverse Multipath Fingerprinting for Millimeter Wave V2I Beam Alignment, 2017
[2] Sayeed, "A Virtual MIMO Channel Representation and Applications", 2003
[3] Heath et al, An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems, 2016

Transmitter and receiver with multiple antennas: multiple input multiple output (MIMO) systems

Each MIMO channel is composed of several individual channels

Examples of antennas for the base station (BS)

Traditional 4G BS

Ericsson 64T64R Massive MIMO Antenna Radio Unit

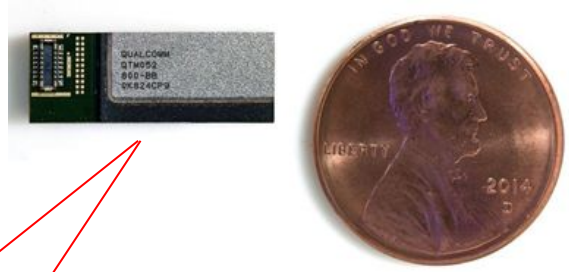
JTower glass antenna for sub 6 GHz frequencies [1]

[1] https://spectrum.ieee.org/5g-antenna-transparent-window , Sep. 2024

Examples of high frequency (mmWave) antenna array for 5G phones

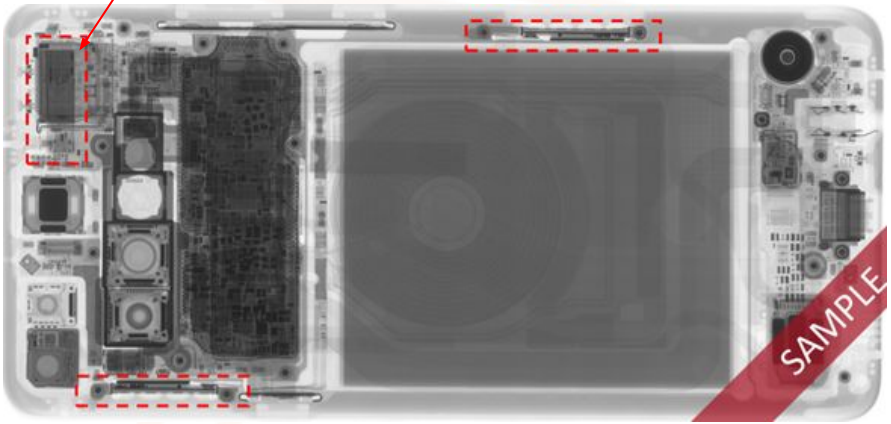
LASSE

Qualcomm QTM052 antenna module



QTM052 features:
- mmWave frequency bands (28 GHz, etc.)
- Estimated 64 antenna elements

Samsung Galaxy S10 5G (3 modules)

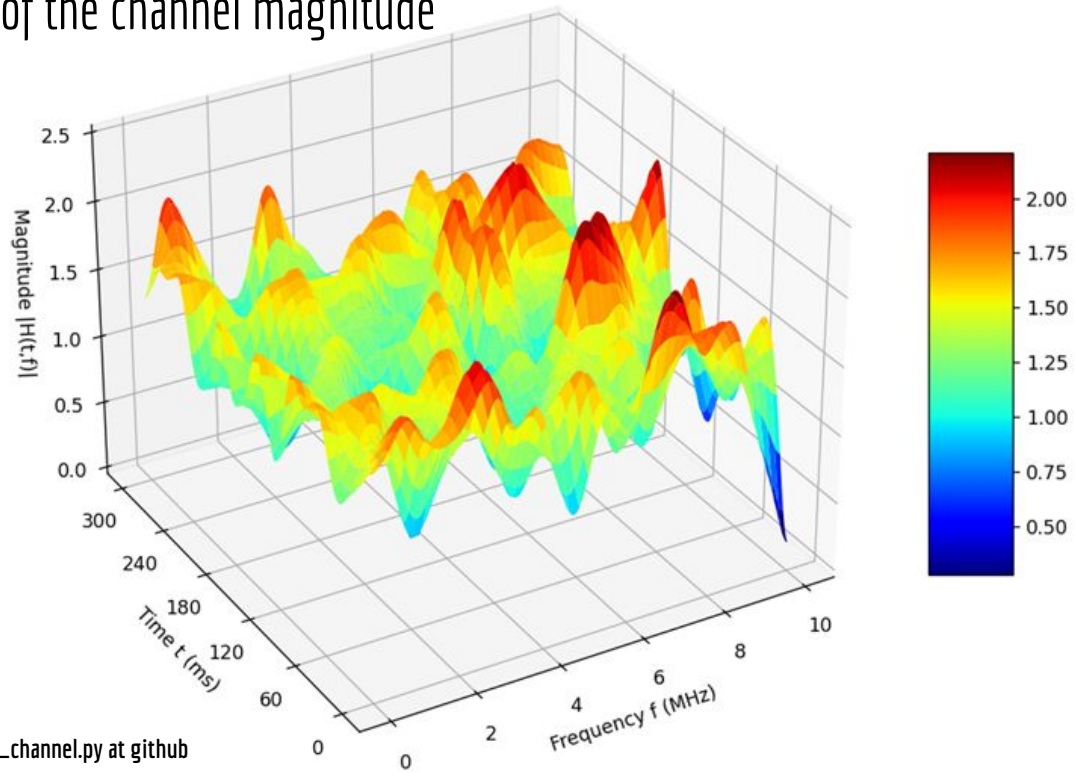


Motorola Moto Mod 5G (4 modules)

[1] SystemPlus Consulting – Motorola Mod 5G teardown
[2] Tech Insights - https://www.techinsights.com/blog/qualcomm-qtm052-mmwave-antenna-module
[3] https://www.microwavejournal.com/articles/31448-first-5g-mmwave-antenna-module-for-smartphones

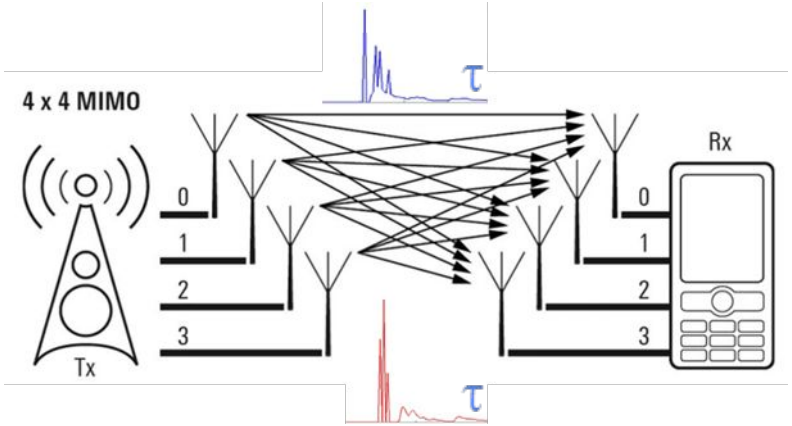
Wireless channel varies over time and frequency

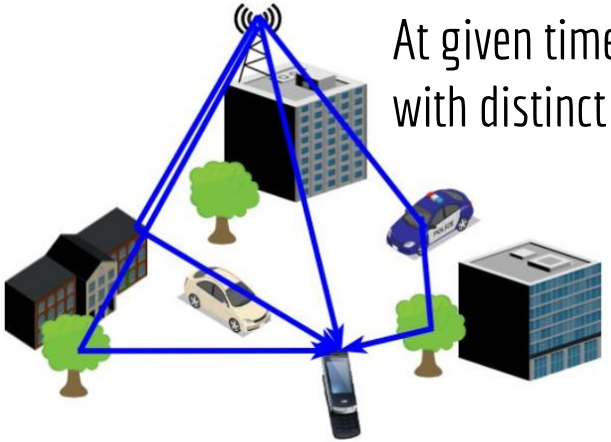
3D plot of the channel magnitude



Randomly created with
sbrt2024_shortcourse_gen_ai\useful_scripts\plot_channel.py at github

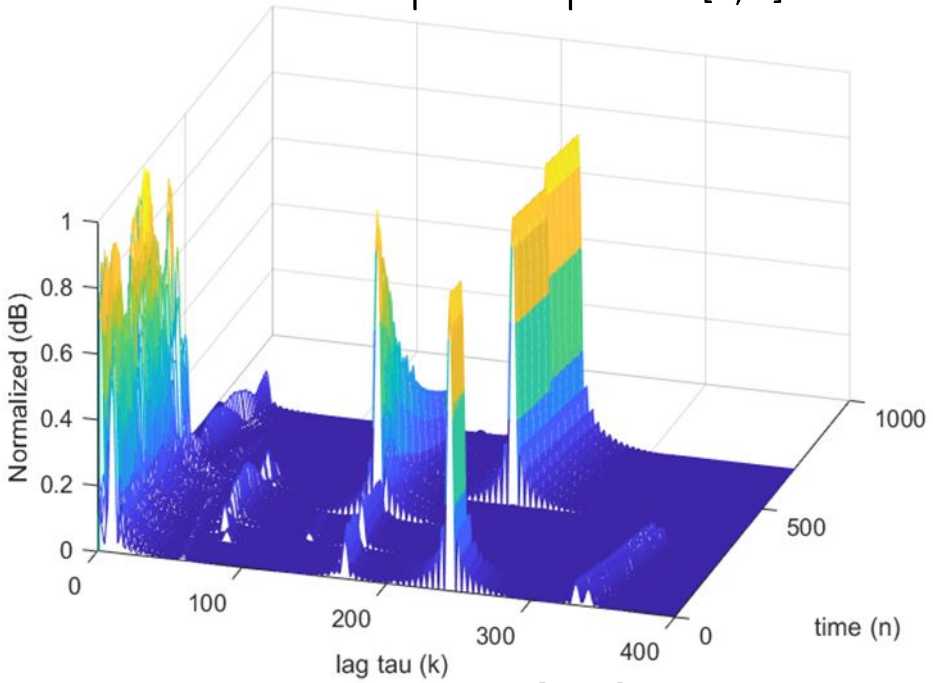
MIMO channels: time-varying linear systems





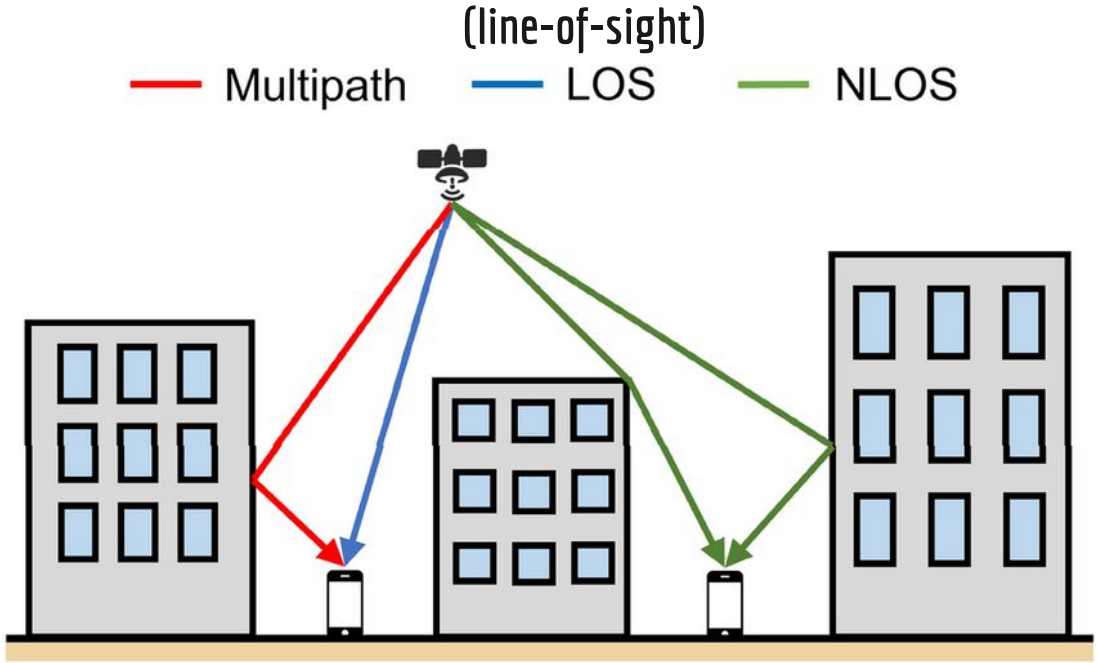
At given time, signal arrives with distinct delays (lags)

Discrete-time impulse response h[n, k]



At each time $n=n_0$, the DFT of the vector $h[n_0, k]$ leads to the channel frequency response $H[n_0, f]$ (also a vector)

Channel as linear combination of multipath components



Parameters that describe each multipath component (MPC):

- complex gain
- departure (RX) angles
- arrival (TX) angles
- delay

Example, for the k-th MPC [1]:

$$\left(L_k, \phi_k^{\mathrm{rx}}, \theta_k^{\mathrm{rx}}, \phi_k^{\mathrm{tx}}, \theta_k^{\mathrm{tx}}, \tau_k\right)$$

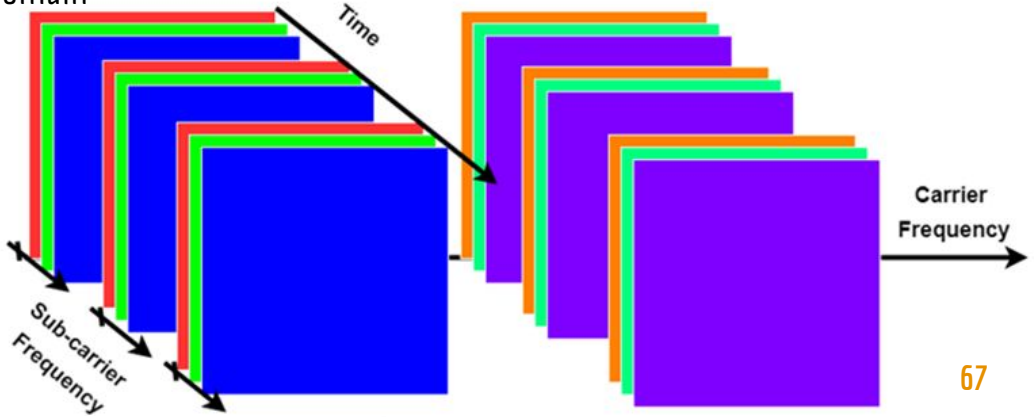
[1] Generative Neural Network Channel Modeling for Millimeter-Wave UAV Communication, https://ieeexplore.ieee.org/document/9782737
[2] Liu, Qi, et al. "NLOS signal detection and correction for smartphone using convolutional neural network and variational mode decomposition in urban environment." *GPS Solutions* 27.1 (2023): 31

Extra!

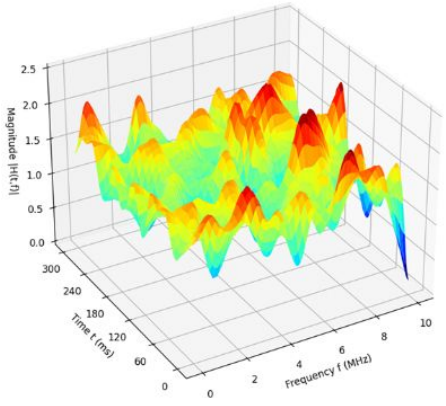
MIMO channels are sophisticated tensors

- MIMO wireless channels can be represented along 7 "dimensions": time n, carrier c and delay k (in time domain) or subcarrier frequency f (in frequency domain). Each of these triple requires four dimensions to represent real and imaginary parts for each pair of Tx and Rx
- At each discrete-time time instant n, a MIMO channel represents N_{rx} x N_{tx} individual channels
- Assuming P samples were used to represent the delays k, each individual channel is represented by a complex-valued vector with P values, k=0, ..., P-1
- Using Q-points DFT (eventually Q=P), in discrete frequency domain f (f is a subcarrier in OFDM), each individual channel can be represented by Q complex values, with f=0, ..., Q-1
- For a given carrier frequency c and time n, these vectors can be arranged as a 3D tensor, of dimension N_{rx} x N_{tx} x P in time-domain or dimension N_{rx} x N_{tx} x Q in frequency-domain

Lining up subcarriers lead to two "videos" (real/imag) for each carrier frequency



Simplifications for millimeter wave and THz channels: narrowband and sparse in angular (virtual) domain

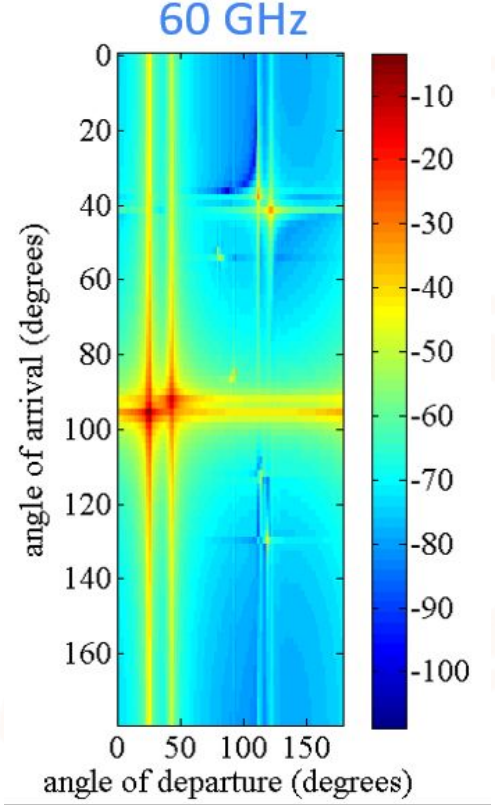


Narrowband assumption, at each time instant the channel is constant. Example: for Ntx = 3 and Nrx = 2 antennas, a MIMO matrix H can be used as:

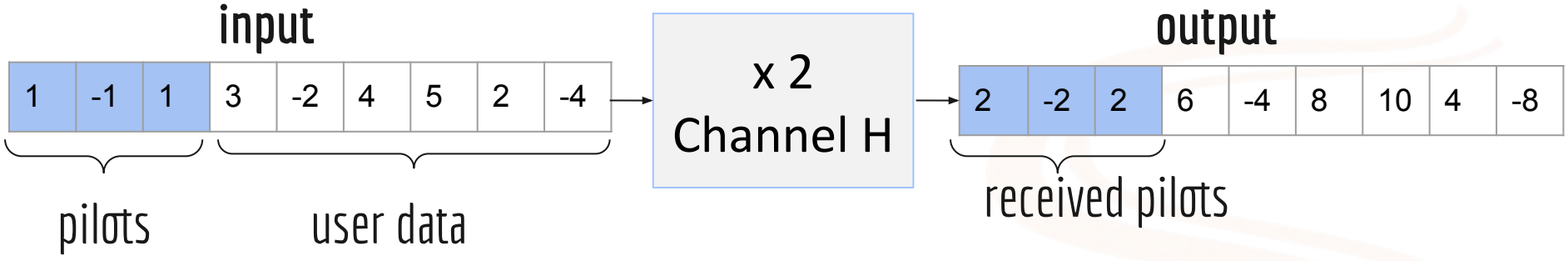
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 2+3i & -1-2i & 4+5i \\ 0+i & -3+4i & 6-i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

In **spatial domain**, H is not sparse, so use **angular domain** instead: H_v = 2D-DFT{ H }

Channel H_v



Introduction to channel estimation



Methods to obtain estimated channel $\hat{\mathbf{H}}$

- conventional least squares (LS)
- deep learning method with received pilots as inputs

Metric: normalized MSE

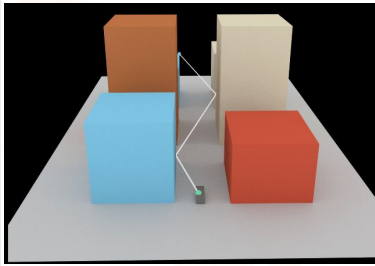
$$\mathrm{NMSE} = \frac{\mathbb{E}\left[\left\|\mathbf{H} - \hat{\mathbf{H}}\right\|^2\right]}{\mathbb{E}\left[\|\mathbf{H}\|^2\right]}$$

The receiver sends the channel state information (CSI) to the transmitter

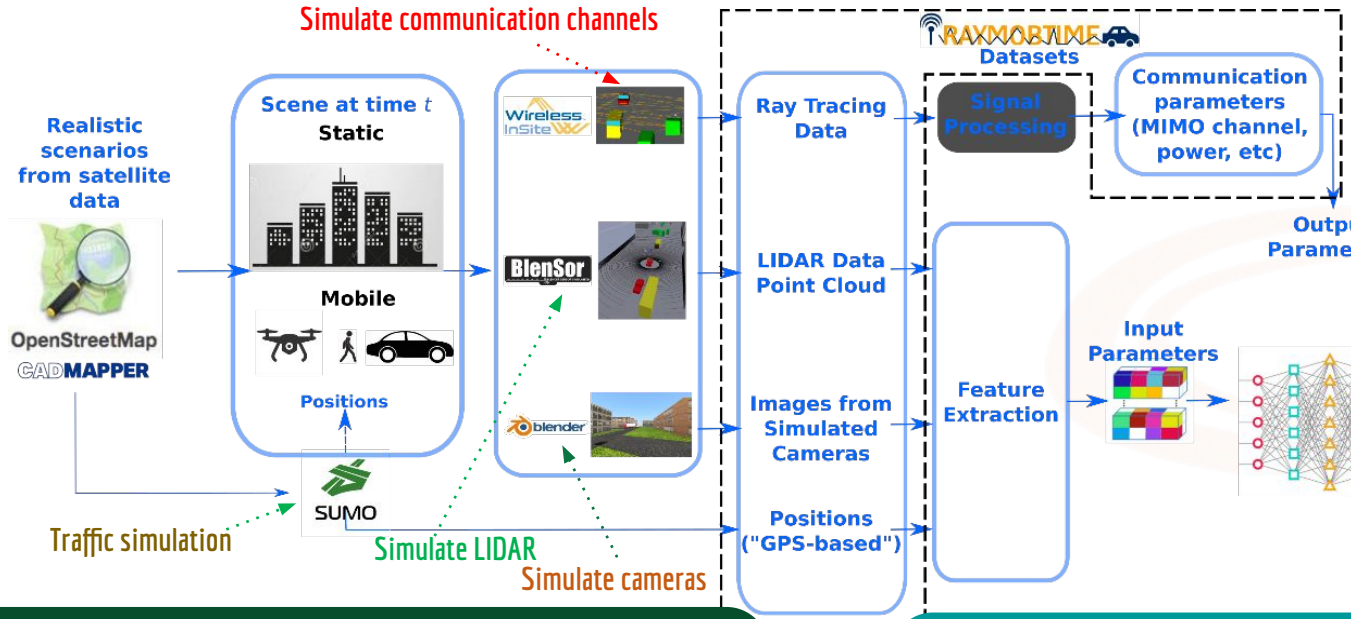
Wireless channel generation methods

Feature	Ray-Tracing Models	Statistical Models	Hybrid Models
Accuracy	High (detailed, site-specific)	Moderate (simplified, less environment-specific)	High, balances accuracy and complexity
Complexity	High (requires detailed 3D environment models)	Low to Moderate (less computationally intensive)	Moderate to High (depends on balance between methods)
Computation Time	Long (requires solving multiple paths, reflections, diffractions)	Fast (closed-form or simplified models)	Moderate (adaptive depending on scenario)
Cost of Setup	High (requires detailed knowledge of environment)	Low (generalized for many environments)	Moderate (can reuse statistical parts in complex scenarios)
Use in Standards	Rare in telecom standards (due to high complexity)	Used in early models, simplified propagation in 2G, 3G	Common (3GPP TR 38.901 for 5G)

Ray-tracing software:
- NVIDIA's Sionna
- Remcom's Wireles InSite



Raymobtime methodology



Example of rays obtained in a traffic jam simulation [1]

Raymobtime provides realistic channels obtained with ray-tracing, taking in account the mobility of transceivers and scatterers and their evolution over time

Multimodal datasets motivated by the increasing interest on designing integrated sensing and communication systems are also available

[1] A. Klautau, P. Batista, N. González-Prelcic, Y. Wang and R. W. Heath, "5G MIMO Data for Machine Learning: Application to Beam-Selection Using Deep Learning," in Proc. of the Information Theory and Applications Workshop (ITA), San Diego, CA, 2018, pp. 1-9. https://arxiv.org/abs/2106.05370

[2] A. Klautau and N. González-Prelcic, "Realistic simulations in Raymobtime to design the physical layer of AI-based wireless systems," ITU News Magazine, No. 5, 2020

Extra!

Available Raymobtime datasets

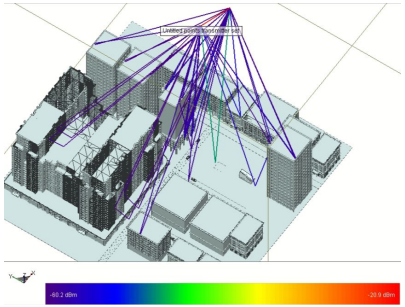
Id	Insite Version	3D Scenario	Carrier frequencies (GHz)	# Rx and type	Time between scenes	Time between episodes	# episodes	# scenes per episode	# valid channels
s000	3.2	Rosslyn	60	10 Mob	100 ms	30 s	116	50	41 K
s001	3.2	Rosslyn	2.8; 5	10 Fix	5 ms	37 s	200	10	20 K
s002	3.2	Rosslyn	2.8; 60	10 Fix	1 s	3 s	1800	1	18 K
s003	3.2	Rosslyn	2.8; 5	10 Fix	1 ms	35 s	200	10	20 K
s004	3.2	Rosslyn	60	10 Mob	1 s	30 s	5000	1	35 K
s005	3.2	Rosslyn	2.8; 5	10 Fix	10 ms	35 s	125	80	100 K
s006	3.2	Rosslyn	28; 60	10 Fix	1 ms	35 s	200	10	20 K
s007	3.3	Beijing	2.8; 60	10 Mob	1 s	5 s	50	40	15 K
s008	3.2	Rosslyn	60	10 Mob	100 ms	30 s	2086	1	20 K
s009	3.3	Rosslyn	60	10 Mob	500 ms	10.5 s	2086	42	27 K

Multimodal:

More than 400 k channels (multimodal, paired frequency bands, distinct sites, etc.)

[1] https://www.lasse.ufpa.br/raymobtime/

CAVIAR: simulation of Communication networks, Artificial intelligence and computer VIsion with 3D computer-generAted imageRy



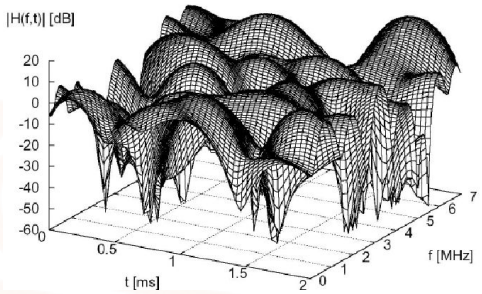
Corresponding ray-tracing

CAVIAR [1, 2] allows simulating 5G / 6G networks in AI + CV applications such as search and rescue using UAVs

CAVIAR is open source and also enables simulation of advanced digital twins

[1] A. Klautau at al, "Generating MIMO Channels for 6G Virtual Worlds Using Ray-Tracing Simulations," IEEE SSP Workshop, July 2021
[2] A. Oliveira et al, "Simulation of Machine Learning-Based 6G Systems in Virtual Worlds," ITU Journal on Future and Evolving Tech., 2021
[3] F. Bastos et al, "The CAVIAR framework: Simulation of 5G/B5G systems in virtual worlds", ITU Kaleidoscope Video Demo, Dec. 2021
[4] J. Borges et al, "CAVIAR: Co-Simulation of 6G Communications, 3-D Scenarios, and AI for Digital Twins", IEEE Internet of Things Journal, Oct. 2024

Some philosophy: wireless channel datasets and Plato's allegory of the cave



Actual channel

Dataset users and readers of their papers

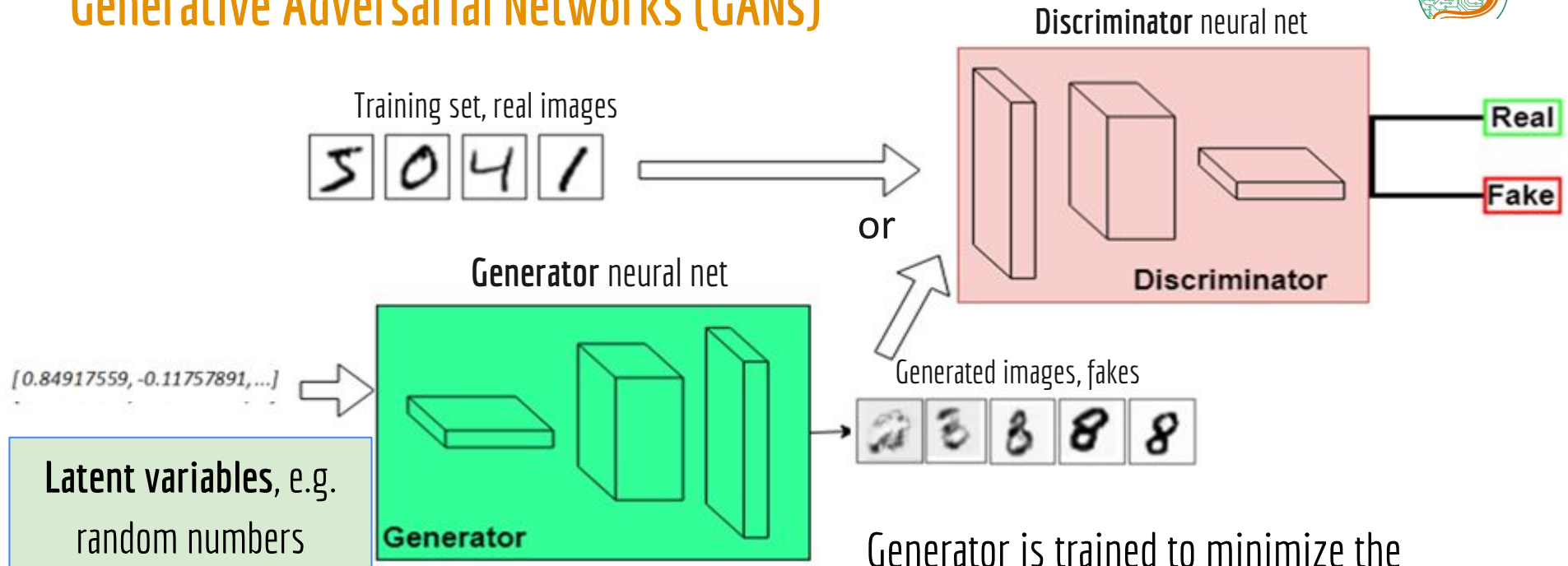
Dataset creators using ray-tracing, 3GPP 38.901, Rayleigh, etc.

"Privileged" people with access to expensive $ field measurements

Part II - GenAI Applied to RAN (towards AI-native)

Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs)

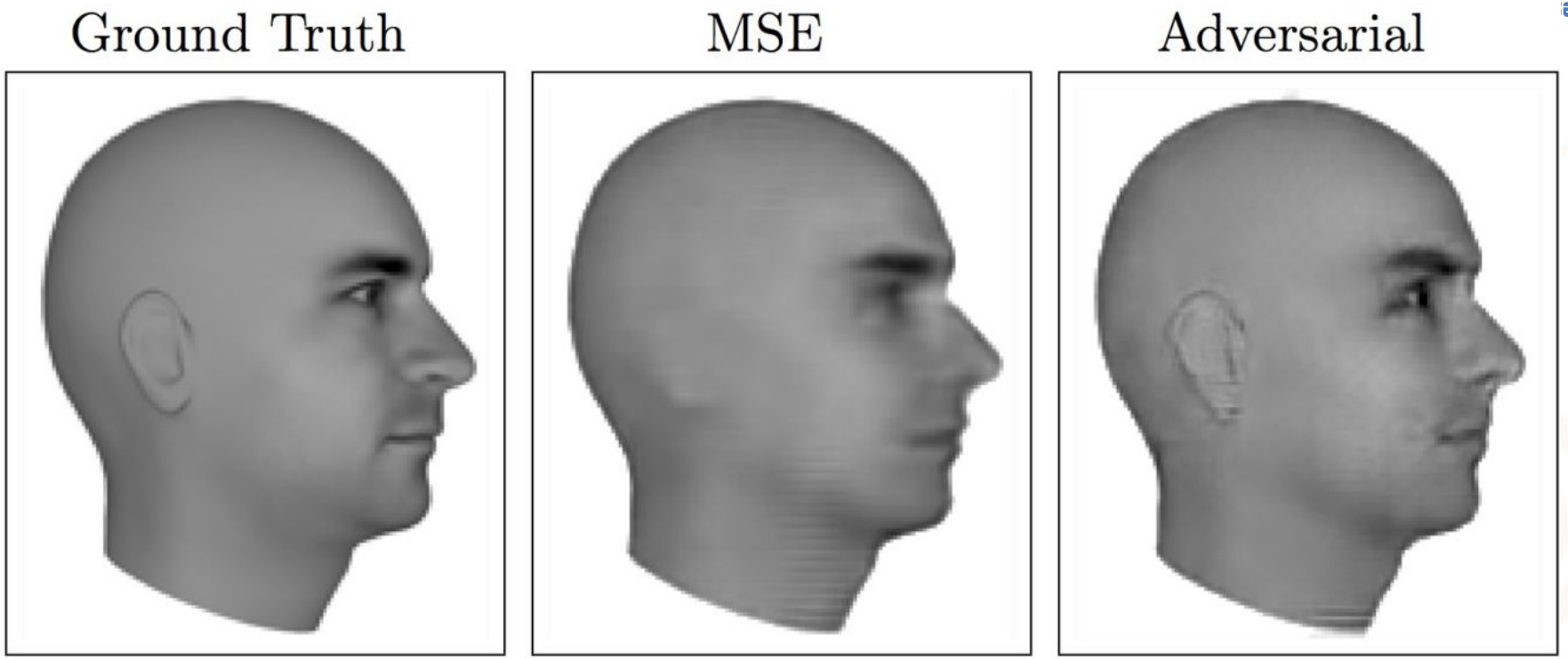


Latent variables, e.g. random numbers created with rand( ) function

Generator is trained to minimize the **adversarial** loss, which differs from conventional ones such as **MSE**

[1] **Generative Adversarial Networks** (2016, Ian Goodfellow), https://arxiv.org/pdf/1701.00160.pdf
[2] **Machine Learning for CSI Recreation in the Digital Twin Based on Prior Knowledge** (2022, Brenda Vilas Boas et al.), https://ieeexplore.ieee.org/document/9897088

GAN's adversarial loss is better than MSE



GenAI trend: reconstruction losses (MSE, etc.) are not enough.
Use KL divergence, etc. to model ("regularize") distributions

User does not directly control what vanilla GANs generate

The generator entries are random and **we don't have control over their outputs**

$z = (0.3, 0.2, -0.6, ...) \xrightarrow{G(z)}$ 

$z \sim \mathcal{N}(0, 1)$

or

$z \sim U(-1, 1)$

The discriminator guides the generator *G(z)* learning to create handwritten numbers but we cannot specify the number we want

$z = (-0.1, 0.1, 0.2, ...) \xrightarrow{G(z)}$ 

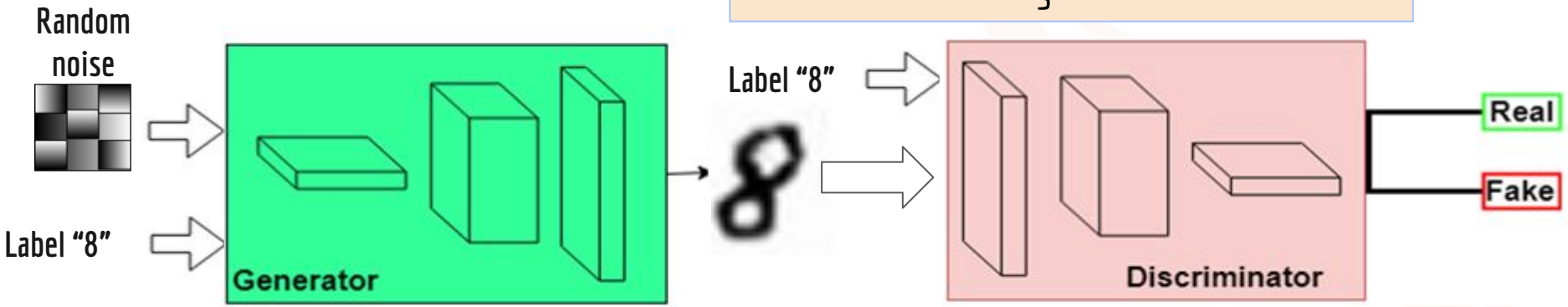
In vanilla GANs **you cannot ask the generator to create digit "7" or "6"**. The model randomly generates images similar to the ones in the training set

GAN — **What is Generative Adversarial Networks GAN?** (2018, Jonathan Hui), https://jonathan-hui.medium.com/gan-whats-generative-adversarial-networks-and-its-application-f39ed278ef09

Conditional GANs allow to specify characteristics of the output

Conditional GANs receive **random noise** + **label** to generate an image belonging to this label class

"Is it a real or fake image?" **and** "Is it digit 8?"



Generator is conditioned for generating digits according to the labels it receives

We learned how to control the latent space!

Image to image translation with Cycle GAN and NVIDIA Style GAN

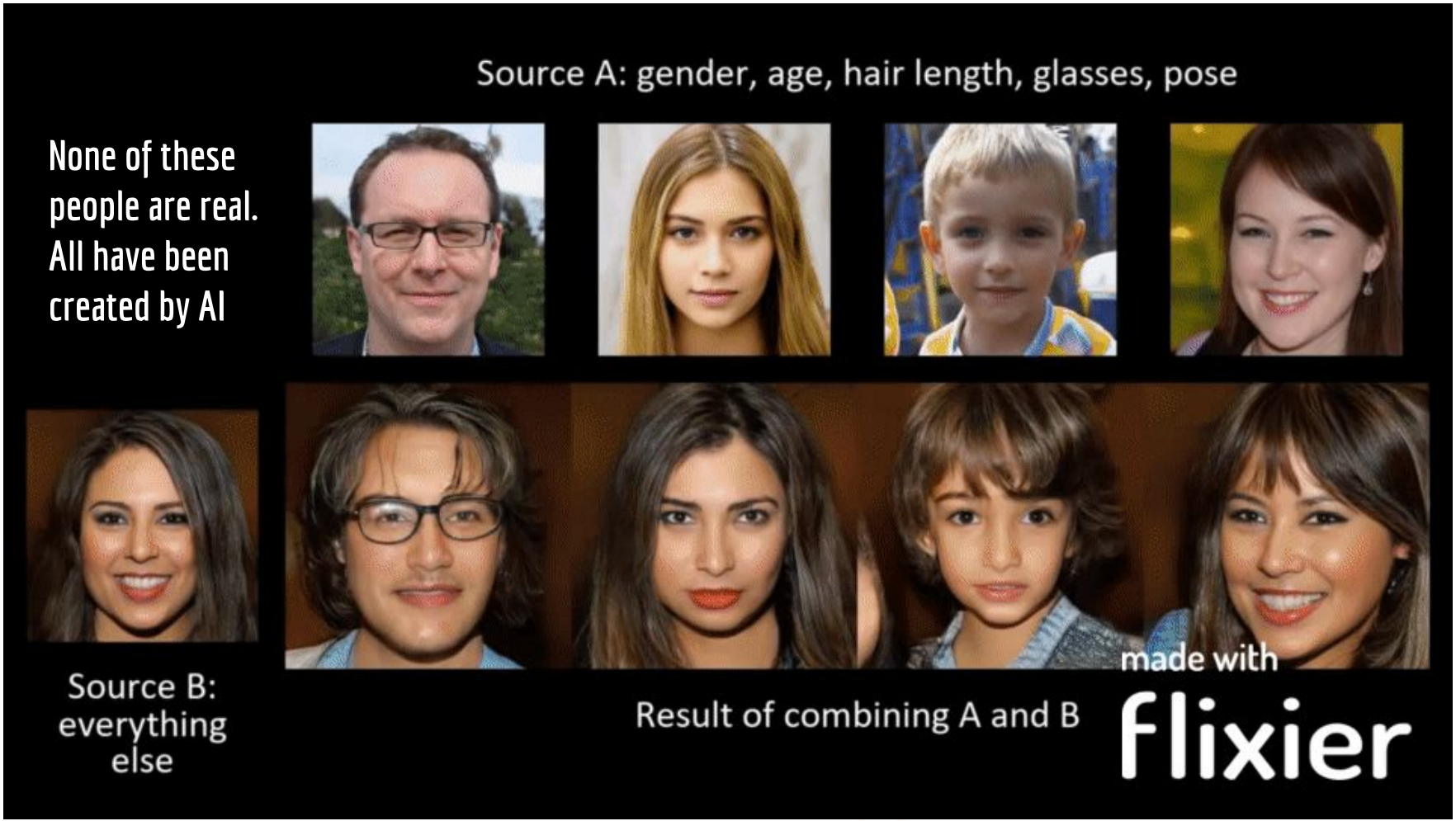
Source A: gender, age, hair length and pose

Source B: all other features

GAN combination of A and B

[1] **Cycle GAN** - https://arxiv.org/pdf/1703.10593
[2] **Style GAN** - https://ieeexplore.ieee.org/document/8953766

None of these people are real. All have been created by AI



[1] https://www.youtube.com/watch?v=kSLJriaOumA&ab_channel=TeroKarrasFI

Application of GAN to Telecom

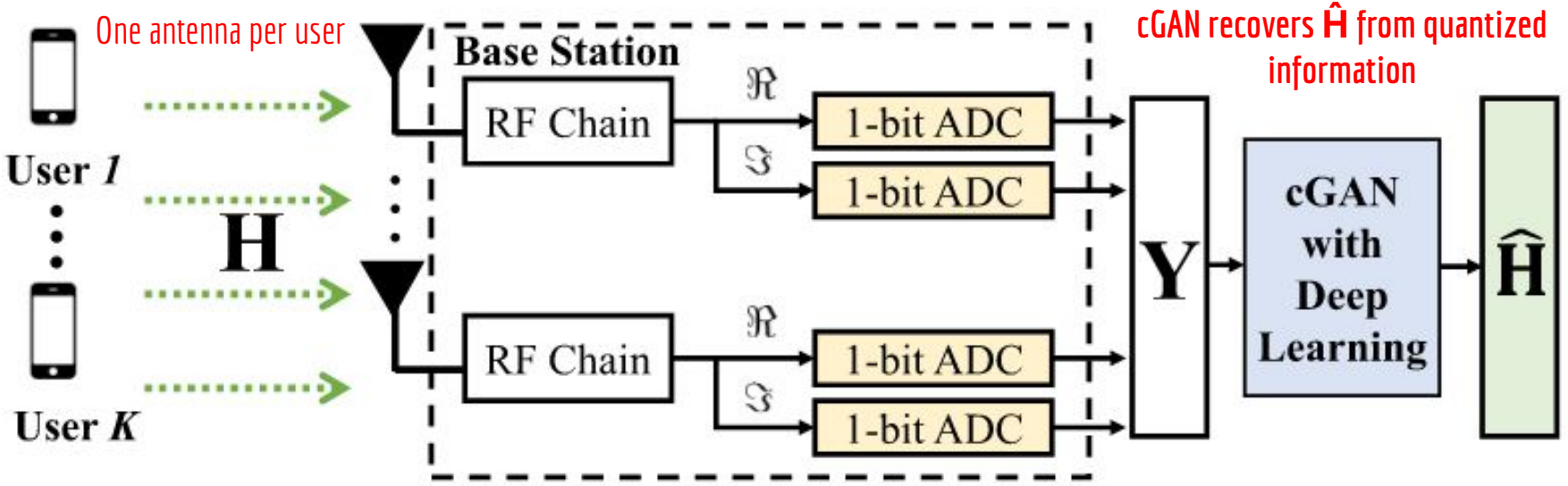
Channel estimation for one-bit multiuser massive MIMO using conditional GAN

Yudi Dong, Huaxia Wang and Yu-Dong Yao

Link to paper: https://ieeexplore.ieee.org/document/9246559 , 2020

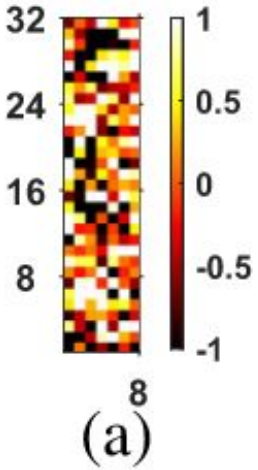
cGAN application: "Channel Estimation for One-Bit Multiuser Massive MIMO Using Conditional GAN"

Using simple One-Bit ADCs to reduce power consumption and hardware complexity



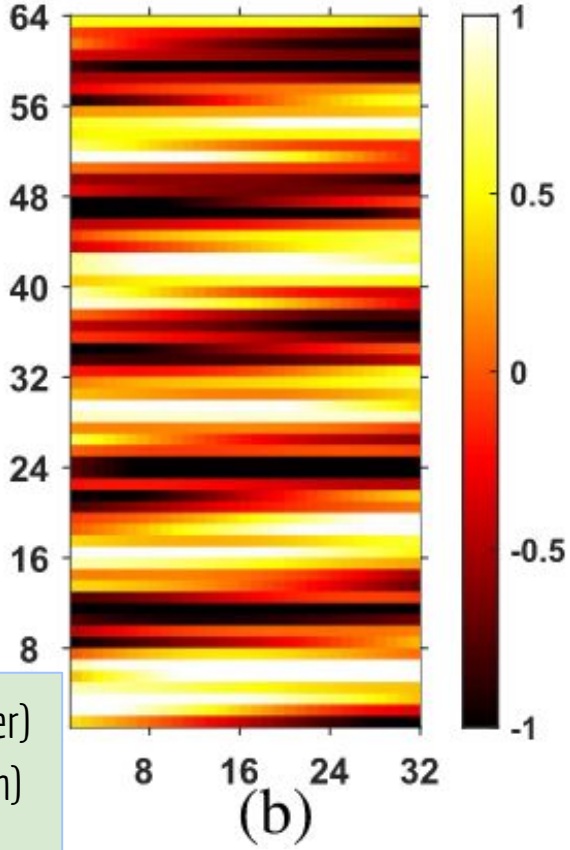
Channel estimation for one-bit multiuser massive MIMO using conditional GAN. IEEE Communications Letters. (2020, Yudi Dong) - https://ieeexplore.ieee.org/document/9246559

Pilots Φ:
Number Users x
Sequence length



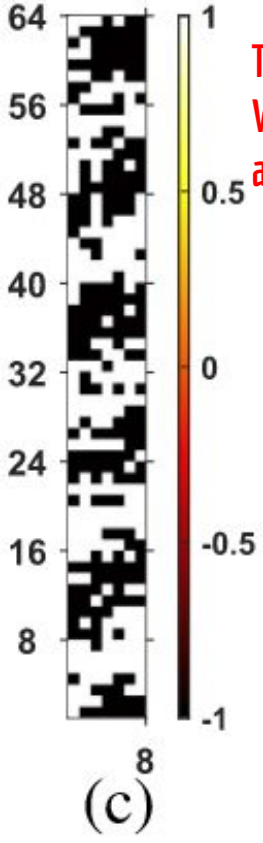
(a)

Channel Matrix H:
Rx Antennas x Number Users



(b)

Received Pilots Y:
Rx Antennas x Sequence length



(c)

The channel generation uses the Wireless InSite ray-tracing to build an indoor sub-6G massive MIMO.

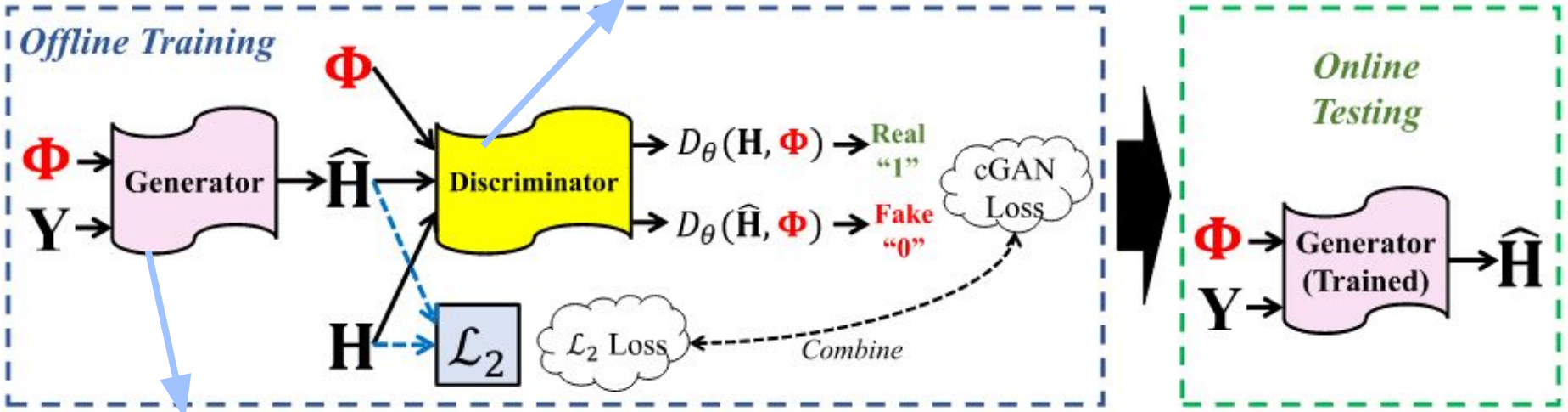
$$\mathbf{Y} = \mathrm{sgn}(\mathbf{H\Phi} + \mathbf{N})$$

Noise matrix

$$\mathrm{sgn}(x) = \begin{cases} 1 & , x \geq 0 \\ -1 & , \text{otherwise} \end{cases}$$

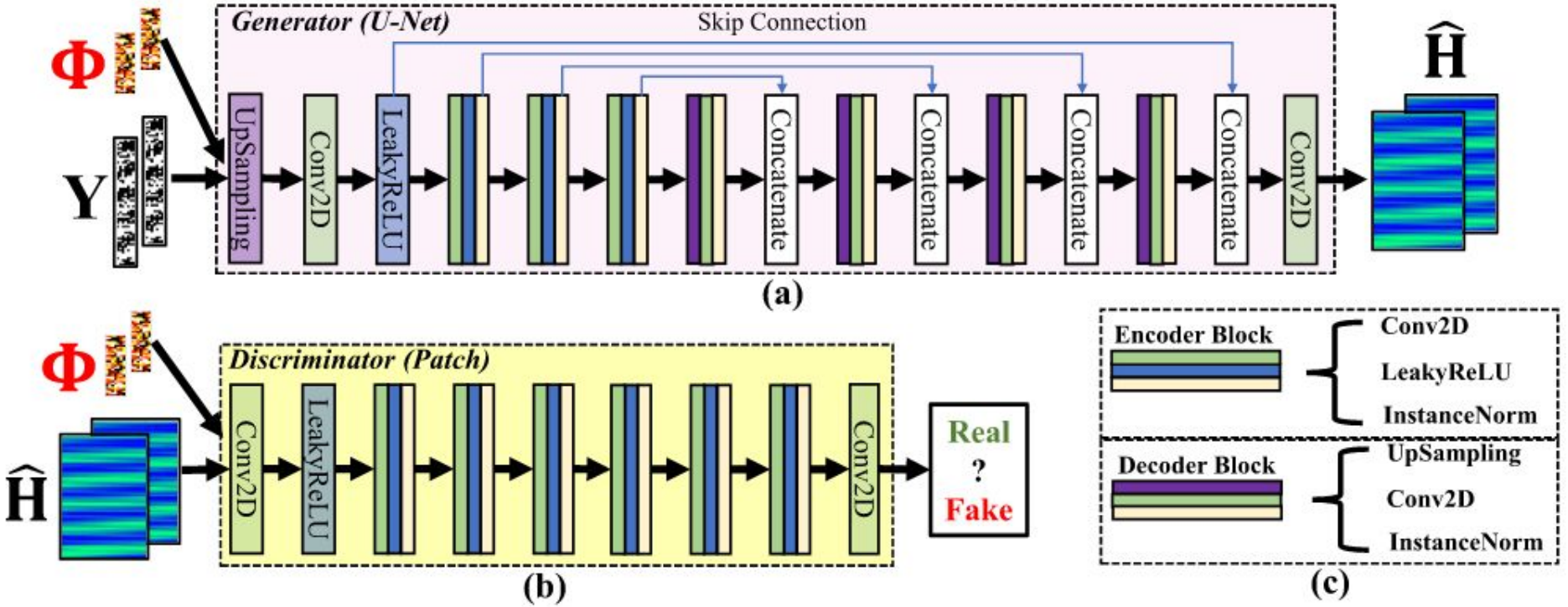
Number users: 32 (1 ant./user)
Rx Antennas: 64 (Basestation)
Pilot duration: 8 symbols

The Discriminator is a binary classifier responsible for identifying the entries as real or fake samples



Conditional GAN uses Φ and Y as inputs to reduce the output unpredictability

Once the algorithm has trained, only the generator is needed for production



A. Generator architecture
B. Discriminator architecture
C. Composition of encoder and decoder block

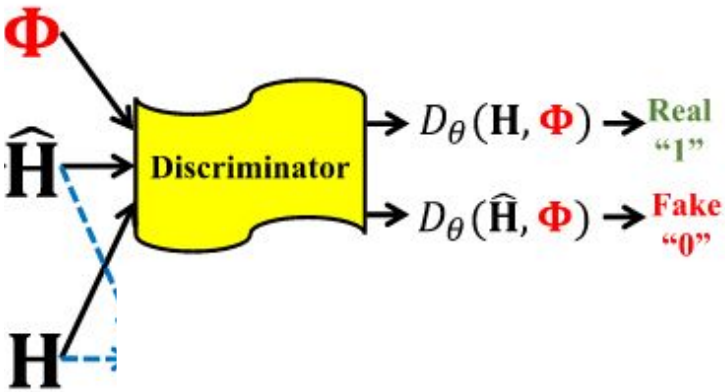
Jupyter notebook implementation presented by Cleverson Nahum

Available at: https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

Folder: Channel_Estimation_cGAN

Discriminator objective

The discriminator wants to **maximize the probability of classifying real samples as real "1"and fake samples as fake "0"**

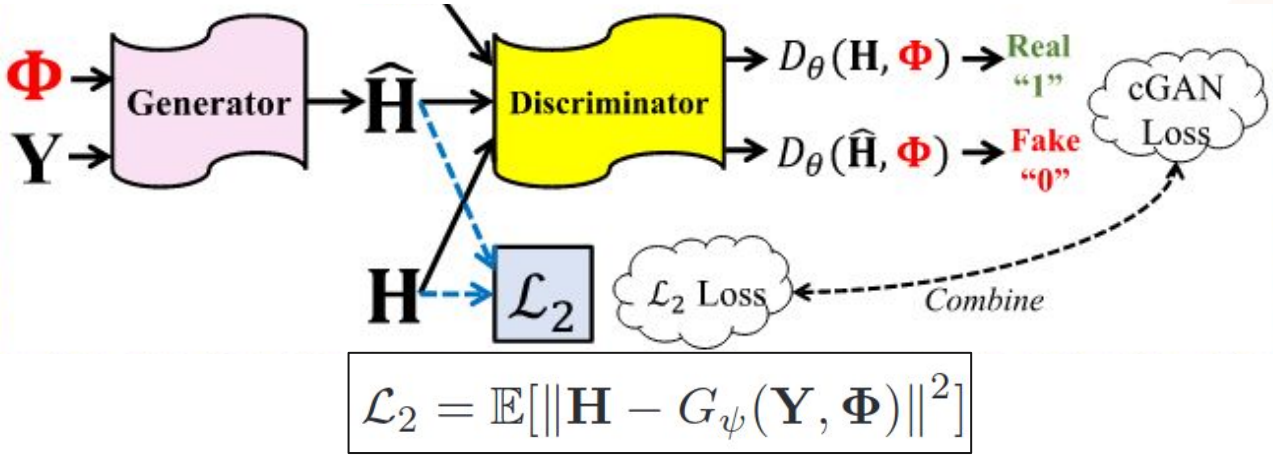


The discriminator D_θ is a probability function that outputs values near "1" when the sample is real and "0" when it is fake

$$\mathcal{L}_{\text{GAN}}(G_\psi, D_\theta, \mathbf{Y}, \mathbf{H}, \mathbf{\Phi}) = \mathbb{E}[\log D_\theta(\mathbf{H}, \mathbf{\Phi})] + \mathbb{E}[\log(1 - D_\theta(G_\psi(\mathbf{Y}, \mathbf{\Phi}))]$$

Generator loss

The generator wants to **fool the discriminator** by generating samples close to the real samples and **decreasing the L_2 in relation to the correct "label" channel H.**



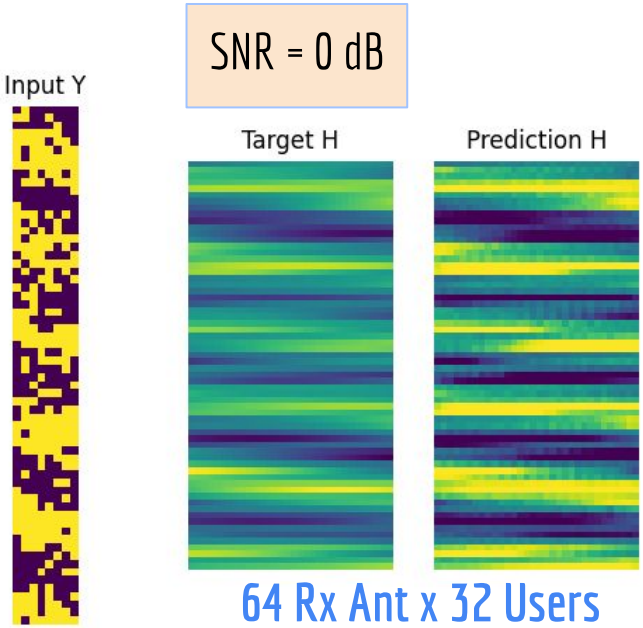
Includes a L_2 loss term so the generator learn to create channel predictions close to the label

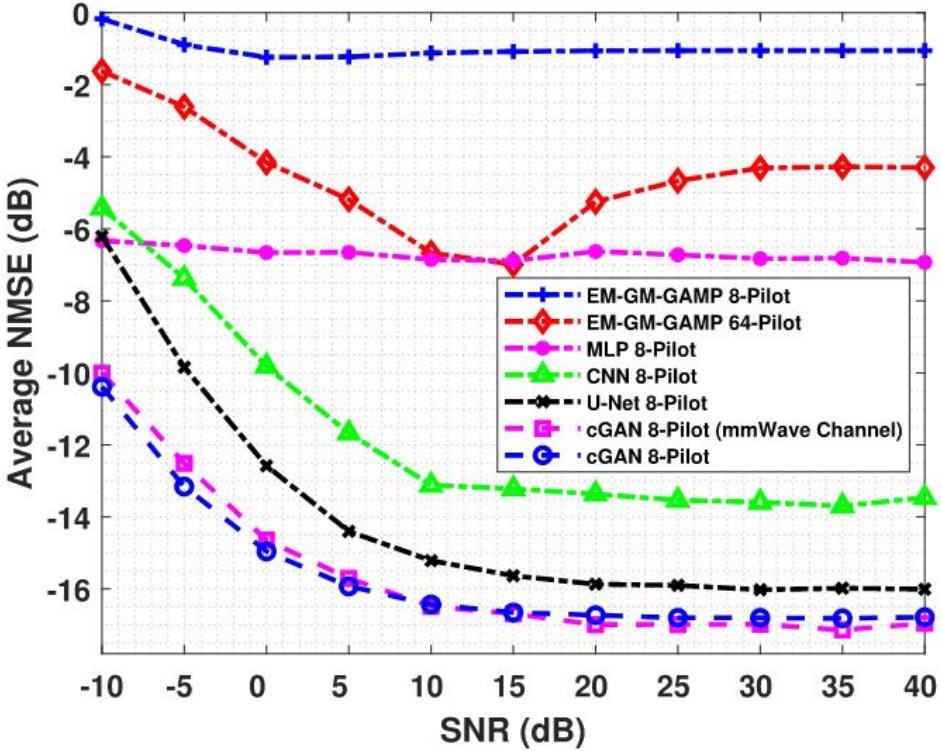
$$\mathcal{L}_2 = \mathbb{E}[\|\mathbf{H} - G_\psi(\mathbf{Y}, \mathbf{\Phi})\|^2]$$

Generator loss function:

$$\mathcal{L}_{GAN}(G_\psi, D_\theta, \mathbf{Y}, \mathbf{H}, \mathbf{\Phi}) + \mathcal{L}_2$$

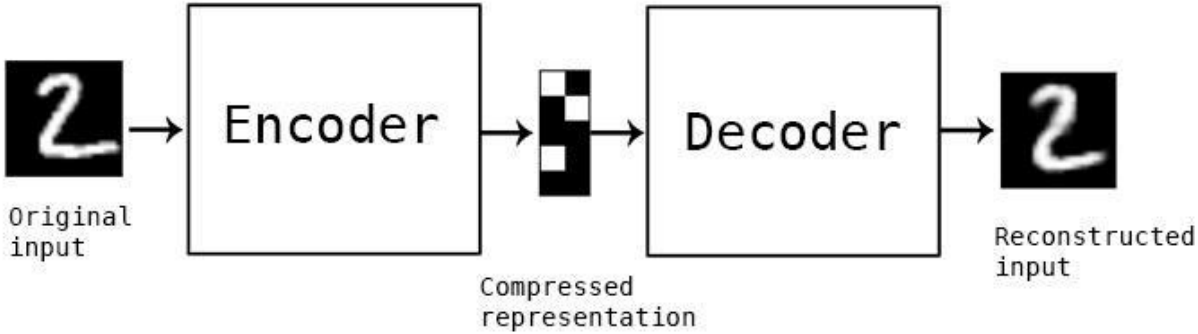
Results for each SNR value





Autoencoders

Latent space for Autoencoder and GANs



Train a single network (encoder+decoder) and later split it into two

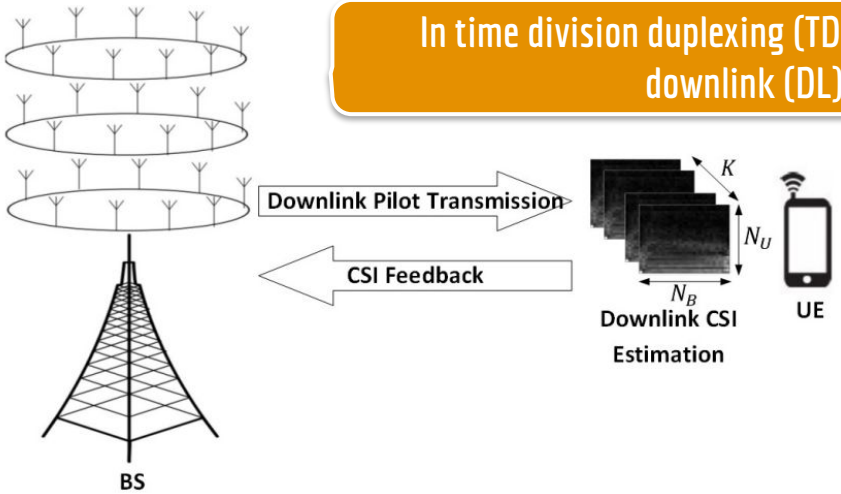
Latent (we cannot observe) -› compressed representation

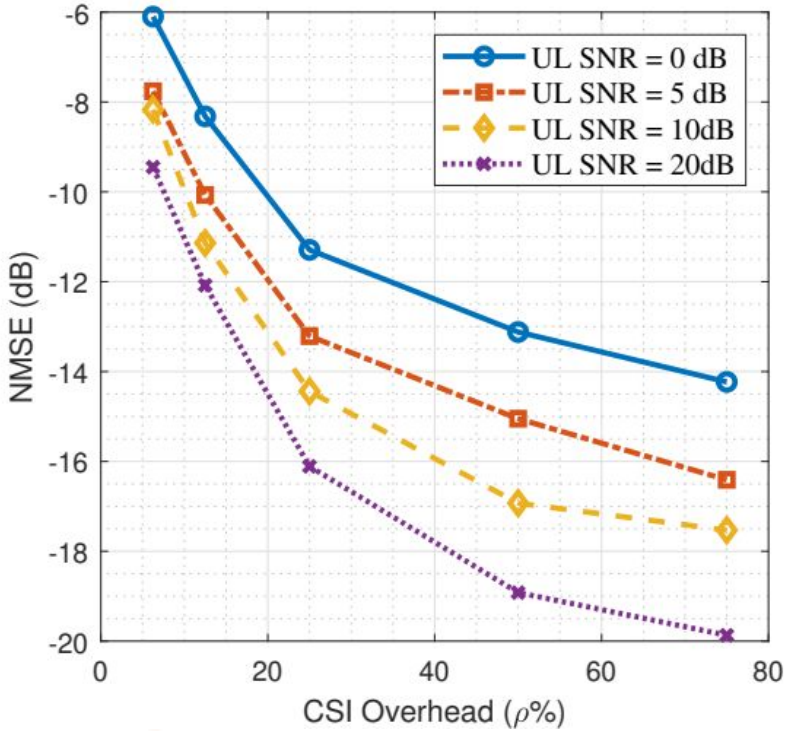
The vanilla autoencoder uses a deterministic mapping to latent space

[1] https://www.biorxiv.org/content/10.1101/811661v2.full
[2] https://blog.curso-r.com/posts/2017-06-26-construindo-autoencoders/

CSI compression (in frequency division duplexing, FDD)

In time division duplexing (TDD) systems, we often assume channel reciprocity such that the downlink (DL) channel is considered equal to the UL channel





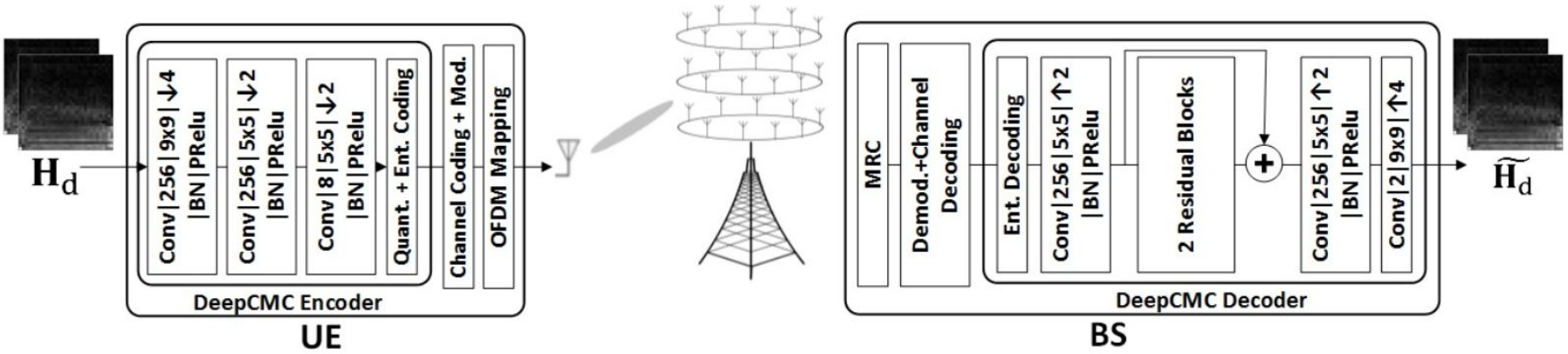
UE uses an autoencoder DNN to compress information and reduce overhead

N_B - # BS (T_x) antennas
N_U - num UE (R_x) antennas
K - # of OFDM subcarriers

Deep Learning for Massive MIMO Channel, State Acquisition and Feedback, Mahdi Mashhadi and Deniz Gunduz, 2020, https://arxiv.org/pdf/2002.06945.pdf

Extra!

CSI compression using autoencoder



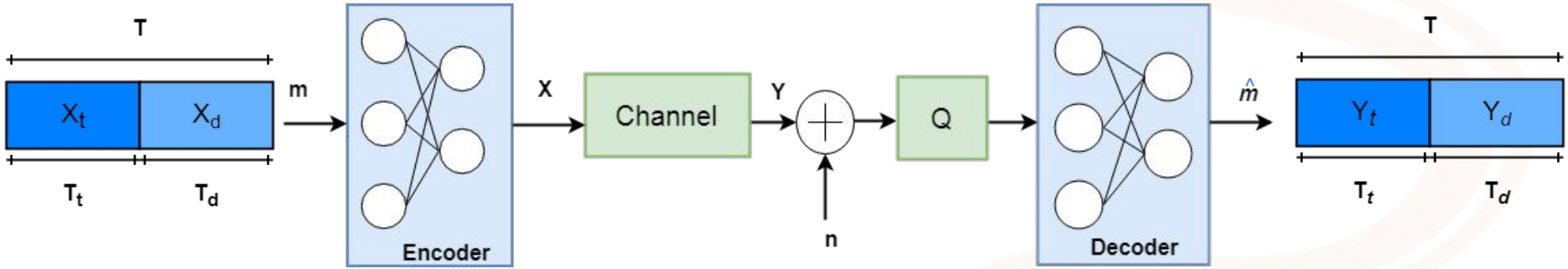
BN - batch normalization
PRelu - parametric ReLU
Quant. - quantization
Ent. Coding - entropy coding
Mod. - modulation

MRC - maximum ratio combining
Demod. - demodulation
Ent. Decoding - entropy decoding

[1] Deep Learning for Massive MIMO Channel, State Acquisition and Feedback, Mahdi Mashhadi and Deniz Gunduz, 2020, https://arxiv.org/pdf/2002.06945.pdf

End-to-end learning with autoencoder

Learn full transmitter and receiver implementations which are optimized for a specific performance metric and channel model [1]



[1] O'Shea et al, Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention, 2016

Extra!

Issue of taking in account the wireless channel in end-to-end autoencoder

LASSE

How to backpropagate the gradient?
☐ Previous works used AWGN or one-dimensional ("h") Rayleigh

Examples of proposed solutions for dealing with channel in end-to-end learning:

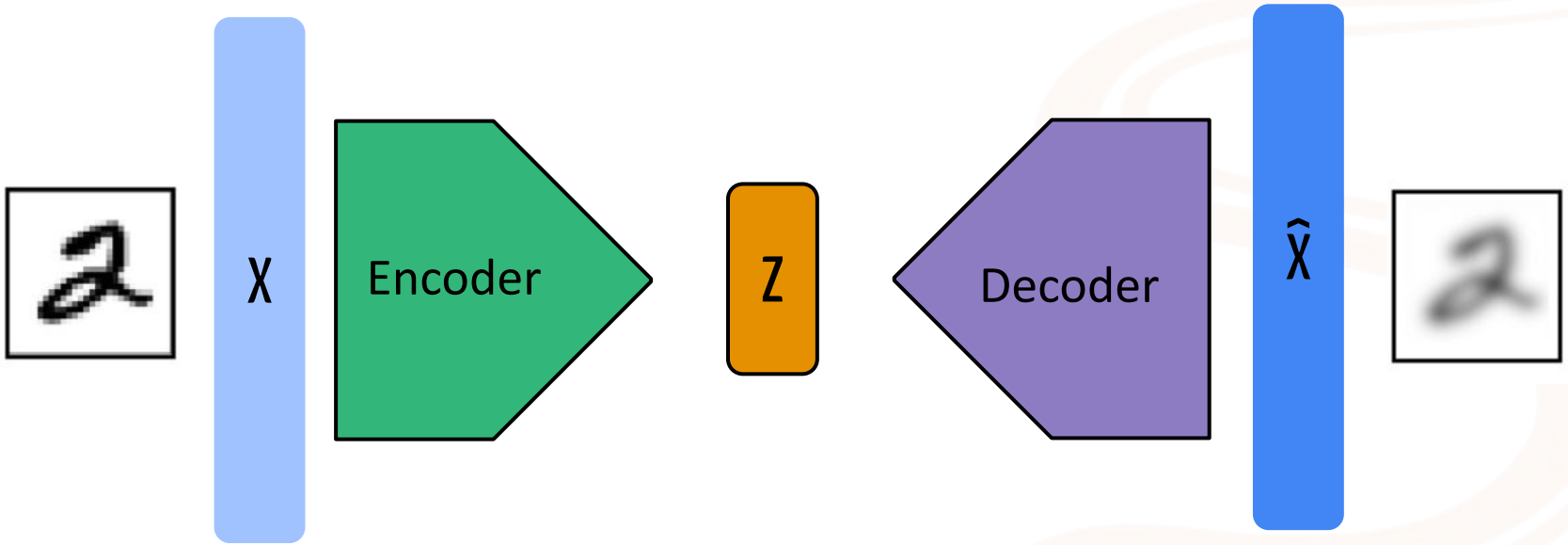
- Do not optimize the transmitter [1]
- Iterate between a) supervised learning of the receiver, b) reinforcement learning of Tx based on estimated gradient of the Rx loss [2]
- Calculate approximate gradients, bypassing the channel [3]
- Use a generative adversarial network (GAN) as a surrogate model [4, 5]

[1] Dörner et al, Deep learning-based communication over the air, 2018
[2] Aoudia and Hydis, End-to-End Learning of Communications Systems Without a Channel Model, 2018
[3] Raj and Kalyani, Backpropagating Through the Air: Deep Learning at Physical Layer Without Channel Models, 2018
[4] O'Shea et al, Physical Layer Communications System Design Over-the-Air Using Adversarial Networks, 2018
[5] Ye et al, Channel Agnostic End-to-End Learning based Communication Systems with Conditional GAN, 2018
[6] Sionna's Python simulation: https://nvlabs.github.io/sionna/examples/Autoencoder.html

Variational Autoencoders (VAEs)

Traditional ("vanilla") autoencoders

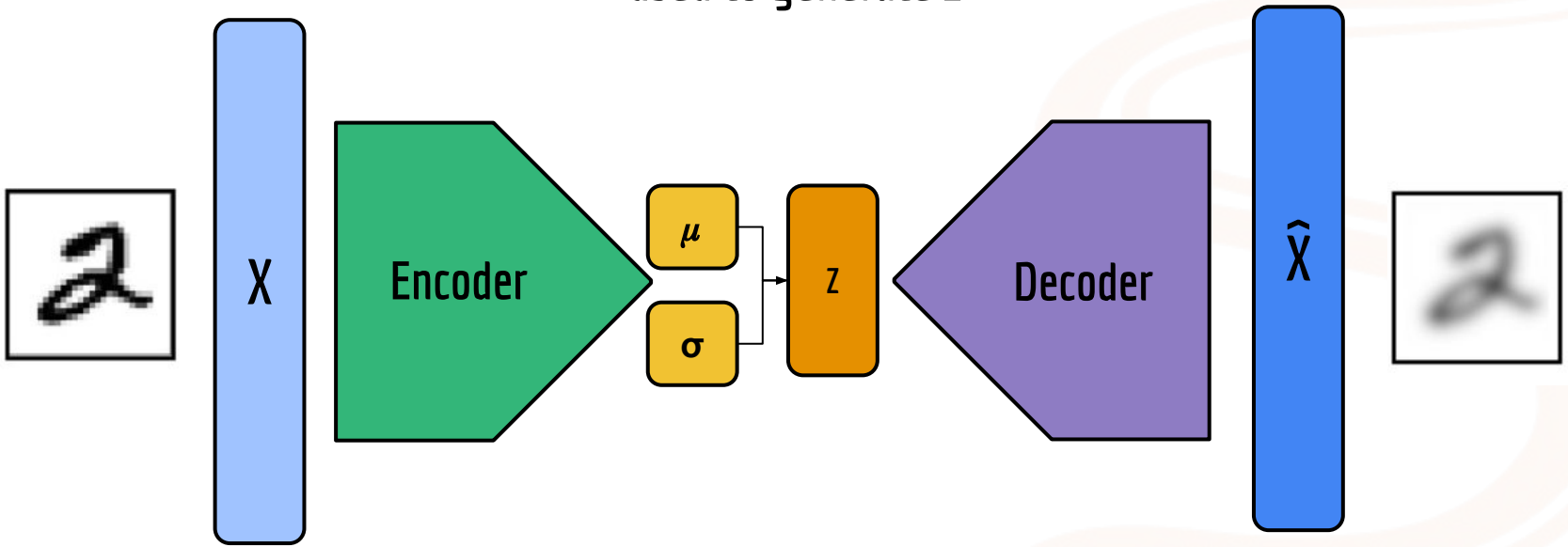
The mappings to z and from z are both deterministic



Based on MIT 6.S191 Introduction to Deep Learning (http://introtodeeplearning.com/)
https://medium.com/@weidagang/demystifying-neural-networks-variational-autoencoders-6a44e75d0271
https://medium.com/geekculture/variational-autoencoder-vae-9b8ce5475f68
https://medium.com/accredian/building-intuition-variational-autoencoders-vaes-9c0908903f93

Variational Autoencoders (VAEs)

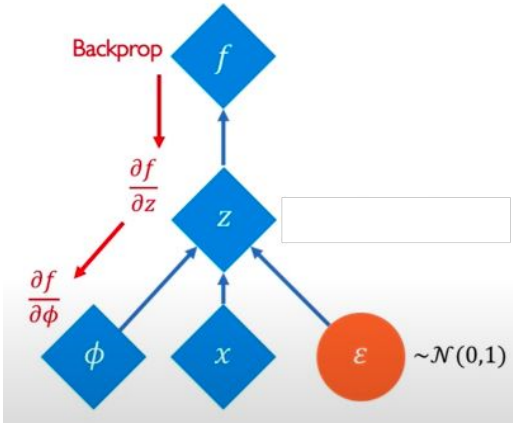
The encoder maps X into a probability distribution, which is then used to generate z

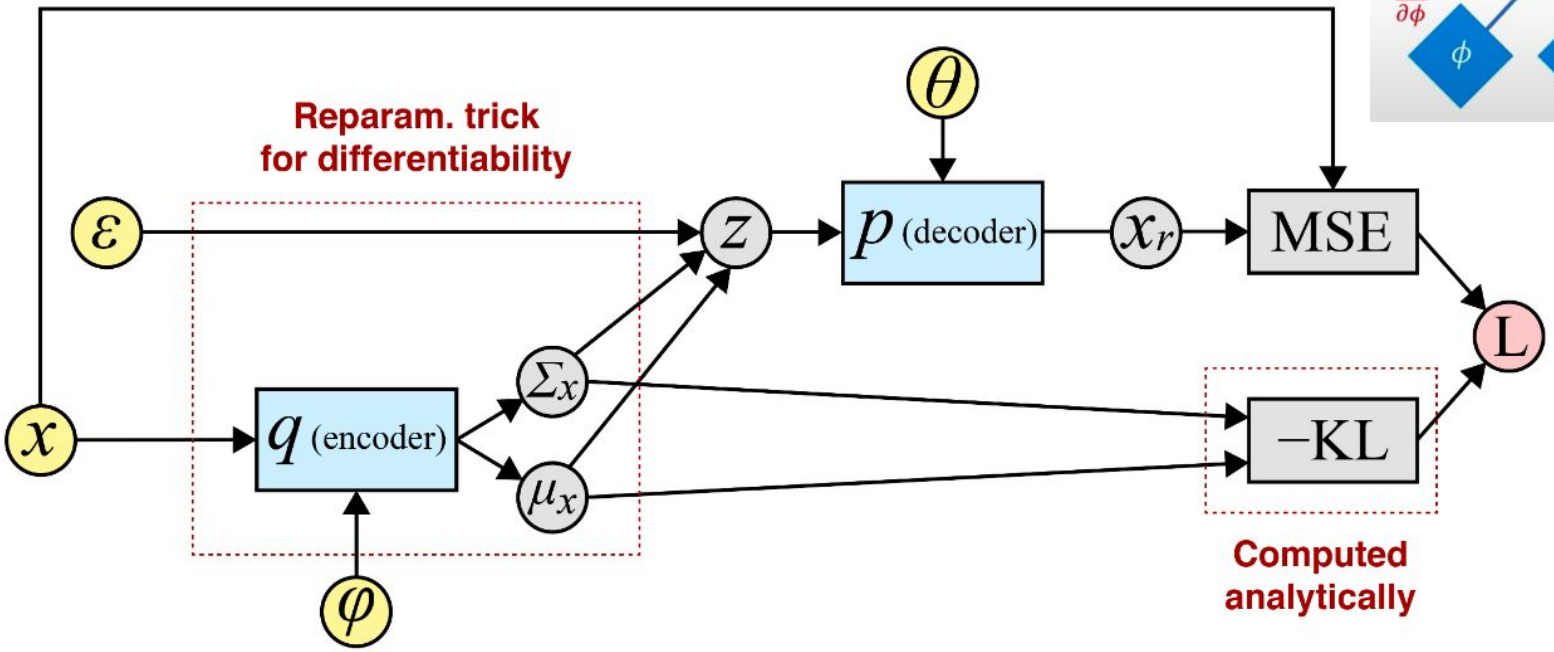


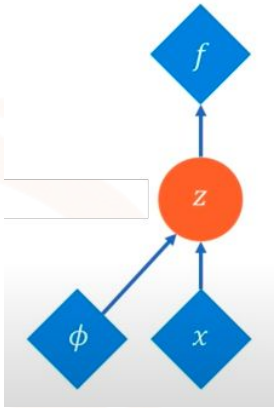
Sample from the mean and standard deviation to compute latent vector

Extra!

VAE reparameterization trick



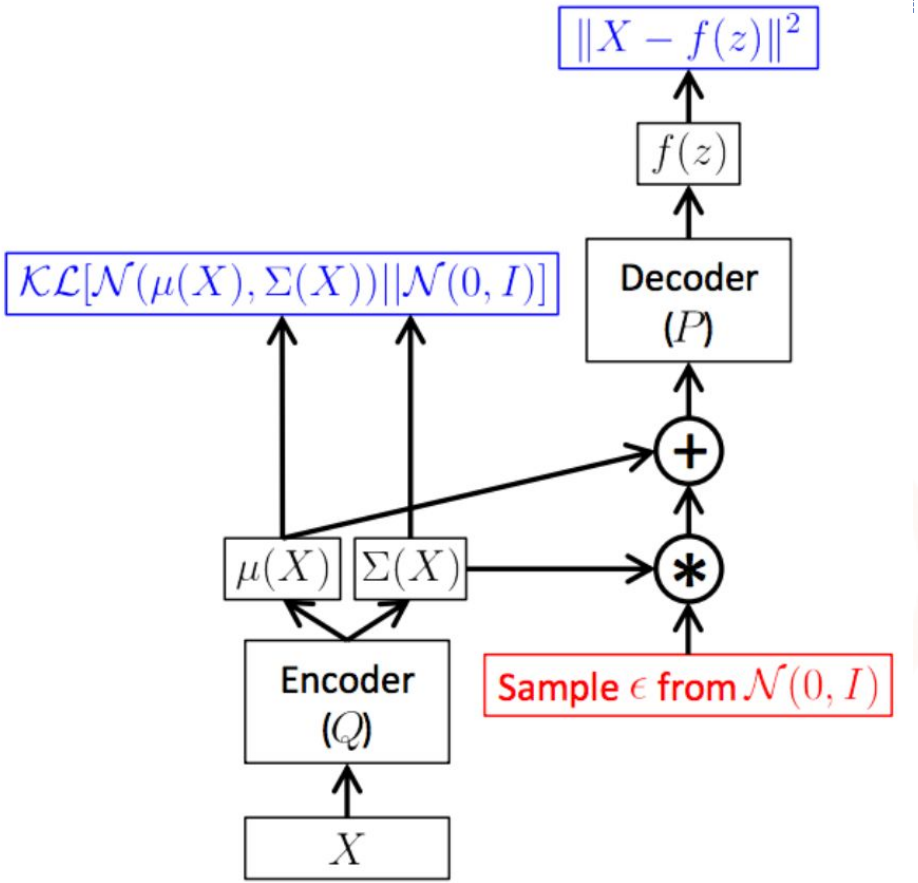




https://gregorygundersen.com/blog/2018/04/29/reparameterization/
https://www.youtube.com/watch?v=rZufA635dq4

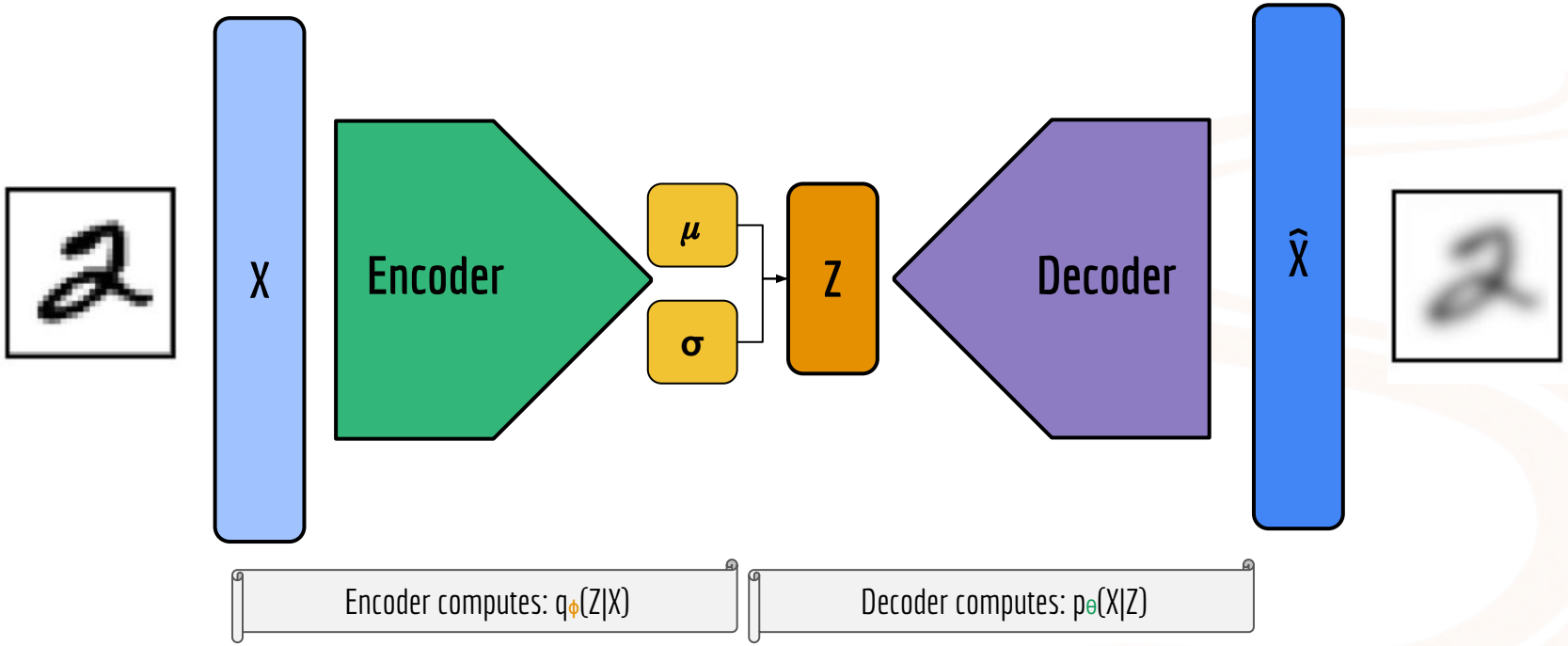
Main structure of a VAE

VAEs rely on the **reparameterization trick** to enable **backpropagation**, and are trained using two types of loss functions: the **reconstruction loss**, which measures how well the output matches the input, and the **KL divergence**, which regularizes the latent space to follow a prior distribution.



https://arxiv.org/abs/1606.05908
https://jeffreyling.github.io/2018/01/09/vaes-are-bayesian.html

VAE optimization



$L(\phi,\theta,x)$ = (reconstruction loss) + (regularization term)

Based on MIT 6.S191 Introduction to Deep Learning (http://introtodeeplearning.com/)

Application of Variational Autoencoder to Telecom

Generative Neural Network Channel Modeling for Millimeter-Wave UAV Communication

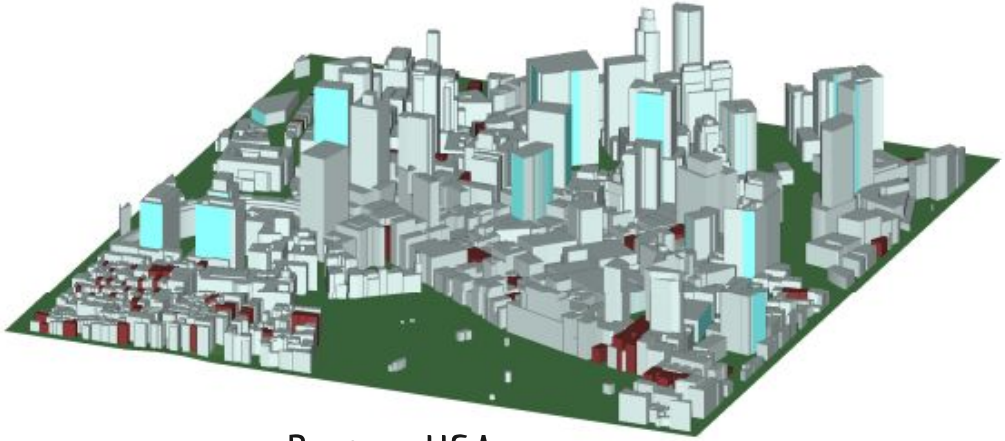
Willian Xia, Sundeep Rangan, Marco Mezzavilla, Angel Lozano, Giovanni Geraci, Vasilii Semkin and Giuseppe Loianno

Link to paper: https://ieeexplore.ieee.org/document/9782737, 2022

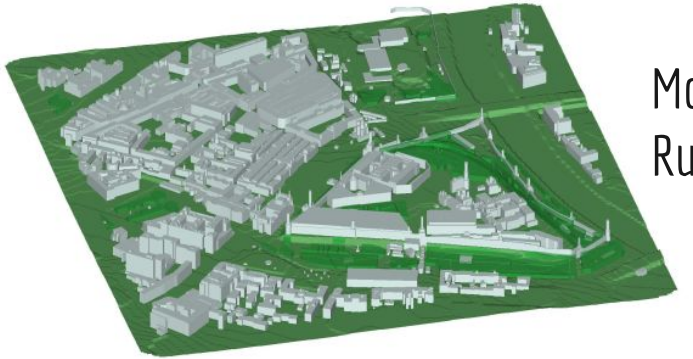
Recall: ray-tracing (RT) allows accurate and site-specific models

Issue: high computational cost for running RT each time a mobile object (transceiver or scatterer) moves

London, UK



Boston, USA

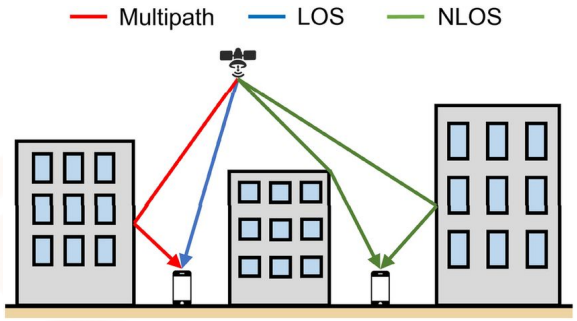


Moscow, Russia

Proposed VAE to generate NLOS air-to-ground channels

Authors adopt K=20 multipath components (MPCs) per channel ⇒ 120 parameters per link:

x = concatenate 20 $\left(L_k, \phi_k^{\mathrm{rx}}, \theta_k^{\mathrm{rx}}, \phi_k^{\mathrm{tx}}, \theta_k^{\mathrm{tx}}, \tau_k\right)$



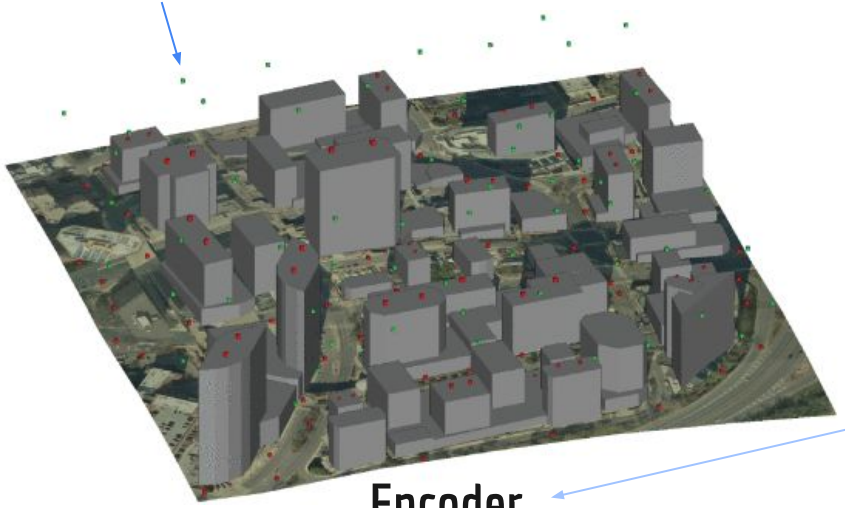
Input is the "link condition" **u**: UAV localization and base station type $\mathbf{u} = [\mathbf{d}, c]$

Want to model the distribution of paths in a link as a function of the link's condition $p(\mathbf{x}|\mathbf{u})$

Adopt VAE generative model, where z is the latent vector $\mathbf{x} = g(\mathbf{u}, \mathbf{z})$

Generative Neural Network Channel Modeling

Dots are the flying drones



	Path VAE encoder	Path VAE decoder
Number of inputs	5 + 120	5 + 20
Hidden units	[200, 80]	[80, 200]
Number of outputs	20 + 20	120 + 120
Number of NN parameters	44520	40720

Encoder

Dimensions: (5, 120)

$$[\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2] = h_{\text{NLOS}}(\mathbf{v}_{\text{path}}, \mathbf{y}_{\text{NLOS}})$$

$$\mathbf{z}_{\text{NLOS}} = \boldsymbol{\mu}_z + \boldsymbol{\sigma}_z \odot \boldsymbol{\epsilon}$$

$\odot$ represents elementwise multiplication

Decoder (generator)

Dimensions: (5, 20)

$$[\boldsymbol{\mu}_y, \boldsymbol{\sigma}_y^2] = g_{\text{NLOS}}(\mathbf{v}_{\text{path}}, \mathbf{z}_{\text{NLOS}})$$

i.i.d. normalized Gaussians

$$\mathbf{y}_{\text{NLOS}} = \boldsymbol{\mu}_y + \boldsymbol{\sigma}_y \odot \mathbf{z}_{\text{out}}$$

Example of results

Average mean absolute error (MAE) of the path loss

	Tokyo, Japan	Beijing, China	London, UK
Default 3GPP Model	0.272	0.247	0.303
Refitted 3GPP Model	0.040	0.056	0.057
Proposed Generative Model	0.036	0.058	0.057

Conclusion: channels generated by the proposed VAE can be integrated into standard 3GPP evaluation methodologies

[1] Standard 3GPP evaluation: Study on Enhanced LTE Support for Aerial Vehicles (Release 15), document TS 36.777
[2] Study on Channel Model for Frequencies From 0.5 to 100 GHz (Release 16), document 38.901, 3GPP, Technical Report, Dec. 2019.

Jupyter notebook implementation presented by Felipe Bastos

Available at: https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

Folder: variational_autoencoders

Diffusion Models

Diffusion models intuition

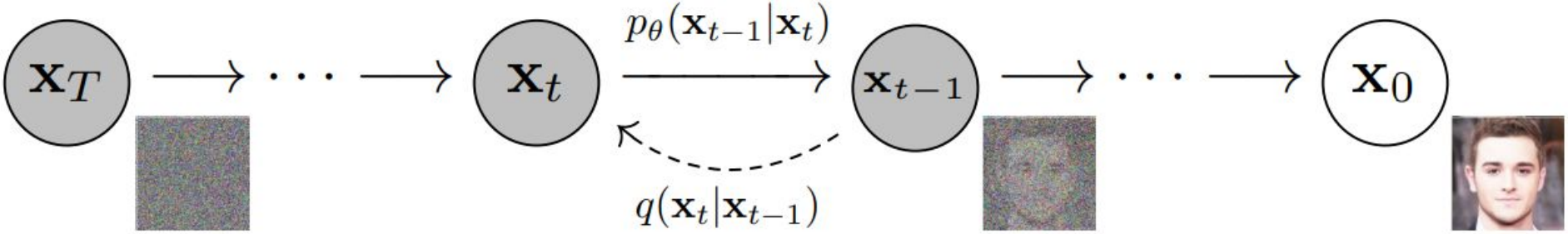


Figure 2: The directed graphical model considered in this work.

Diffusion models are used to learn a data distribution by a process of cleaning noise added to images, then being able to generate data from inputs that are simply noise

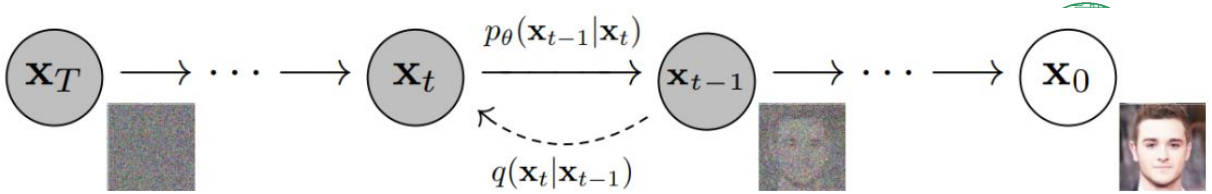
HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. **Denoising diffusion probabilistic models**, 2020.
https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html

Basic facts about the diffusion model

- The **forward** (inference) process is restricted to a simple functional form (e.g. **Gaussian**), in such a way that the **reverse** (**generative**) process will have the same functional form
- During learning only the **mean and covariance for a Gaussian** diffusion kernel need be estimated
- The task of estimating a probability distribution is reduced to the task of performing **regression** on the functions which set the mean and covariance of a sequence of Gaussians
- The learned distributions can be multiplied by any second distribution to **compute a posterior** when denoising an image or estimating a wireless channel

[1] Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015 - https://arxiv.org/abs/1503.03585
[2] https://github.com/Sohl-Dickstein/Diffusion-Probabilistic-Models

Diffusion model math



Well behaved distribution	$\pi\left(\mathbf{x}^{(T)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(T)}; \mathbf{0}, \mathbf{I}\right)$
Forward diffusion kernel	$q\left(\mathbf{x}^{(t)}\|\mathbf{x}^{(t-1)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(t)}; \mathbf{x}^{(t-1)}\sqrt{1-\beta_t}, \mathbf{I}\beta_t\right)$
Reverse diffusion kernel	$p\left(\mathbf{x}^{(t-1)}\|\mathbf{x}^{(t)}\right) =$	$\mathcal{N}\left(\mathbf{x}^{(t-1)}; \mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right), \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right)\right)$
Training targets		$\mathbf{f}_\mu\left(\mathbf{x}^{(t)}, t\right), \mathbf{f}_\Sigma\left(\mathbf{x}^{(t)}, t\right), \beta_{1\ldots T}$
Forward distribution	$q\left(\mathbf{x}^{(0\cdots T)}\right) =$	$q\left(\mathbf{x}^{(0)}\right)\prod_{t=1}^{T} q\left(\mathbf{x}^{(t)}\|\mathbf{x}^{(t-1)}\right)$
Reverse distribution	$p\left(\mathbf{x}^{(0\cdots T)}\right) =$	$\pi\left(\mathbf{x}^{(T)}\right)\prod_{t=1}^{T} p\left(\mathbf{x}^{(t-1)}\|\mathbf{x}^{(t)}\right)$
Log likelihood	$L =$	$\int d\mathbf{x}^{(0)} q\left(\mathbf{x}^{(0)}\right)\log p\left(\mathbf{x}^{(0)}\right)$

[1] Deep Unsupervised Learning using Nonequilibrium Thermodynamics, 2015 - https://arxiv.org/abs/1503.03585

Application of Diffusion Models to Telecom

Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation

Benedikt Fesl, Michael Baur, Florian Strasser, Michael Joham, Wolfgang Utschick

Link to paper: https://arxiv.org/abs/2403.03545, 2024

Extra!

Diffusion models for channel estimation

Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation

LASSE

This capacity of generating new data based on a previous learned distribution is valuable, for instance, for **CHANNEL ESTIMATION**

Currently, channel estimation methods are using generative models such as

- Gaussian Mixture Models (**GMMs**)
- Mixture of Factor Analyzers (**MFAs**)
- Generative Adversarial Networks (**GANs**)
- Variational Autoencoders (**VAEs**)

More recently, Diffuse Models (**DMs**), recognized among the most powerful generative models, are also being used for wireless communications in applications such as channel coding. These however suffer from a large computational overhead, hindering their use in real-time applications, e.g. channel estimation

FESL, Benedikt et al. Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation, 2024. https://arxiv.org/abs/2403.03545

Diffusion models for channel estimation

Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation

Diffuse Models (**DMs**), recognized among the most powerful generative models, are also being used for wireless communications. DMs however suffer from a large computational overhead, hindering their use in real-time applications, e.g. channel estimation

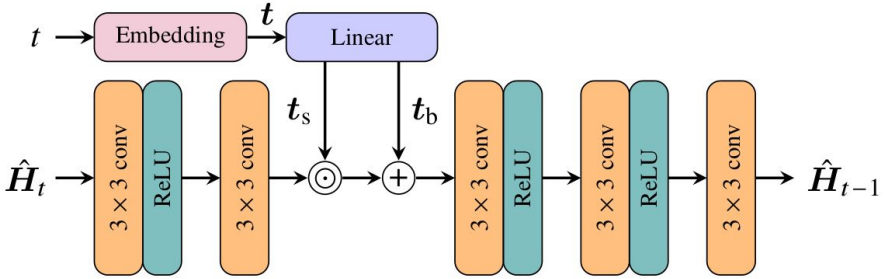


Fig. 1. DM architecture with a lightweight CNN and positional embedding of the SNR information. The NN parameters are shared across all timesteps.

To address this complexity issue, instead of the original sophisticated NN architecture used in the Diffuse Model approach, the authors used a lightweight Convolutional Neural Network (CNN), therefore reducing the number of parameters in the model

The above lightweight CNN role is to learn the channel distribution in the angular domain, which shows a sparse structure

FESL, Benedikt et al. **Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation**, 2024. https://arxiv.org/abs/2403.03545

Diffusion models for channel estimation

Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation

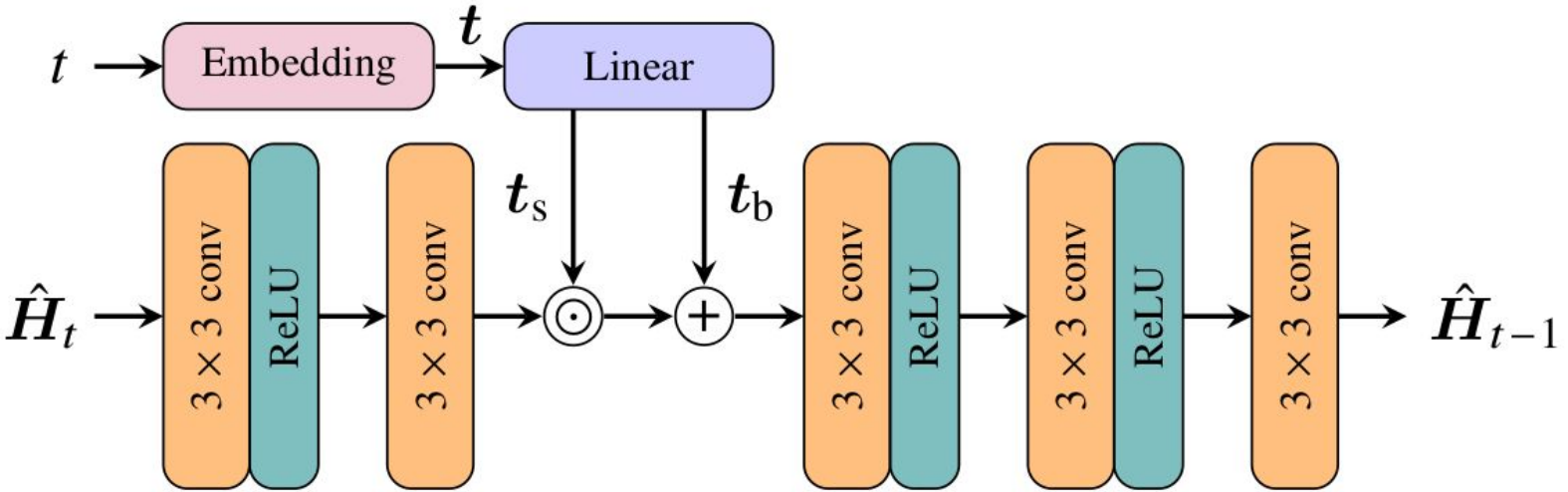


Fig. 1. DM architecture with a lightweight CNN and positional embedding of the SNR information. The NN parameters are shared across all timesteps.

FESL, Benedikt et al. **Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation**, 2024.
https://arxiv.org/abs/2403.03545

Diffusion models for channel estimation

Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation

When it comes to Mean Squared Error (MSE) in channel estimation, the optimal method is using the conditional mean estimator (CME) [TSE, 2005, Appendix A.3]

(TSE, 2005) TSE, David; VISWANATH, Pramod. Fundamentals of wireless communication. Cambridge university press, 2005

$$\mathbb{E}[\boldsymbol{h}|\boldsymbol{y}] = \int \boldsymbol{h} p(\boldsymbol{h}|\boldsymbol{y}) \mathrm{d}\boldsymbol{h} = \int \boldsymbol{h} \frac{p(\boldsymbol{y}|\boldsymbol{h}) p(\boldsymbol{h})}{p(\boldsymbol{y})} \mathrm{d}\boldsymbol{h}$$

However, this method has two downsides, the first being its high nonlinearity, the second being its dependence on the knowledge of the prior distribution $p(\boldsymbol{h})$ to be able to be computed

So the authors use the trained diffusion model to generate this prior $p(\boldsymbol{h})$

FESL, Benedikt et al. **Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation**, 2024.
https://arxiv.org/abs/2403.03545

Extra!

Diffusion models for channel estimation

Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation

A - Offline diffusion model training phase

The authors use FFT to transform the channels in their dataset to the angular domain, which is sparse in massive MIMO. This results in having less parameters to be learned, and by consequence, a more stable and fast training process

Steps:

1 - Take the training dataset composed of M_{train} channel matrices and use FFT to transform it to the angular domain

2 - Use the dataset, now in the angular domain, for training the diffusion model to generate the channel estimate $\hat{H}$

Algorithm 1 Channel estimation via a DM as generative prior.

Offline DM Training Phase

Require: Training dataset $\mathcal{H} = \{\boldsymbol{H}_m\}_{m=1}^{M_{\text{train}}}$

1: Transform into angular domain $\tilde{\mathcal{H}} = \{\text{fft}(\boldsymbol{H}_m)\}_{m=1}^{M_{\text{train}}}$

2: Train DM $\{f_{\boldsymbol{\theta},t}^{(T)}\}_{t=1}^{T}$ with $\tilde{\mathcal{H}}$, cf. [5]

FESL, Benedikt et al. **Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation**, 2024.
https://arxiv.org/abs/2403.03545

Diffusion models for channel estimation

Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation

B - Online diffusion model-based channel estimation phase

Steps:

$$\hat{\boldsymbol{H}}_{\mathrm{LS}} = \boldsymbol{Y}\boldsymbol{P}^{\mathrm{H}} = \boldsymbol{H} + \tilde{\boldsymbol{N}}$$

$\boldsymbol{N}$ is additive white Gaussian noise
$\boldsymbol{P} \in \mathbb{C}^{N_{\mathrm{tx}} \times N_{\mathrm{p}}}$ is the pilot matrix

where $\tilde{\boldsymbol{N}} = \boldsymbol{N}\boldsymbol{P}^{\mathrm{H}}$ is AWGN with variance η^2 due to $\boldsymbol{P}$ being unitary.

3 - The first step is to decorrelate the pilot matrix. This is done by computing the least squares solution shown in the right

4 - Then to normalize the observation variance

$$\hat{\boldsymbol{H}}_{\mathrm{init}} = (1+\eta^2)^{-\frac{1}{2}}\hat{\boldsymbol{H}}_{\mathrm{LS}}$$

5 - Transform the channel data to angular domain

$$\hat{\boldsymbol{H}}_{\mathrm{ang}} = \mathrm{fft}(\hat{\boldsymbol{H}}_{\mathrm{init}})$$

6 - Estimate the DM timestep

$$\hat{t} = \arg\min_{t} |\mathrm{SNR}(\boldsymbol{Y}) - \mathrm{SNR}_{\mathrm{DM}}(t)|$$

7 - Initialize the DM reverse process

$$\hat{\boldsymbol{H}}_{\hat{t}} = \hat{\boldsymbol{H}}_{\mathrm{ang}}$$

8 - Perform the stepwise iteration to estimate $\hat{\boldsymbol{H}}$ when $t = 0$

for $t = \hat{t}$ **down to** 1 **do**
$\hat{\boldsymbol{H}}_{t-1} \leftarrow f_{\boldsymbol{\theta},t}^{(T)}(\hat{\boldsymbol{H}}_t)$
end for

9 - Transform back to spatial domain

$$\hat{\boldsymbol{H}} \leftarrow \mathrm{ifft}(\hat{\boldsymbol{H}}_0)$$

FESL, Benedikt et al. **Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation**, 2024.
https://arxiv.org/abs/2403.03545

Diffusion models for channel estimation

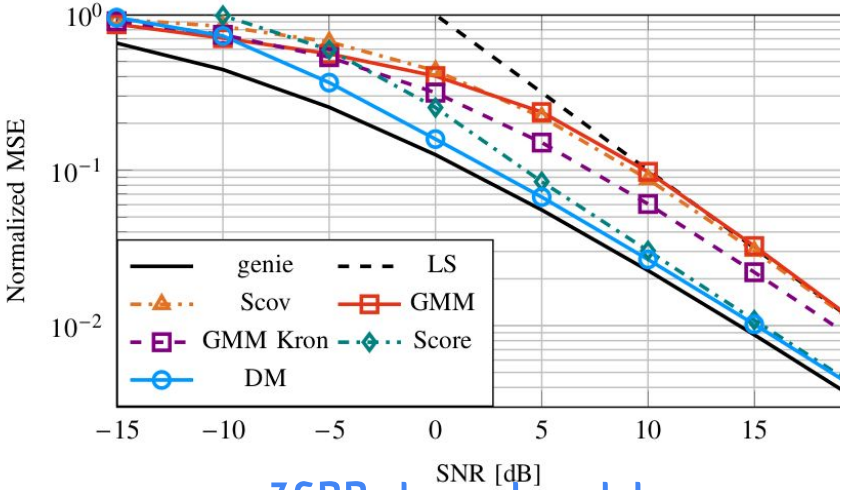
Main results

TABLE I (adapted)

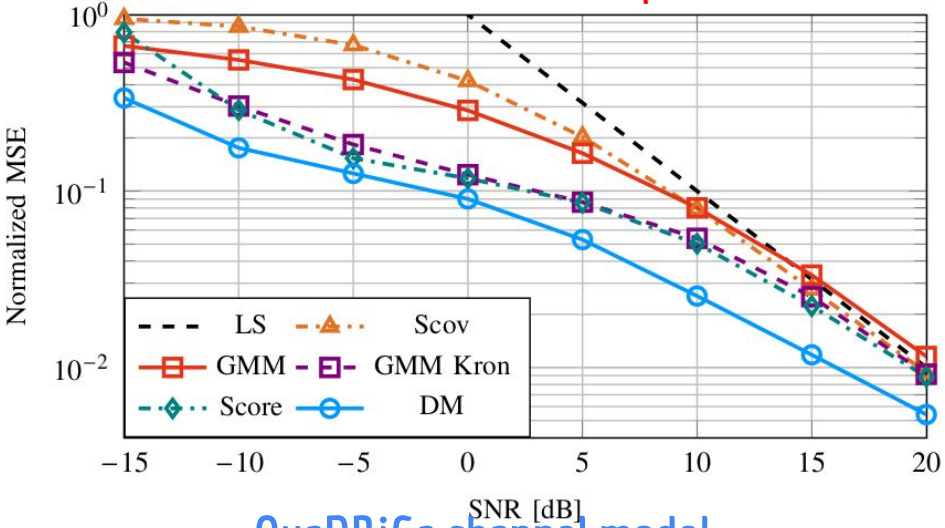
MEMORY ANALYSIS FOR $(N_{RX}, N_{TX}) = (64, 16)$.

Method	Parameters
Scov	$1.05 \cdot 10^6$
GMM	$1.35 \cdot 10^8$
GMM Kron	$7.22 \cdot 10^4$
Score	$5.89 \cdot 10^6$
DM (proposed)	$\mathbf{5.50 \cdot 10^4}$

The one with fewer parameters



3GPP channel model



QuaDRiGa channel model

FESL, Benedikt et al. **Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation**, 2024. https://arxiv.org/abs/2403.03545

Jupyter notebook implementation presented by João Borges

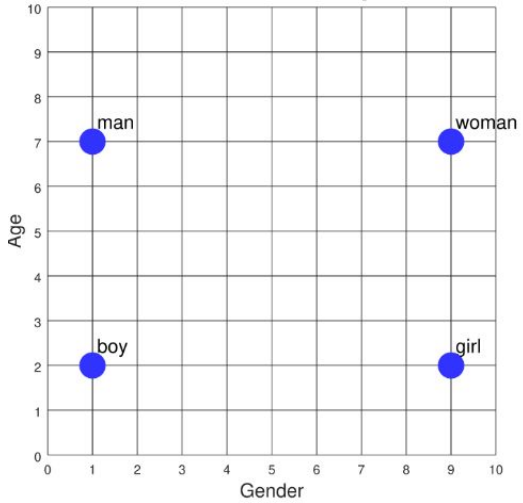
Available at: https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

Folder: Diffusion_channel_est

Part III - Foundations of Transformers and LLMs

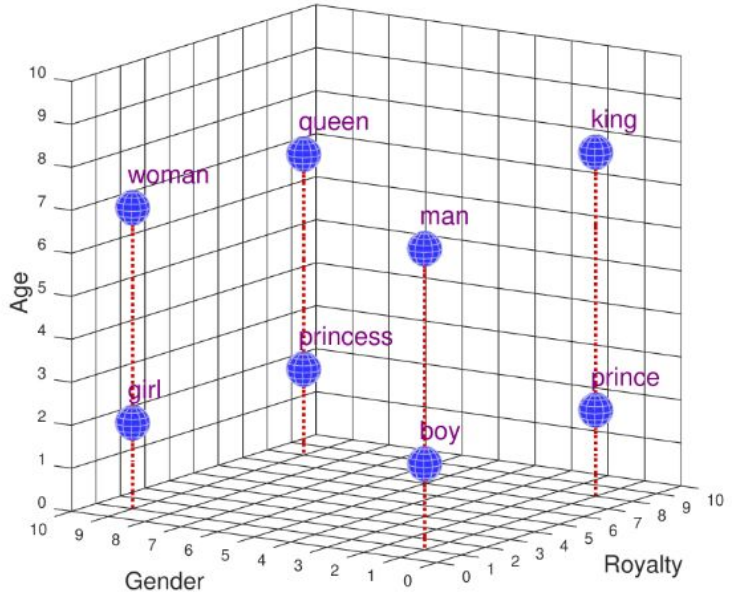
History of language models (not necessarily large) and embeddings

Word embeddings



Word Coordinates	Gender	Age
man	1	7
woman	9	7
boy	1	2
girl	9	2

2D



Word Coordinates	Gender	Age	Royalty
man	1	7	1
woman	9	7	1
boy	1	2	1
girl	9	2	1
king	1	8	8
queen	9	7	8
prince	1	2	8
princess	9	2	8

3D

Embeddings have some similarities to latent spaces but are often deterministic and not “hidden”

https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/tutorial.html
https://www.tensorflow.org/text/guide/word_embeddings

Language models evolution

- N-gram models
- Figure of merit: perplexity

	Unigram	Bigram	Trigram
Perplexity	962	170	109

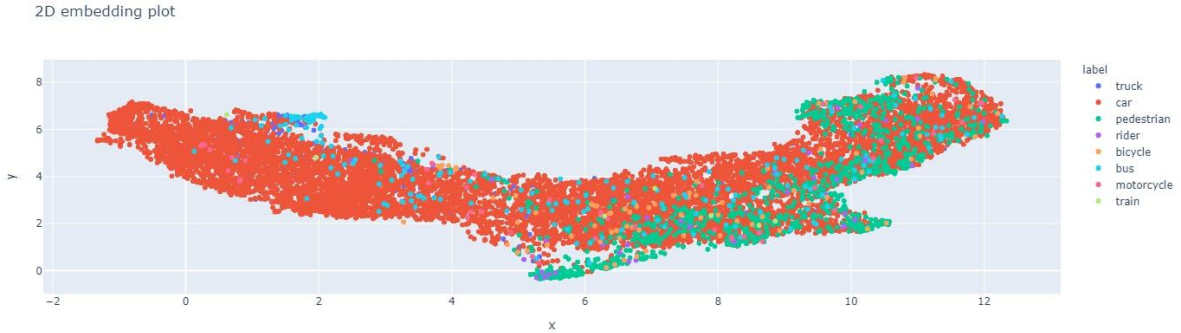
O maior do Norte é o ...

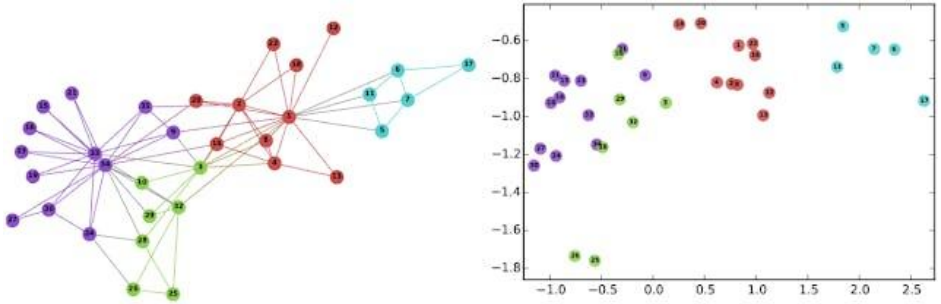
Figure of [1]: 2D t-SNE projection of semantic embedding of dimension 60

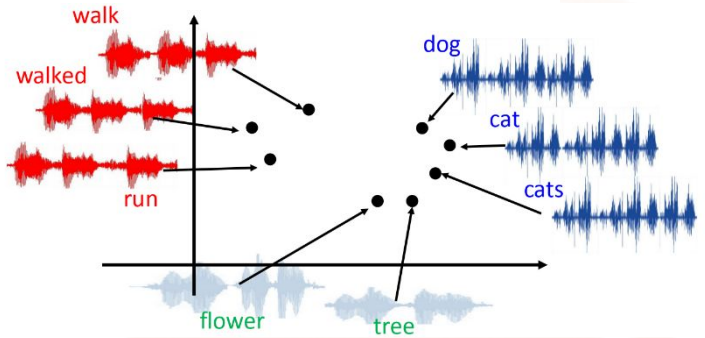
- Static embedding: word2vec
- Contextual embedding: BERT, RoBERTa, SpanBERT

[1] Daniel Jurafsky and James H. Martin. 2024. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3rd edition. https://web.stanford.edu/~jurafsky/slp3
[2] T. Mikolov et al, "Efficient Estimation of Word Representations in Vector Space," 2013 (word2vec)

Embeddings are available for any kind of data (even sequences)





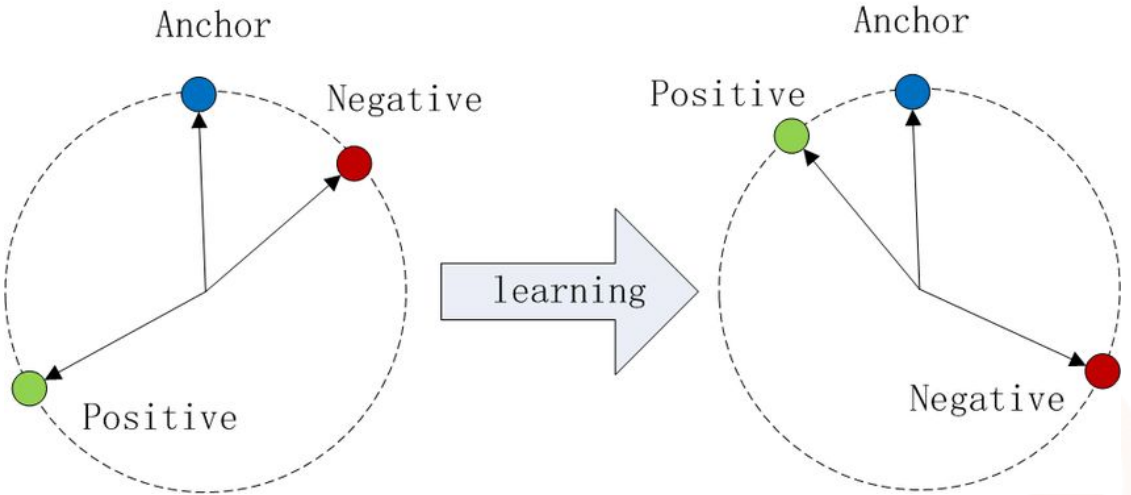


https://encord.com/blog/embeddings-machine-learning/
https://people.csail.mit.edu/weifang/project/spml17-audio2vec/
https://thenewstack.io/how-to-get-the-right-vector-embeddings/

Extra!

How to create embeddings in practice???

One example:
Triplet loss



$$L_{\text{tri}} = [D(f_a, f_p) - D(f_a, f_n) + m]_+$$

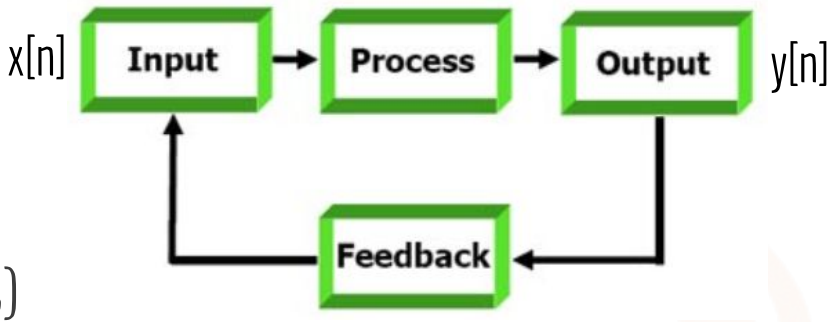
https://www.timescale.com/blog/a-beginners-guide-to-vector-embeddings/
https://huggingface.co/AskYoutube/AskVideos-VideoCLIP-v0.1
https://github.com/tjddus9597/Beyond-Binary-Supervision-CVPR19/tree/master/code

Recurrent neural networks

Extra!

Recurrence from the perspective of digital signal processing

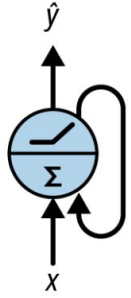
- Non-recurrent difference equation (related to FIR filters)
  - x[n] and y[n] are the input and output at time n
  - Example:
  - y[n] = x[n] + 0.5 x[n-1] + 0.2 x[n-8]



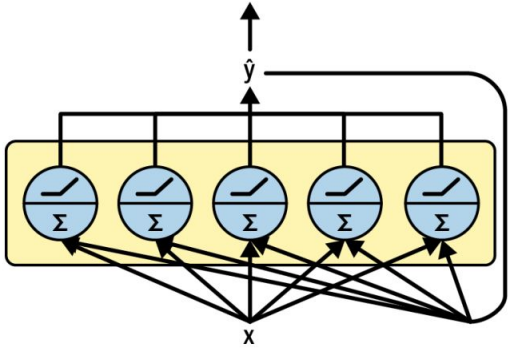
- Recurrent difference equation (related to IIR filters)
  - There is "**feedback**": the output influences the input
  - The input depends on the output y[n] and its delayed versions (e.g. y[n-4])
  - Example:
  - y[n] = x[n] + 0.5 y[n-1] + 0.3 y[n-4]

Recurrent neuron and layer of neurons

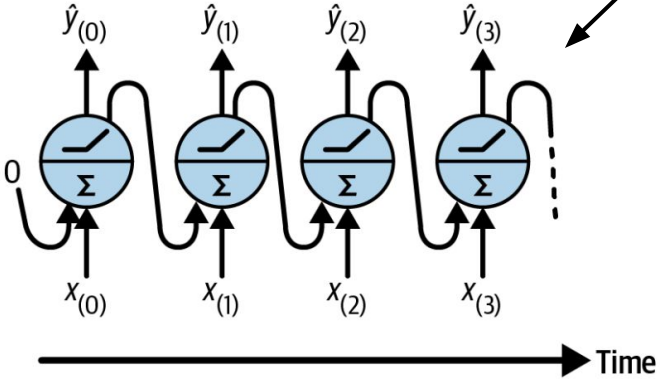
Single neuron



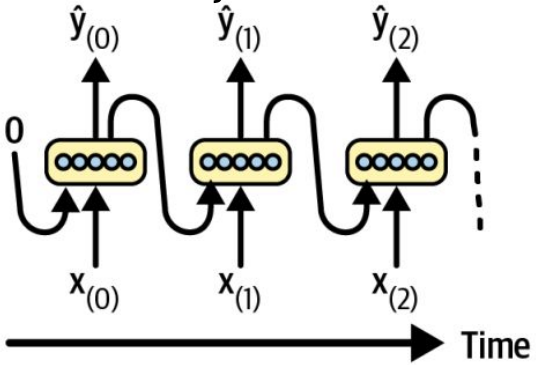
Layer of neurons



Unrolling through time



Multilayer RNN



$$\hat{\mathbf{y}}(t) = f(\mathbf{W}_x\mathbf{x}(t) + \mathbf{W}_y\hat{\mathbf{y}}(t-1) + \mathbf{b})$$

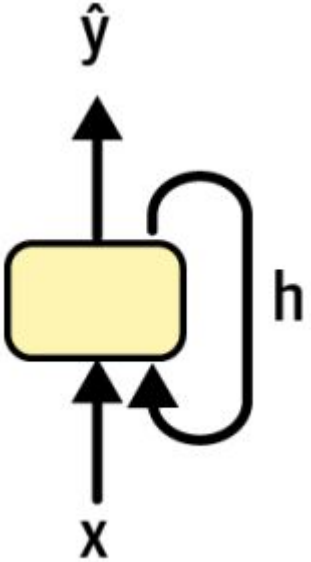
Define number of steps to unroll and train with "backprop over time"

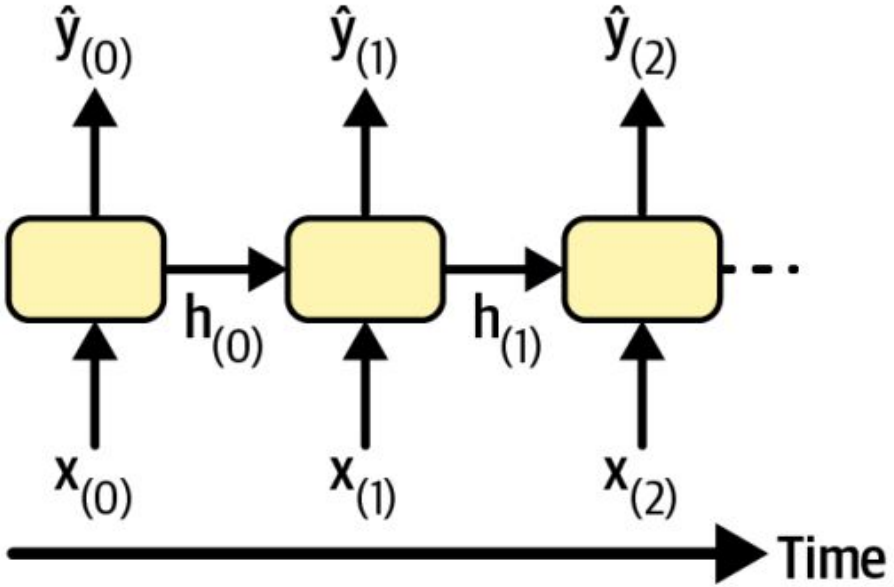
Memory cell: part of NN that preserves some (hidden) state across time steps

Additional degree of freedom: cell's hidden state and its output may be different

$$\mathbf{h}(t) = f(\mathbf{W}_{xh}\mathbf{x}(t) + \mathbf{W}_{hh}\mathbf{h}(t-1) + \mathbf{b}_h)$$

$$\hat{\mathbf{y}}(t) = g(\mathbf{W}_{hy}\mathbf{h}(t) + \mathbf{b}_y)$$



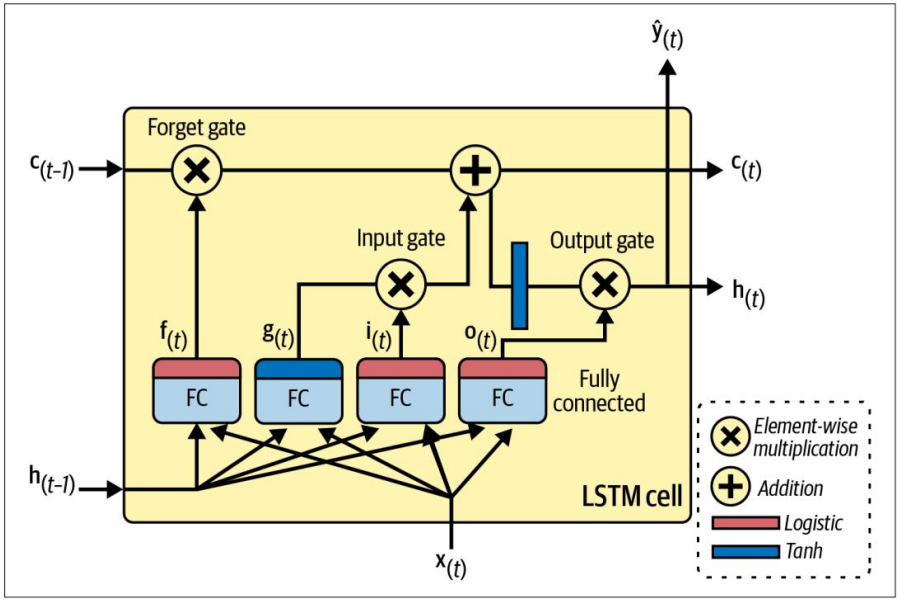


Extra!

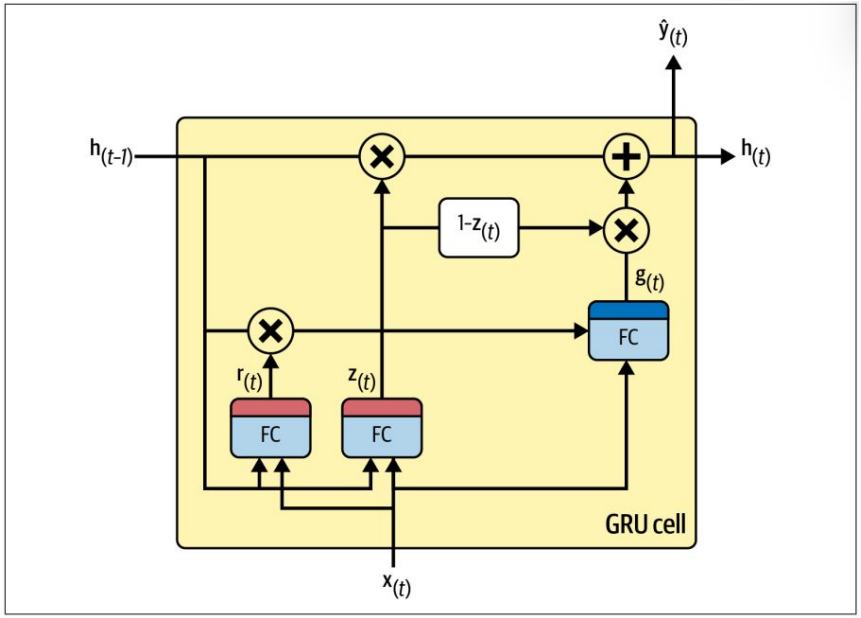
Several RNN memory cells (LSTM, GRU, etc.)

LASSE

Long short-term memory (LSTM), 1997



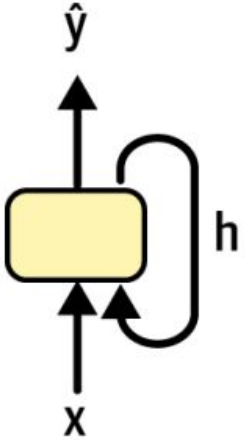
Gated recurrent unit (GRU), 2014



New concept: “gated” cells

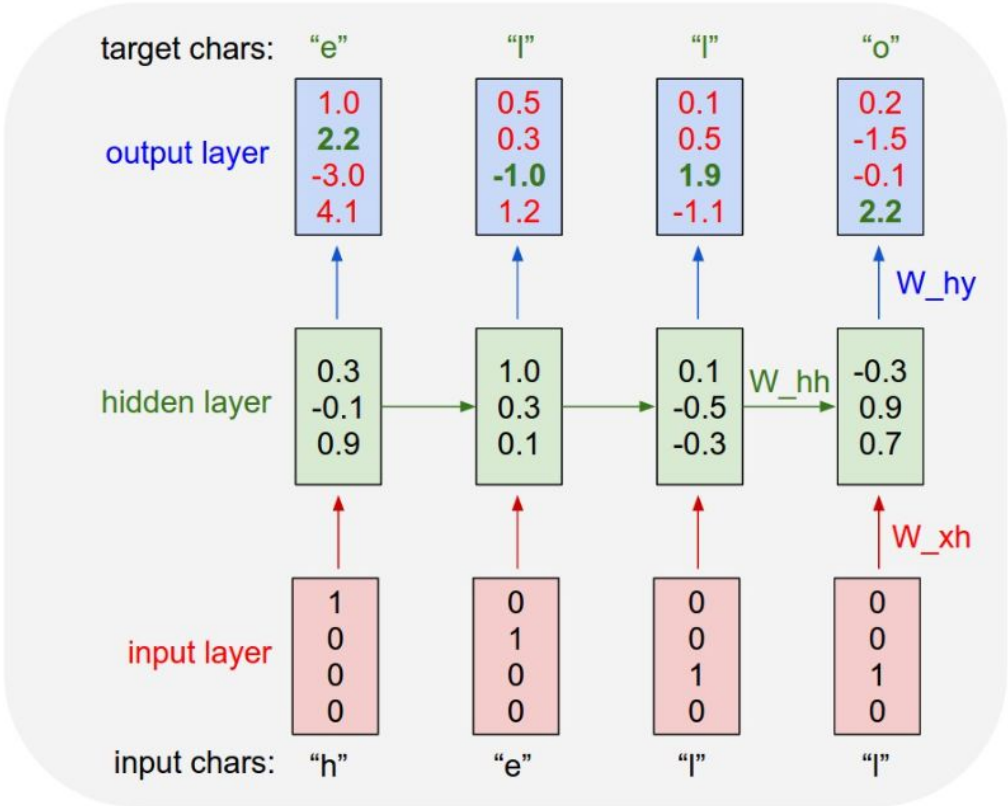
Minimalist recurrent neural network (Karpathy's min-char-rnn.py)

Character-level language model




```
# hyperparameters
hidden_size = 3
seq_length = 4
```

Example with vocabulary 'h', 'e', 'l', 'o'



Compare: When used from ChatGPT, OpenAI o1-preview has a context window of 32,000 tokens!

https://karpathy.github.io/2015/05/21/rnn-effectiveness/
https://mkffl.github.io/2019/07/08/minimalist-RNN.html
https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85

Attention Mechanism

In a sequence-to-sequence task, as long as the sequence is short, everything works fine in the RNN model

If T is too large?

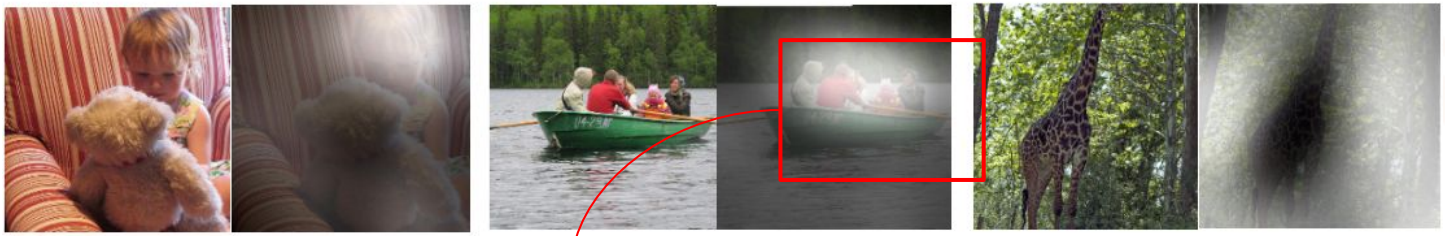
Attention was initially proposed to solve long sequence problems affected by RNN models

Extra!

Attention Mechanism

Intuitive way to identify the use of attention mechanism

The biological visual attention is the ability to dynamically restrict processing to a subset of the visual field. First attempts at adapting attention mechanisms to neural networks go back to the late 1980s [1]



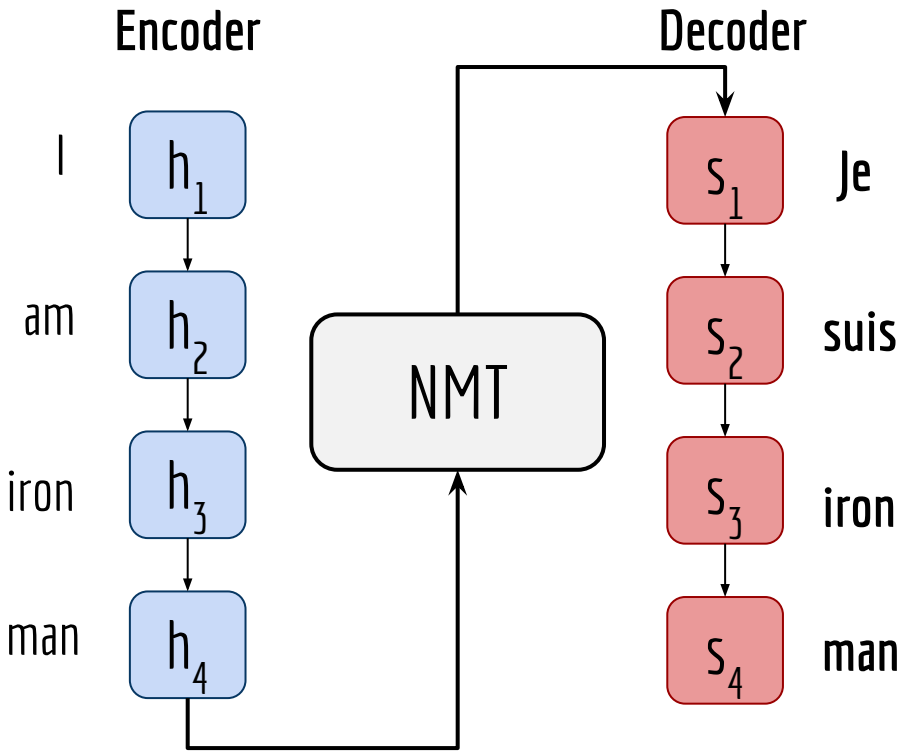
It can be said that 2015 is the golden year of attention mechanisms with novel approaches for encoder-decoder neural machine translation (NMT)

Image-to-text task using attention mechanism to focus in some parts of the image [2], published in 2015, before the transformers paper (2017)

[1] https://arxiv.org/abs/2204.13154

[2] Show, Attend and Tell: Neural Image Caption Generation with Visual Attention (2015, Kelvin Xu, et al.) https://arxiv.org/abs/1502.03044

History of attention mechanism in DL



NMT: neural machine translation

- Compressing all necessary information of a source sentence into a fixed-length vector makes it difficult for the neural network to capture all semantic details of long sentences
- Essence of the attention mechanism: capture a new context vector for every timestep

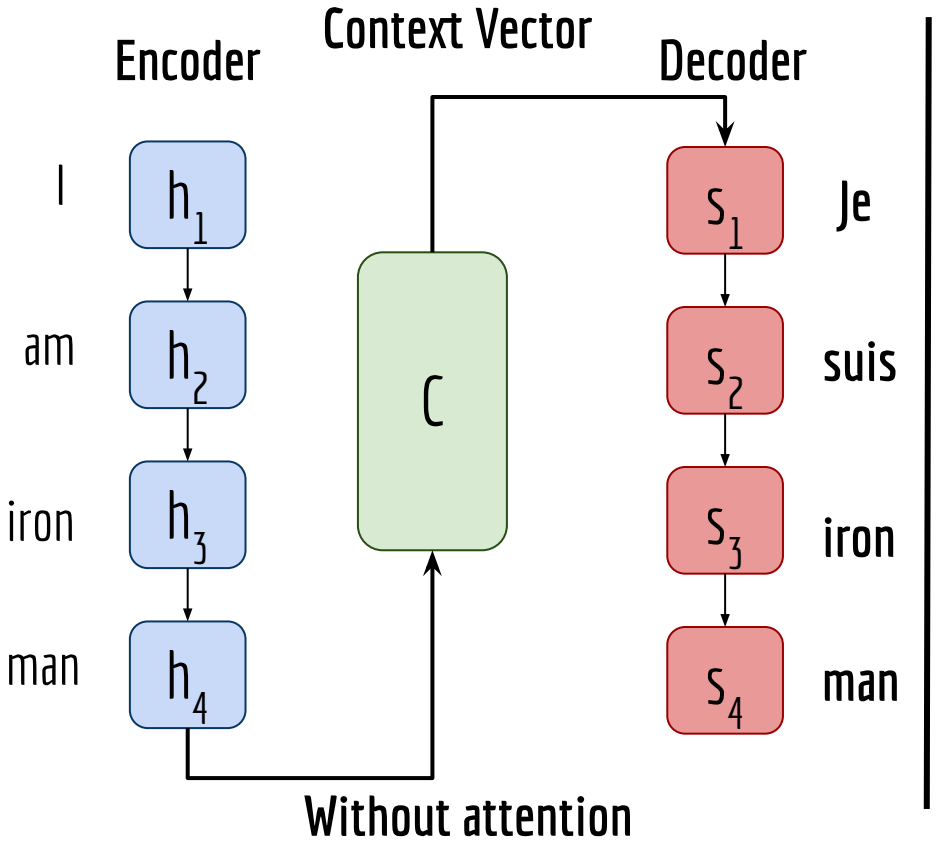
https://arxiv.org/abs/2204.13154

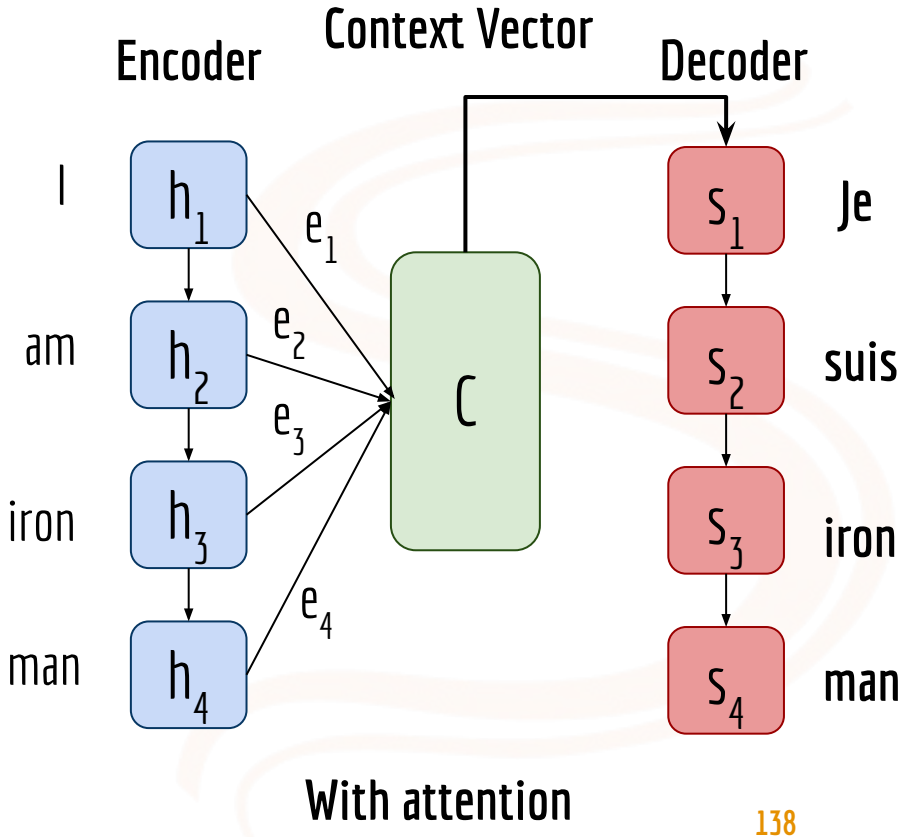
Attention Mechanism

Before the paper "attention is all you need"

Sequence-to-Sequence Problem

e_i: Attention score

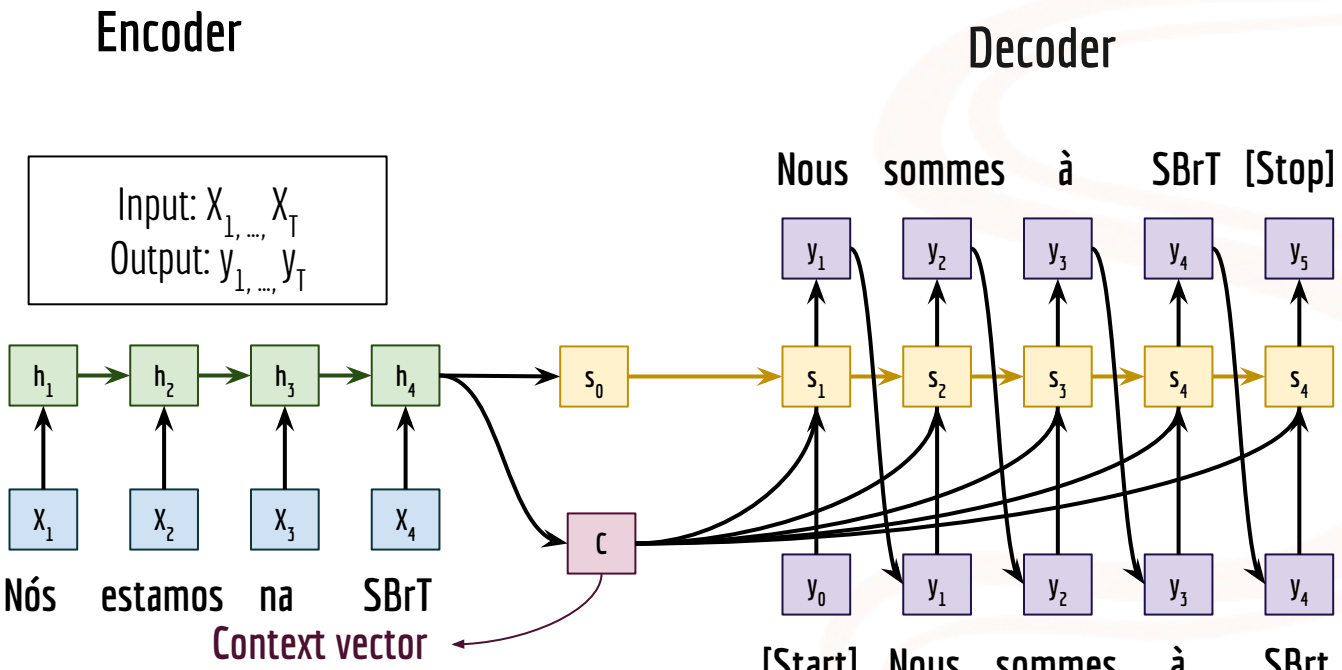




Extra!

Attention Mechanism

Before the paper "attention is all you need"



Neural machine translation by jointly learning to align and translate, Bahdanau et al, 2016

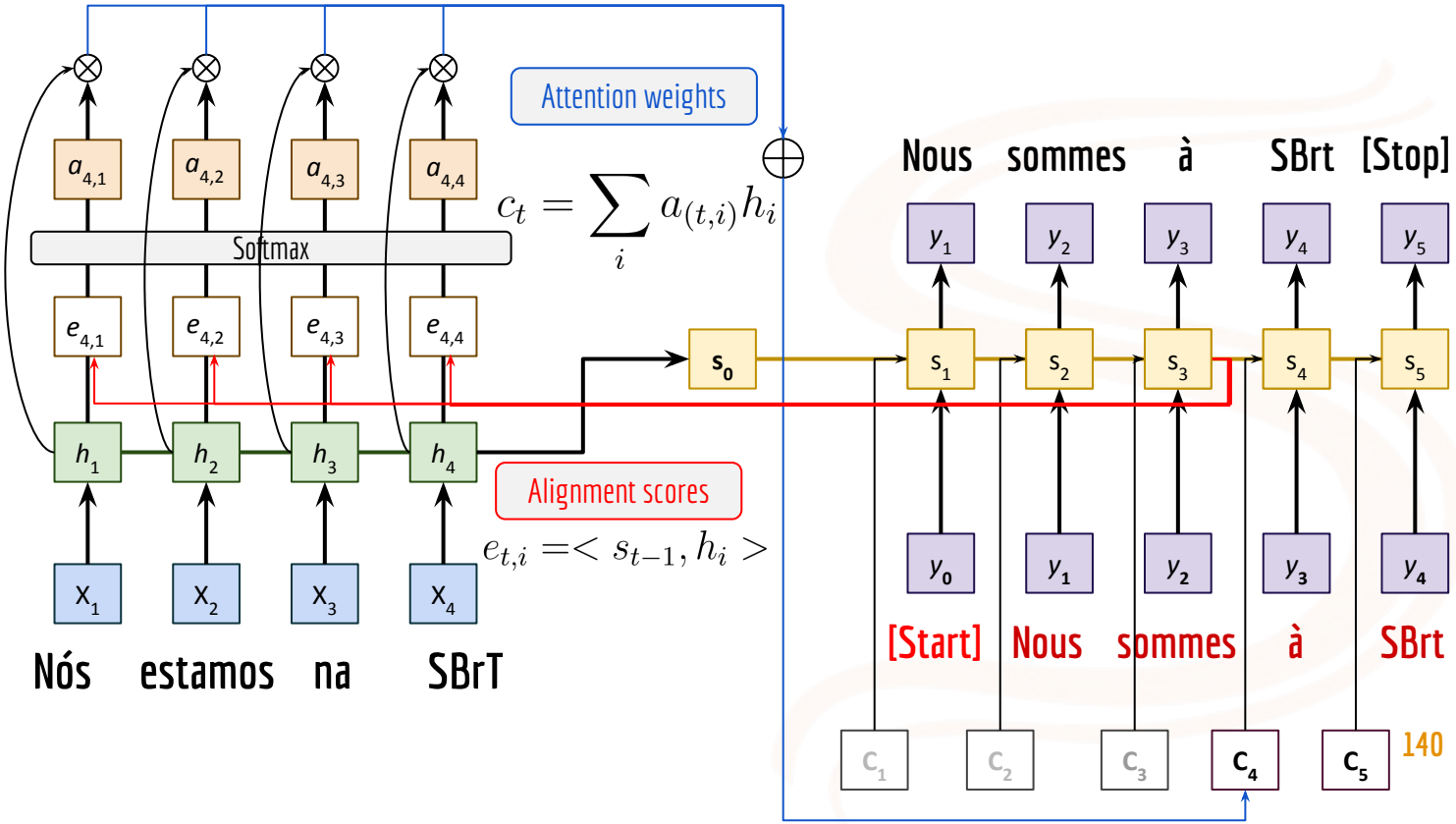
Extra!

Attention Mechanism

Before the paper "attention is all we need"

Attention was initially proposed to solve long sequence problems affected by RNN models

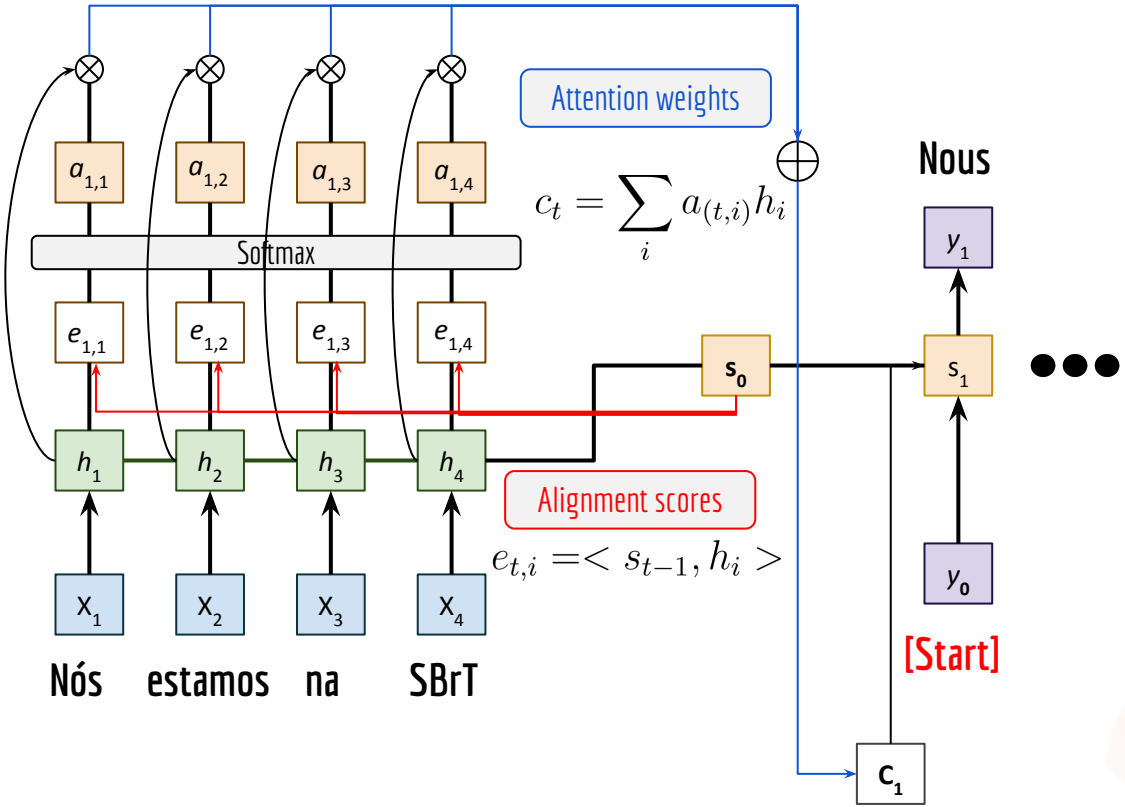
The essence of the attention mechanism is to capture a new context vector for every timestep



Extra!

Attention Mechanism

Let us practice!



Considering

$s_0 = 2$;

$\mathbf{h} = [2, 8, 10, 7]$.

1. Evaluate the alignment score **e**
2. Evaluate the attention weight c_1 in a **hard** attention approach

Answers:

1. **e** = [4, 16, 20, 14]
2. $c_1 = 434$

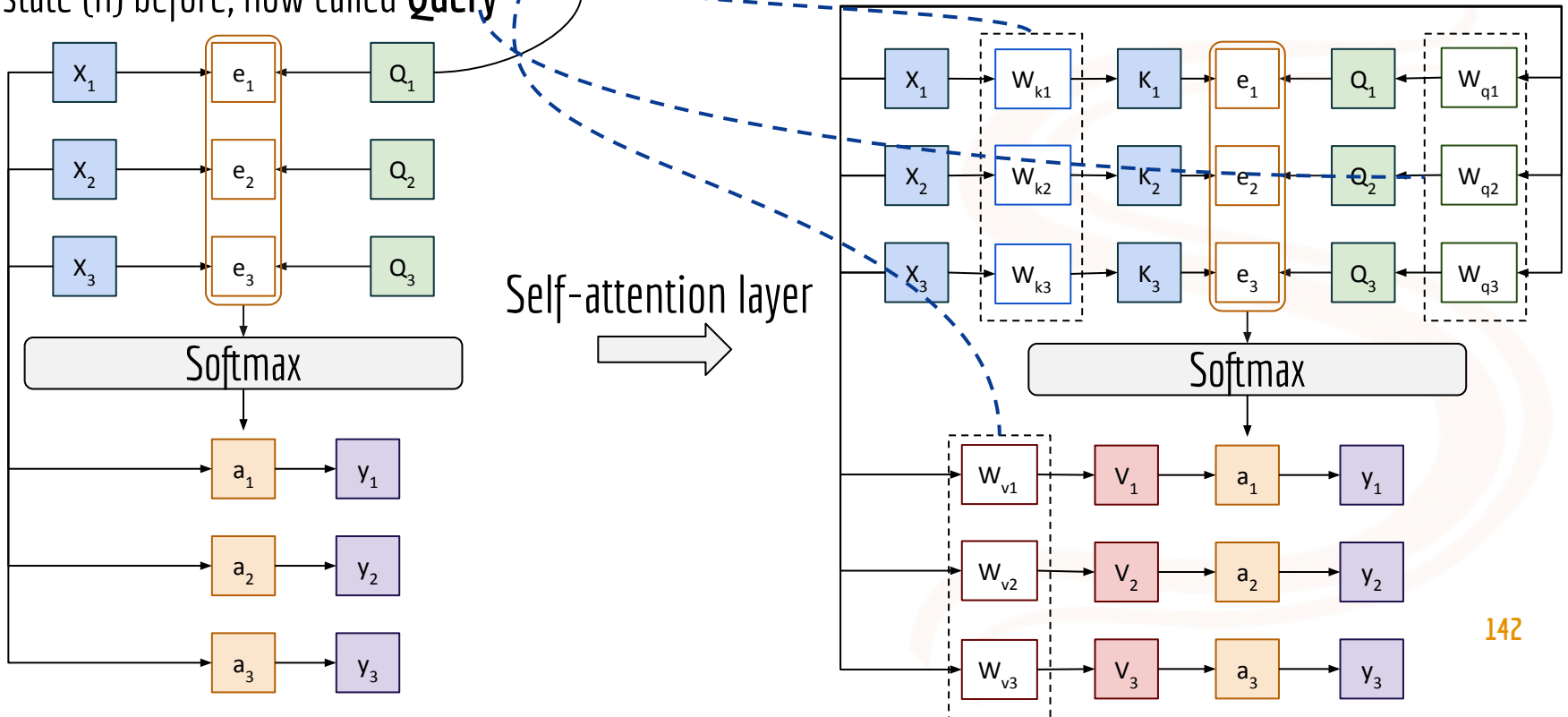
Extra!

Attention Mechanism

General Attention Layer

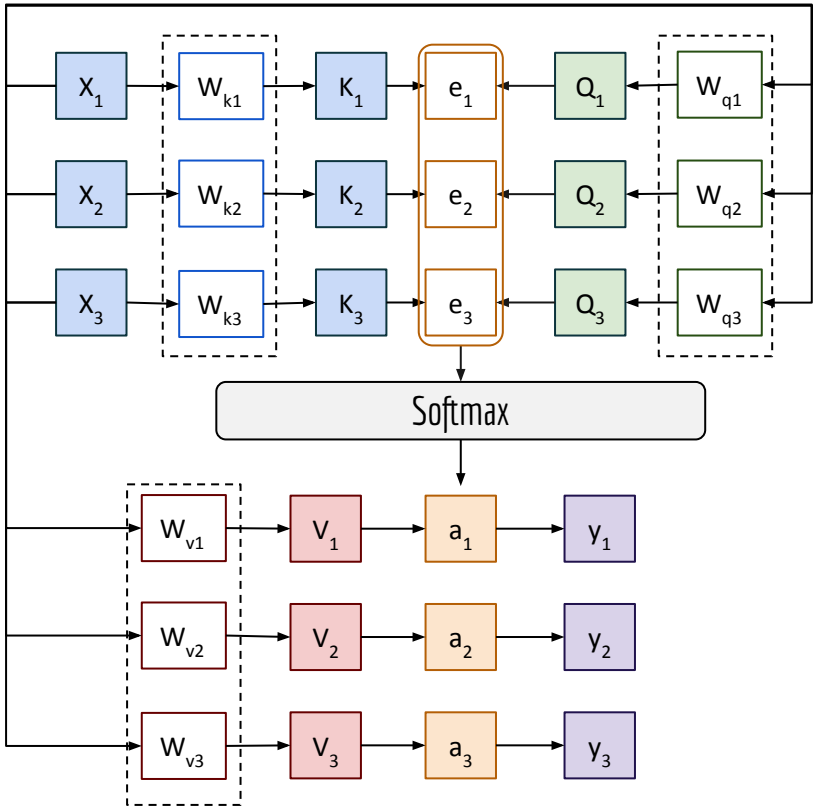
Hidden state (h) before, now called **Query**

Weight matrices to obtain Query (Q), Key (K), Values (V) matrices

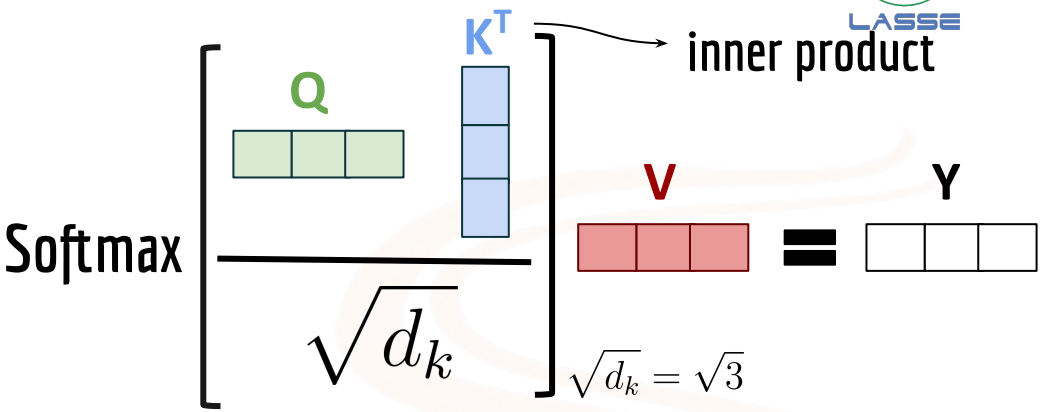


Self-Attention Mechanism

Train 3 weight matrices: Q, K and V



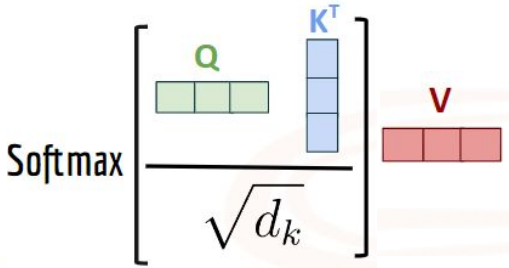
$\sqrt{d_k}$: Normalization factor



The names query (Q), key (K) and value (V) can have interpretations using analogies, e.g., with retrieval systems

In essence: given a sequence (embedding vectors), Q and K use inner products and softmax to find relevant pairs, and the actual values (V) are scaled by these probabilities

Analogy of visual attention and the role of Q,K and V in transformers



A group of people sitting on a boat in the water.

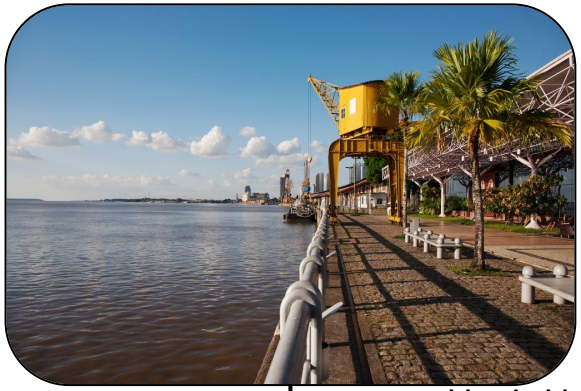
A giraffe standing in a forest with trees in the background.

Show, Attend and Tell: Neural Image Caption Generation with Visual Attention (2015, Kelvin Xu, et al.)

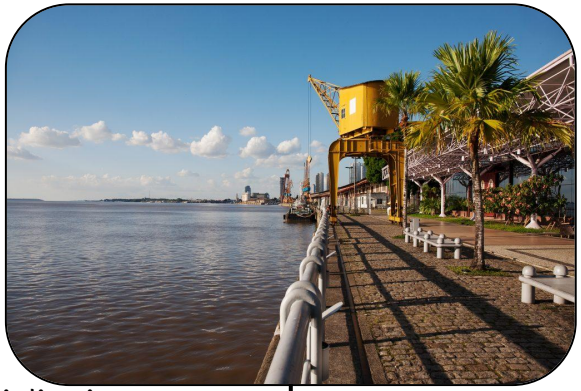
Multi-head Attention

Intuition

https://www.youtube.com/watch?v=mMa2PmYJlCo



Q



K^T

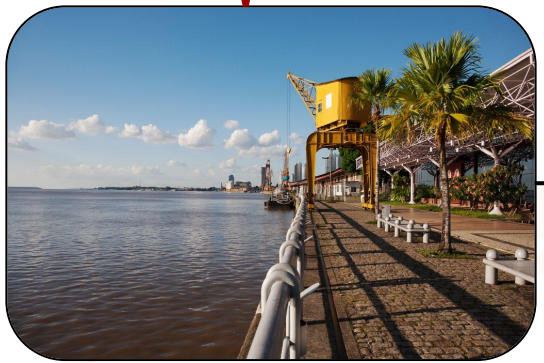
Matrix Multiplication

1

Multi-head Attention

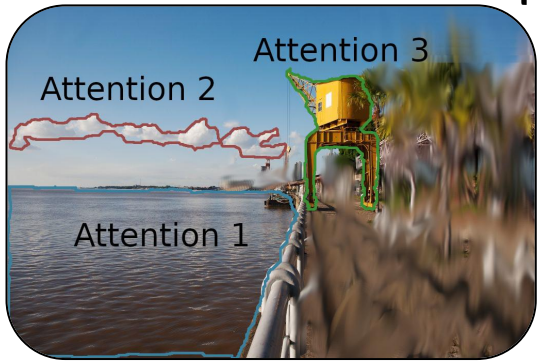
Matrix Multiplication

V



2

Y



1 Obtain attention filters

2 Focus on relevant information

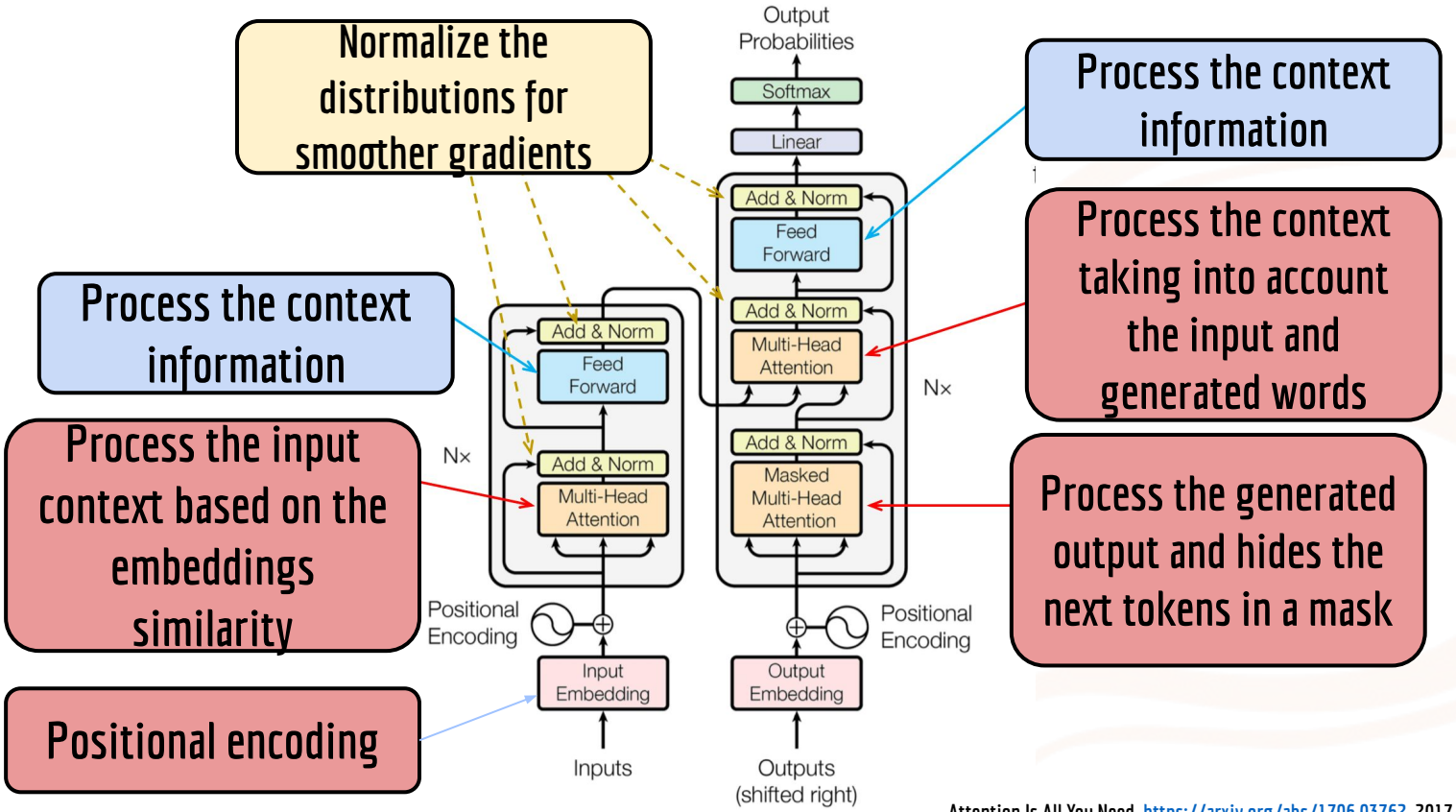
Limitation: RNNs are “sequential”:

$$\hat{\mathbf{y}}(t) = f(\mathbf{W}_x\mathbf{x}(t) + \mathbf{W}_y\hat{\mathbf{y}}(t-1) + \mathbf{b})$$

Transformers

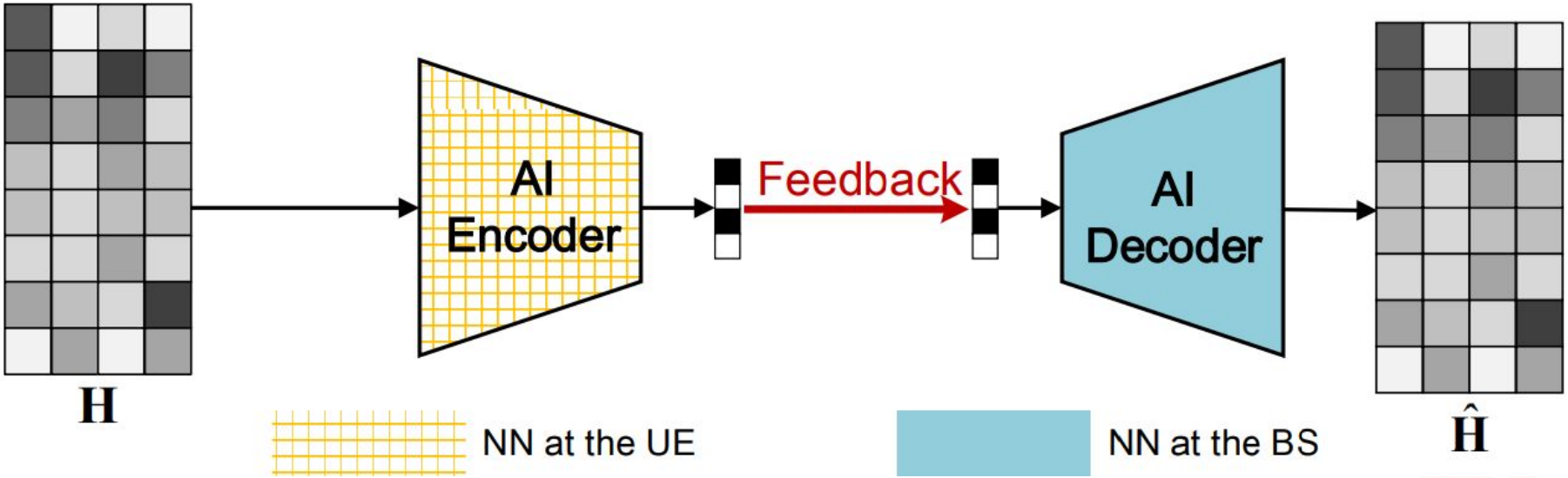
While RNNs are inherently sequential and challenging to parallelize, transformers are designed for parallel execution, making them more efficient for processing large datasets

Transformers and "self-attention"



Attention Is All You Need, https://arxiv.org/abs/1706.03762, 2017

Recall:
3GPP efforts in standardizing AI for CSI feedback in 6G



[1] AI for CSI Feedback Enhancement in 5G-Advanced, J. Guo et al, 2022, https://ieeexplore.ieee.org/document/9970357

Application for Transformers

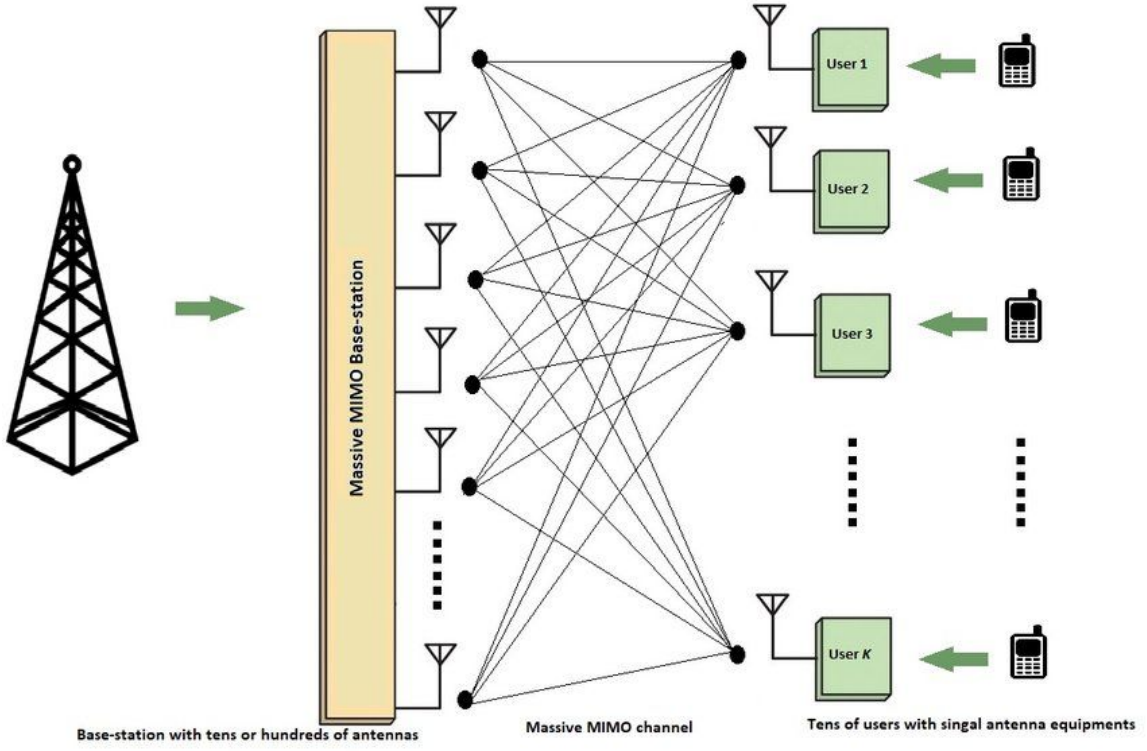
Transformers Empowered CSI Feedback for Massive MIMO Systems

Yang Xu, Mingqi Yuan and Man-On Pun

Link to paper: https://ieeexplore.ieee.org/abstract/document/9602863, 2021

Overview

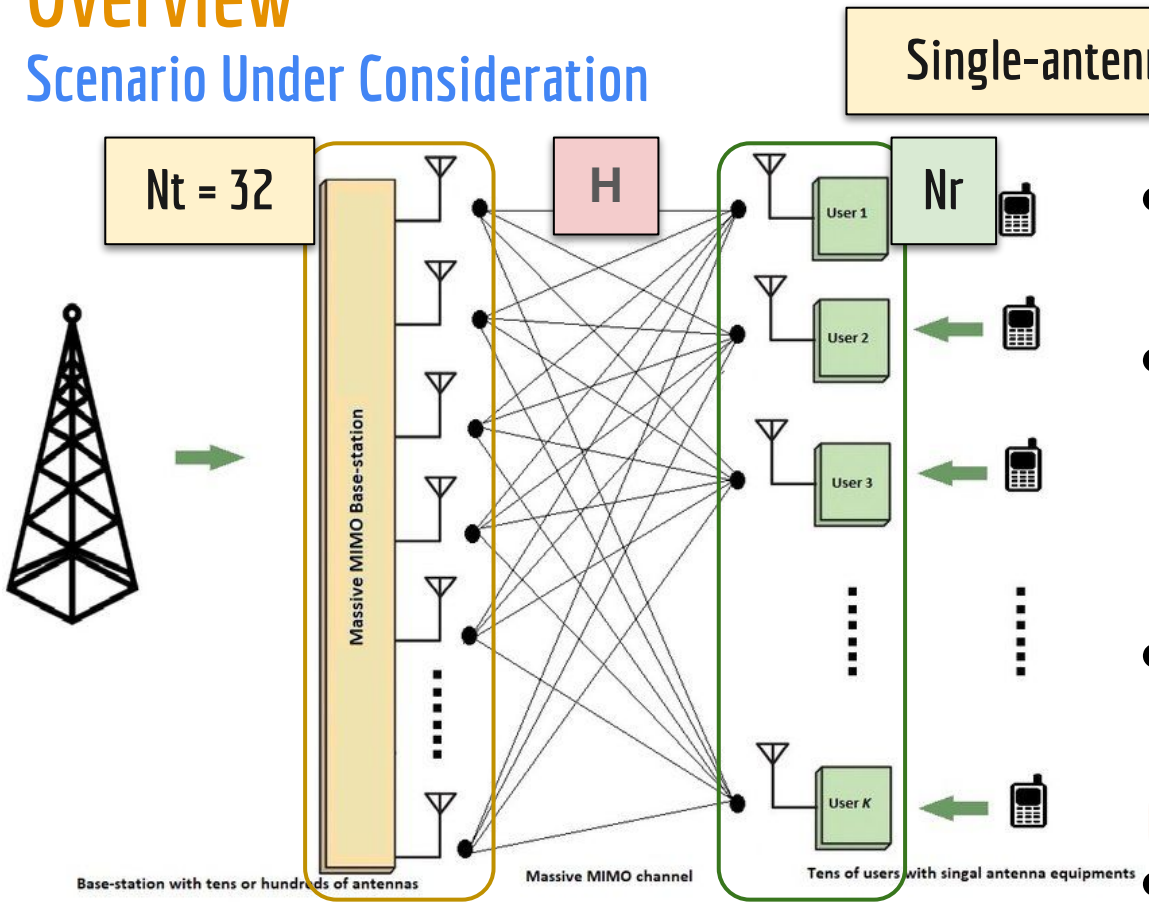
Scenario Under Consideration



- Downlink multi-user massive MIMO;
- Frequency-division duplexing (FDD) ⟶ DL and UL channels are not reciprocal;
- UEs must estimate the DL channels through CSI-RS pilots, then return the estimated channel as “CSI Feedback”.

Overview

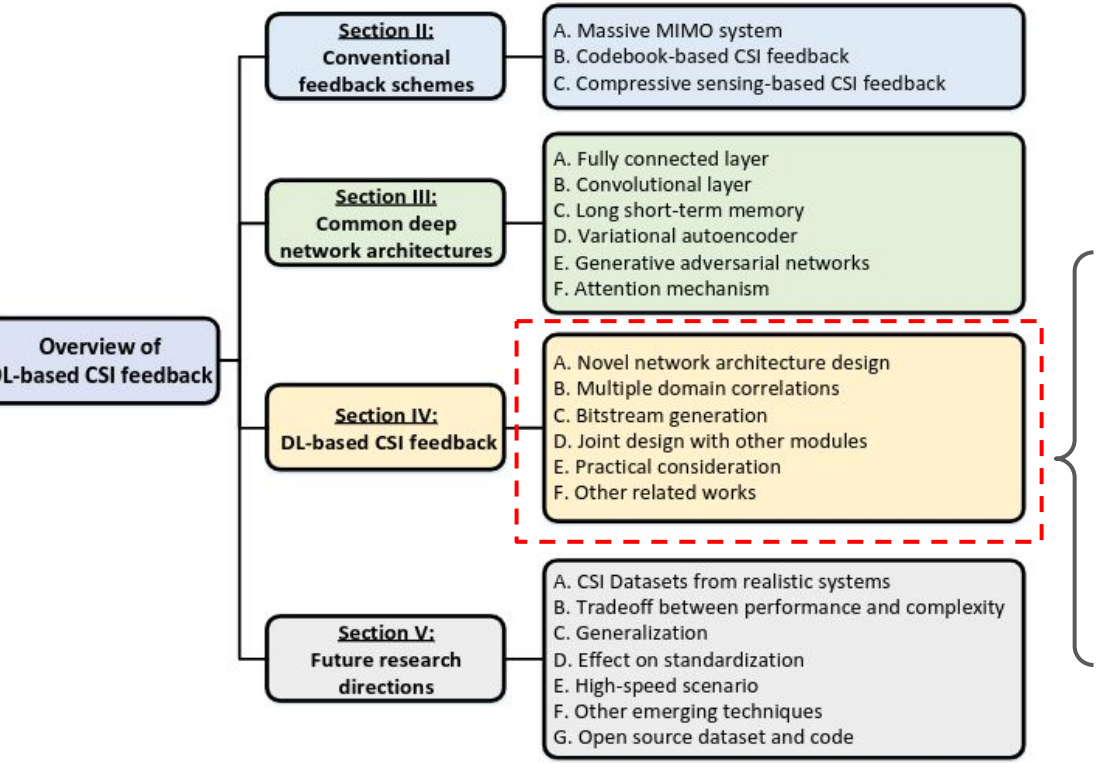
Scenario Under Consideration



- In SISO scenarios, there is only one channel to be estimated
- In massive MIMO, there are **Nt** ✕ **Nr** different channels, demanding extra time-frequency resources to transmit the CSI feedback.
- This calls for techniques to compress the CSI feedback before transmission
- OFDM with 256 subcarriers

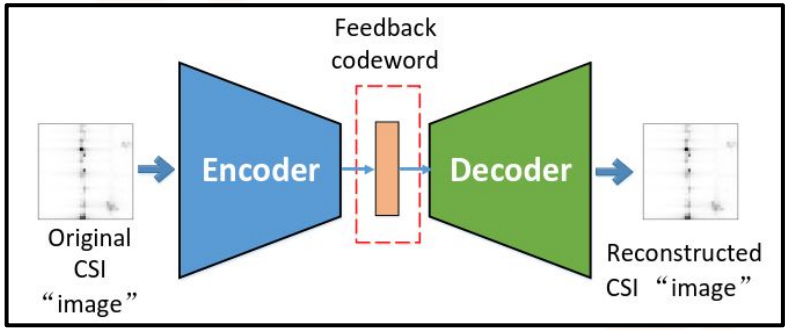
Extra!

CSI Feedback Approaches



DL-based CSI feedback

Concept of end-to-End communication [1]



Such as the CSINet Architecture

Overview of Deep Learning-based CSI Feedback in Massive MIMO Systems, 2022, (Guo et al.)

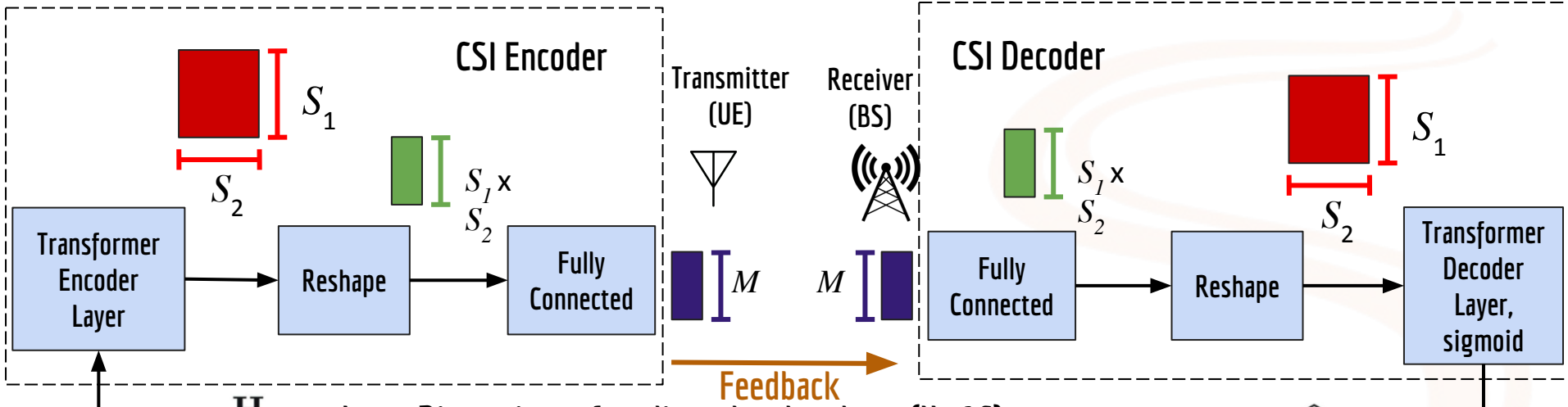
Proposed DL-Based CSI Transformer Architecture

S_1= 64 (embedding length)

S_2= 32 (number of antennas)

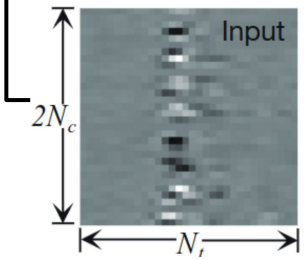
M=128

Single-antenna UE



$\mathbf{H}$

Input Dimensions after discard and reshape (N_c=16)

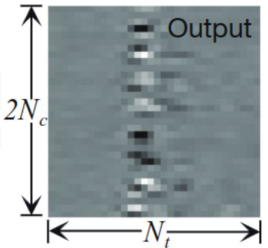


(32, 32)

CSITransformer

This architecture is transformer-based using encoder and decoder

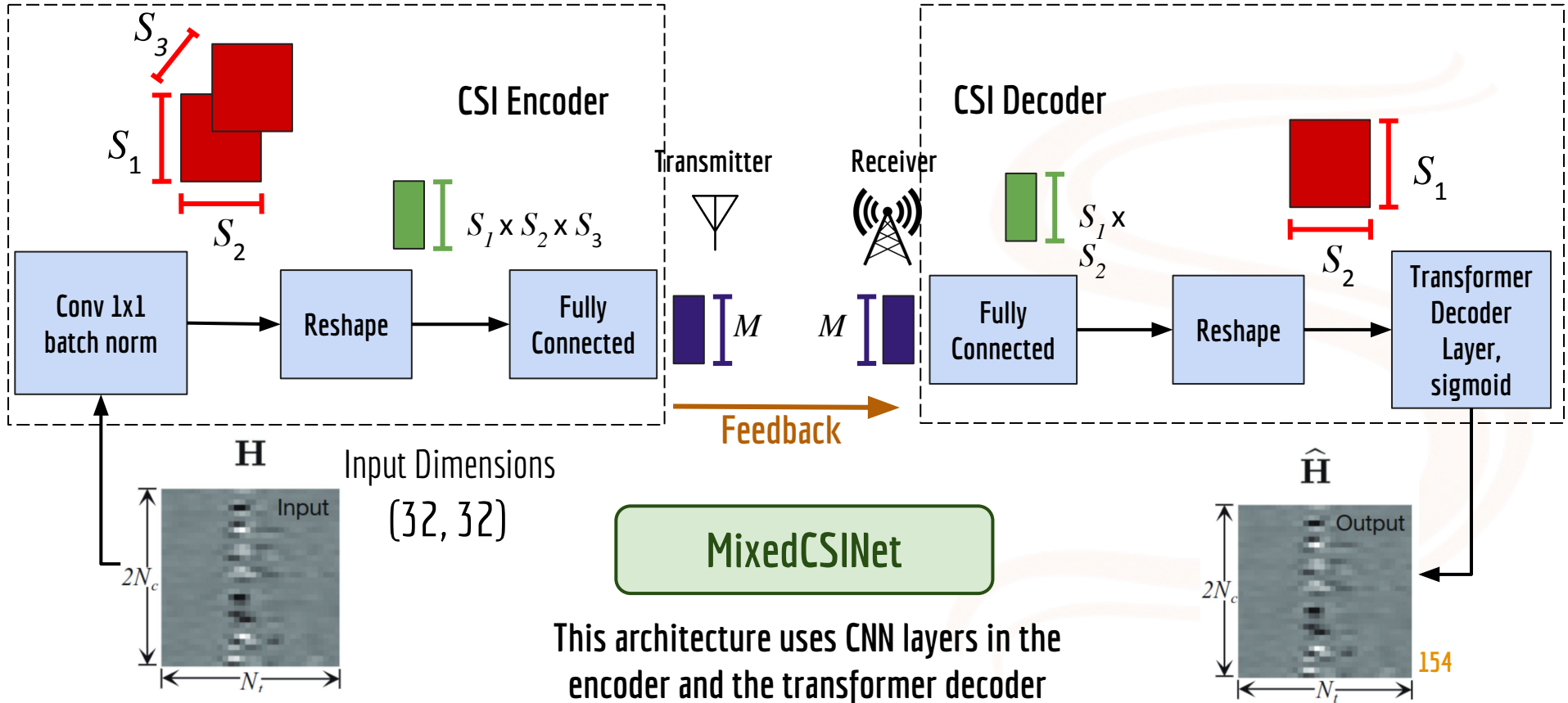
$\hat{\mathbf{H}}$



Extra!

DL-Based CSI Transformer Architectures

Single-antenna UE



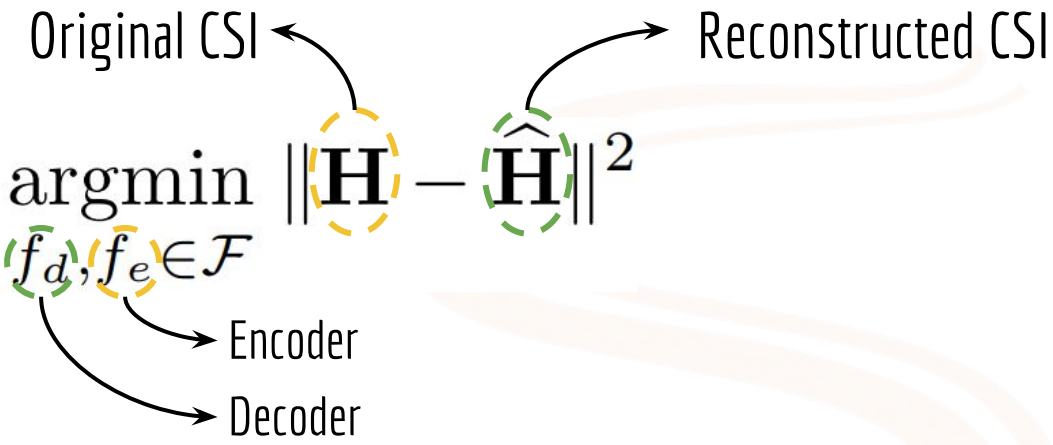
MixedCSINet

This architecture uses CNN layers in the encoder and the transformer decoder

Extra!

Objective and Evaluation Metrics

Optimization problem



Normalized mean-square error

$$\mathrm{NMSE} = E\left\{ \frac{\|\mathbf{H} - \hat{\mathbf{H}}\|^2}{\|\mathbf{H}\|^2} \right\}$$

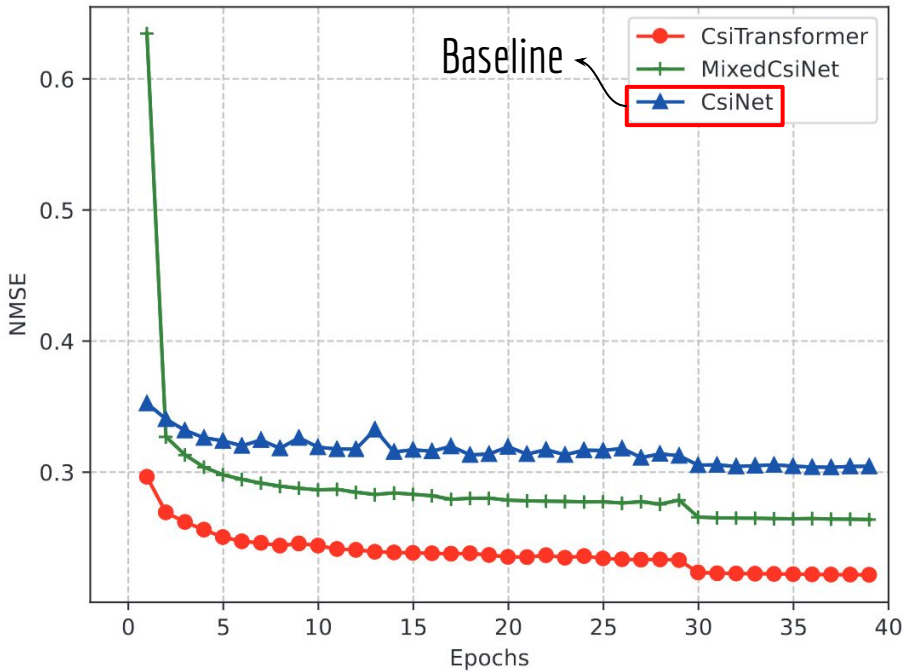
Extra!

Simulation Parameters

Parameter	Value
BS Antennas	32 (ULA)
UE Antennas	1
Subcarriers	256
Carrier frequency	3.5 GHz
Examples	256k (train) plus 64k (test)
Training epochs	40
Learning rate	10^{-3} (first 30 epochs) and 10^{-4} (after)

Results

Assuming a noiseless channel



Cosine similarity

Scheme	NMSE	ρ
CS-CsiNet	0.0682	0.964
CsiNet	0.0203	0.989
CsiTransformer	**0.0069**	**0.996**
MixedCsiNet	0.0100	0.995

CsiTransformer provides better CSI reconstruction

Jupyter notebook implementation presented by Cláudio Modesto

Available at: https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

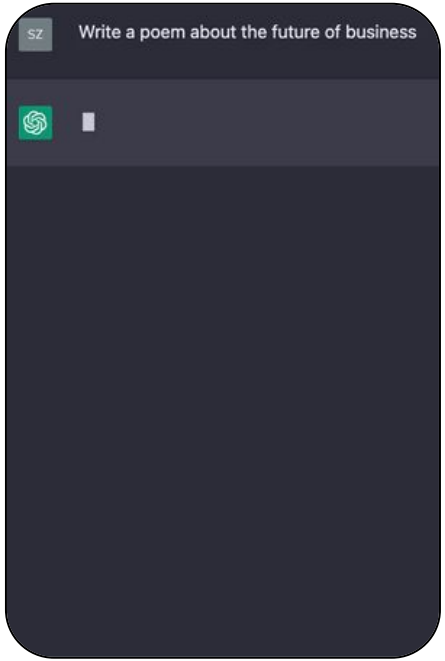
Folder: Csi_feedback_transformer

https://www.youtube.com/watch?v=eMlx5fFNoYc

GPT

G P T

Generative



Pre-trained

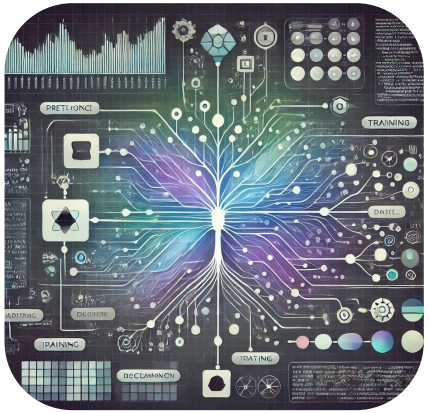
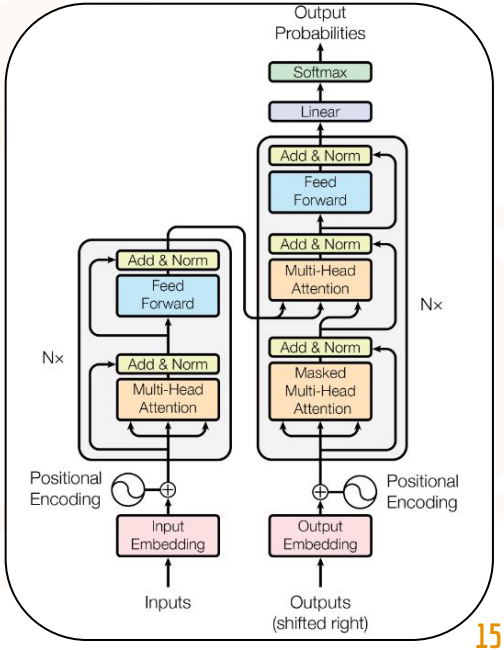


Image generated from chatGPT, pre-trained with an enormous amount of data from several sources

Transformer

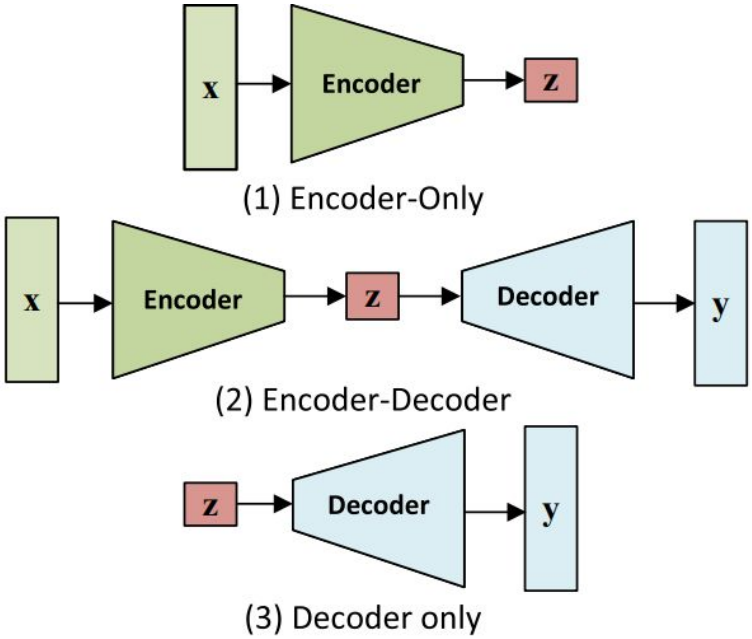


Large Language Models (LLMs)

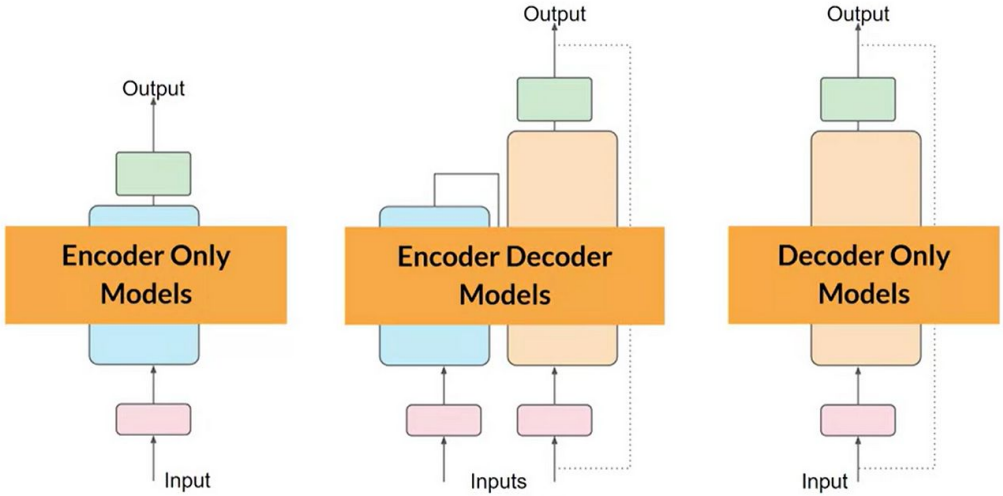
Visualize LLMs with https://bbycroft.net/llm

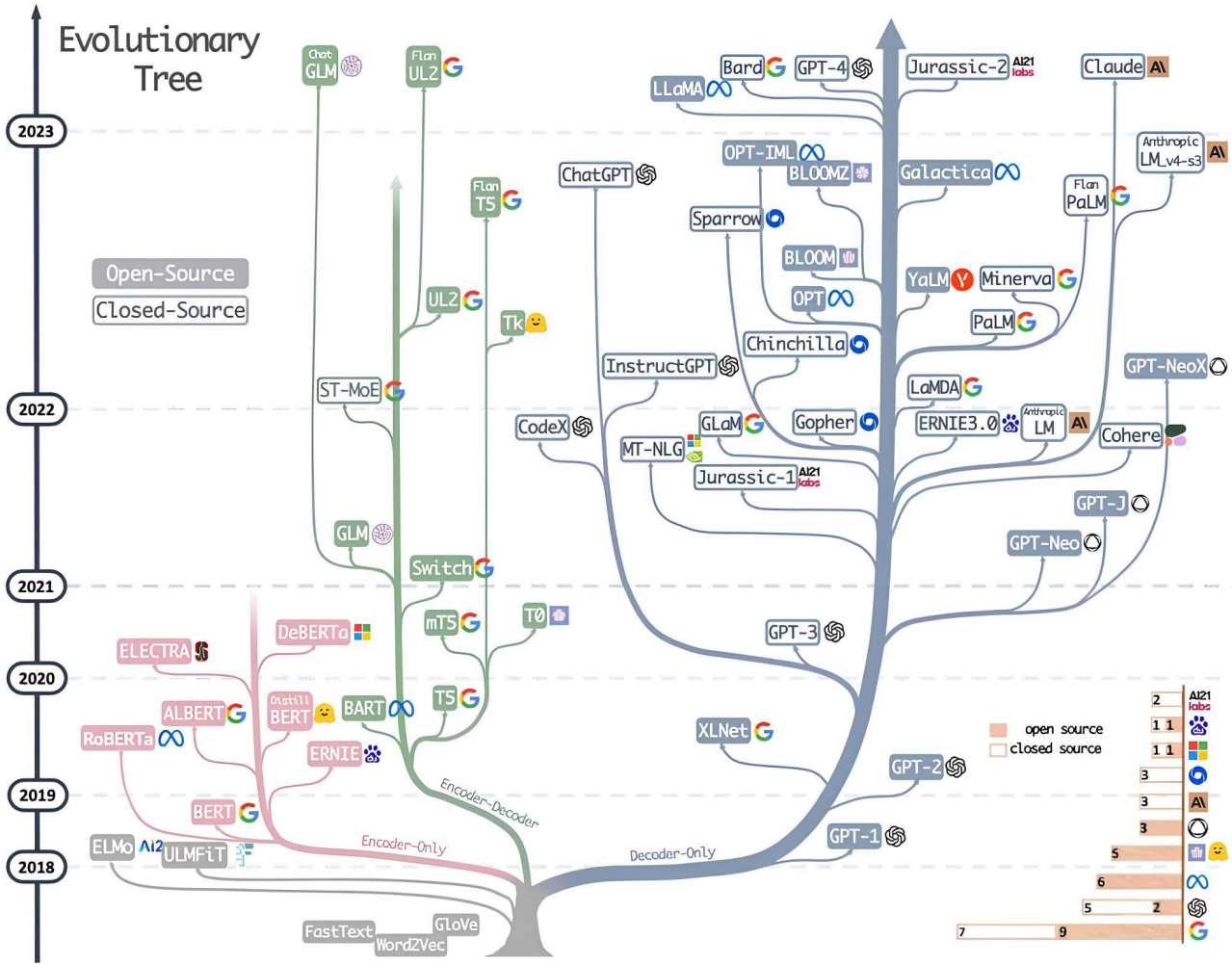
Encoders and decoders in the context of LLMs

Notation for the vanilla autoencoder



Encoders and decoders are more **sophisticated** in LLMs



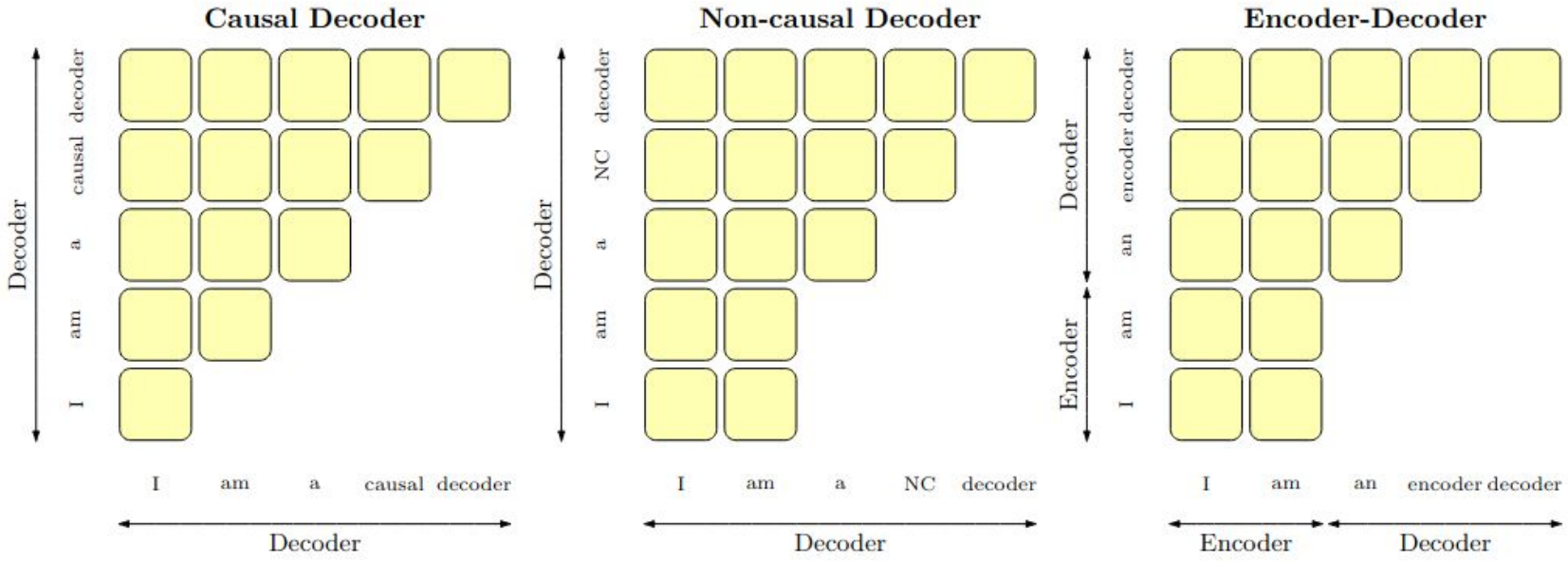


Large Language Models (LLM) Ecosystem

https://github.com/Mooler0410/LLMsPracticalGuide

Extra!

Language Modeling



Full Language Modeling May targets the force be with you

Prefix Language Modeling May the force targets be with you

Masked Language Modeling May targets the force be with you

Prompt Elements

Instructions: directions given to the model

Context data: additional information that serves to increase the accuracy of responses

Input data: the text to be processed by the LLM

Completion: the model's answer to all the given information

Classify the text into neutral, negative or positive:

Context: Evaluating a sentence as negative, neutral or positive consists of identifying the emotional tone or underlying intention of the message.

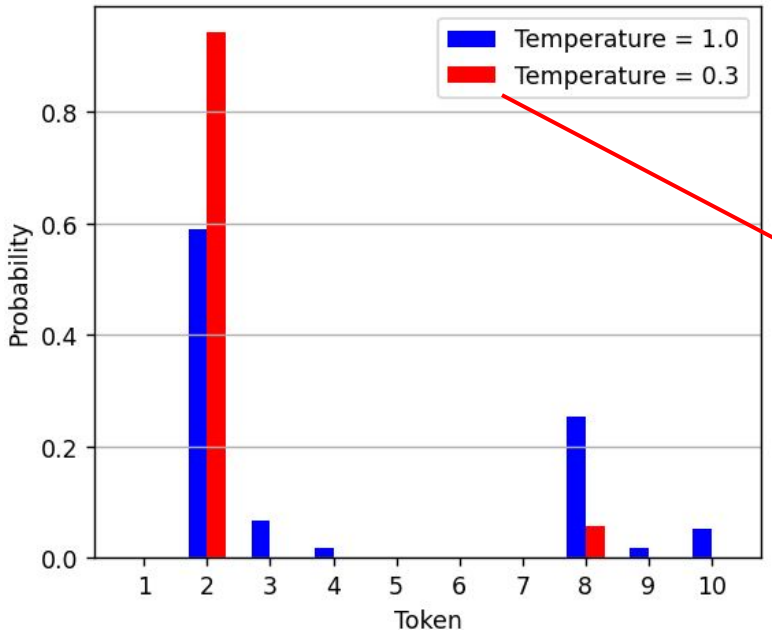
Text: I think the food was okay.

Sentiment: Positive

https://www.cse.iitd.ac.in/~mausam/courses/col772/spring2023/lectures/22-promptengg.pdf

Temperature Sampling

$$\text{softmax}(x_i) = \frac{e^{x_i/T}}{\sum_j e^{x_j/T}}$$

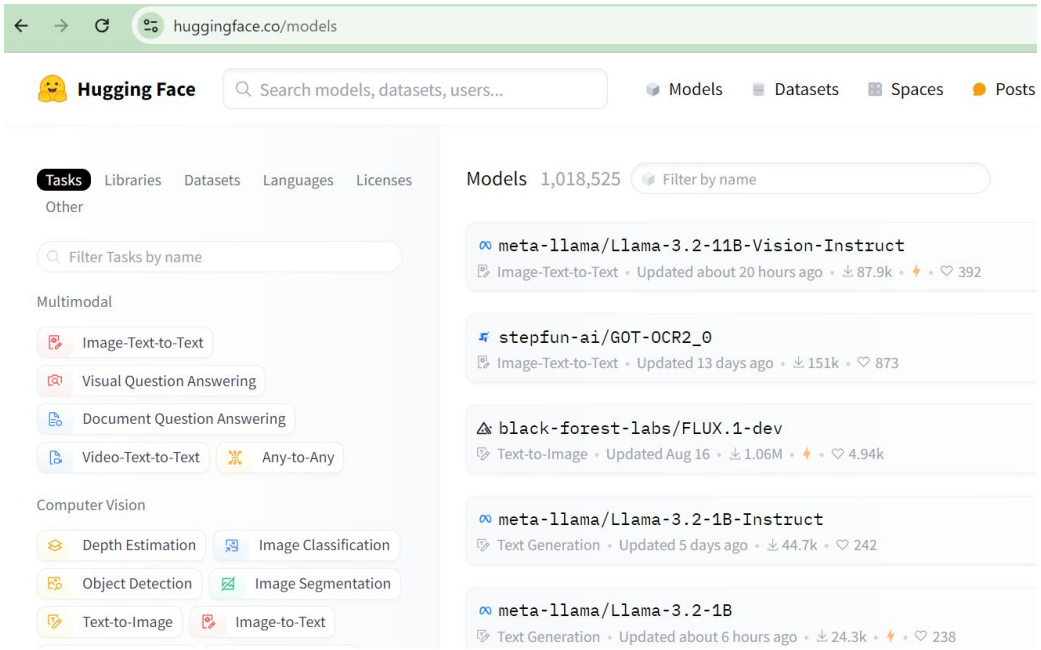


Typical values for ChatGPT:

0.0 to 0.3: Focused and deterministic
0.4 to 0.7: Balanced
0.8 to 1.0: Creative and diverse

Lower temperature: the probability concentrates on most probable tokens

Hugging Face and pipeline examples



Available:

- 1 million **models**
- 220K **datasets**

APIs such as transformers: https://huggingface.co/docs/transformers/en/index and associated pipeline objects for inference

Check https://www.langchain.com and others!

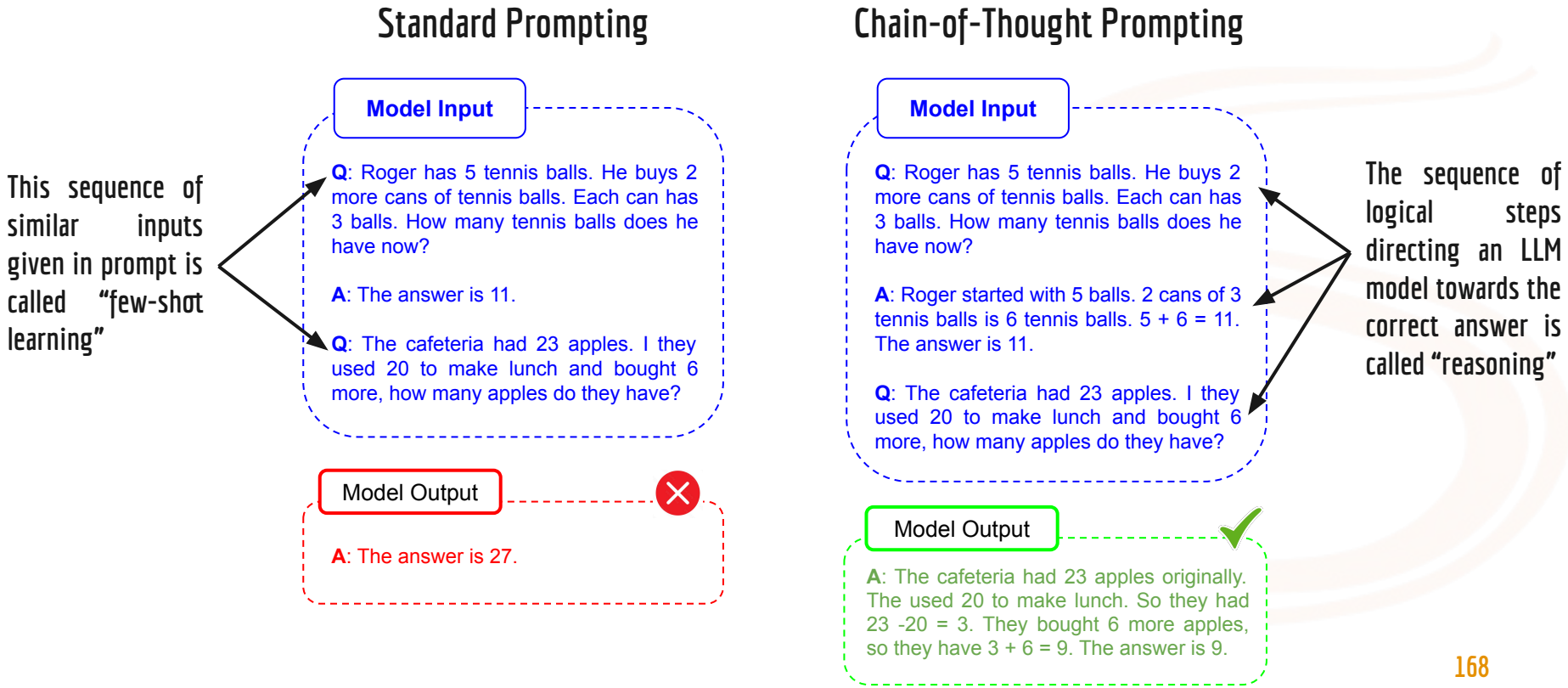
https://www.youtube.com/watch?v=LyRGSijgGT8 - YOLO VISION 2023 | Open Source Vision With Hugging Face Transformers
https://www.youtube.com/watch?v=QEaBAZQCtwE - Hugging Face
https://huggingface.co/docs/transformers/en/installation

Jupyter notebook implementation presented by Aldebaro Klautau

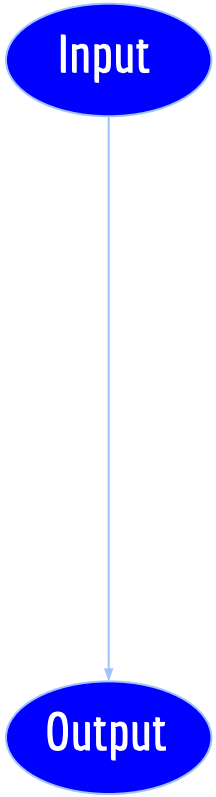
Available at: https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

Folder: HuggingFace_Pipelines

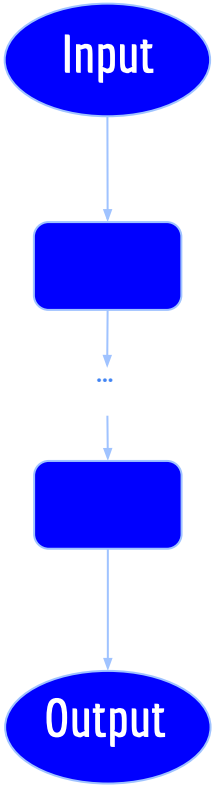
Prompt Engineering with Chain-of-Thoughts



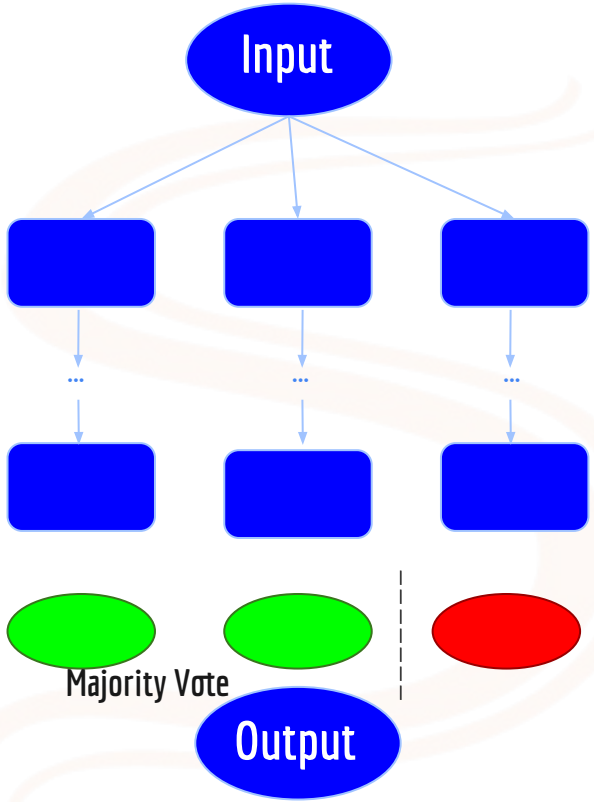
Reasoning with Chain-of-Thoughts



Input-Output Prompting (IO)

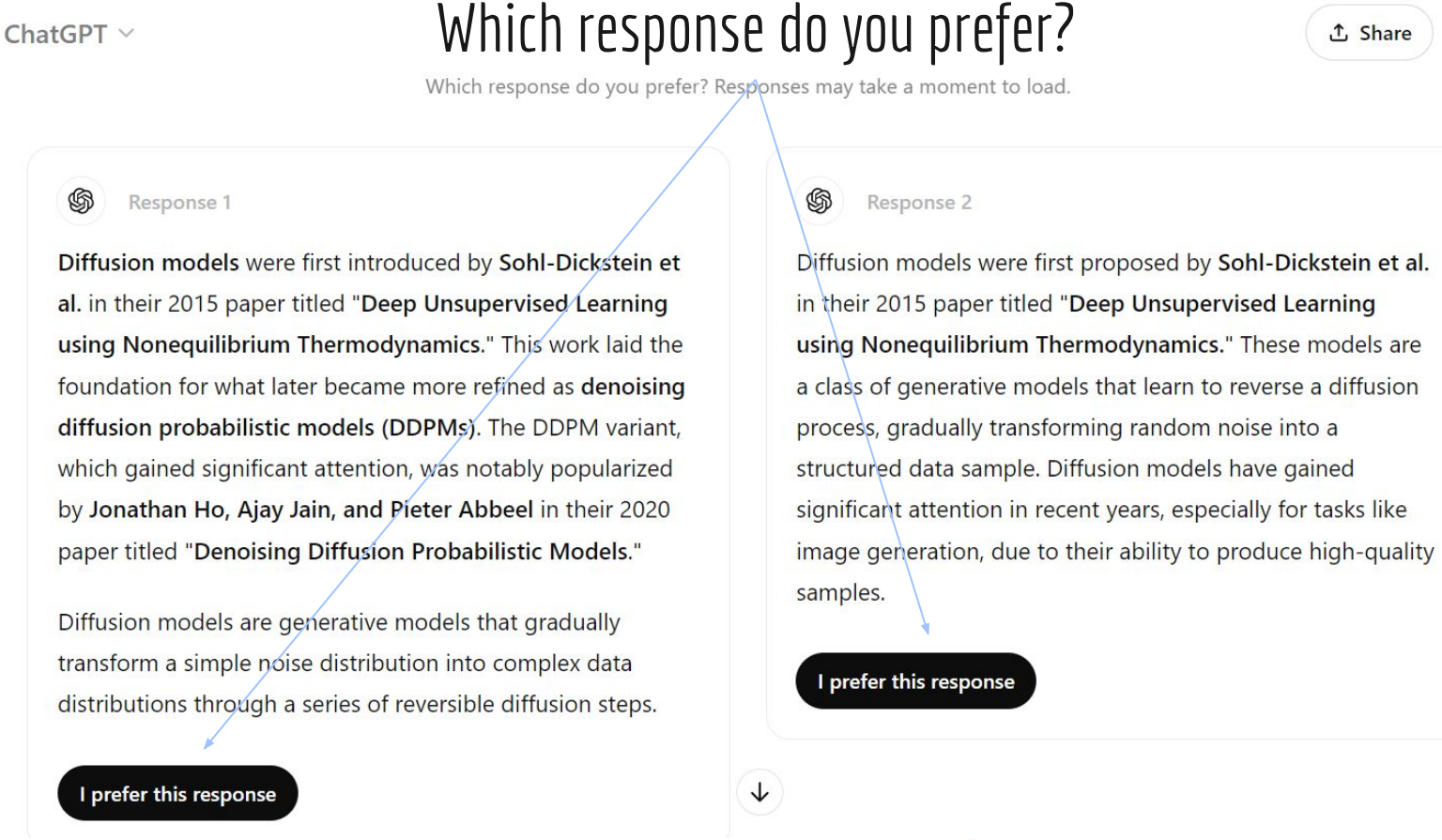


Chain-of-Thoughts (CoT)



Self-Consistency with CoT (CoT-SC)

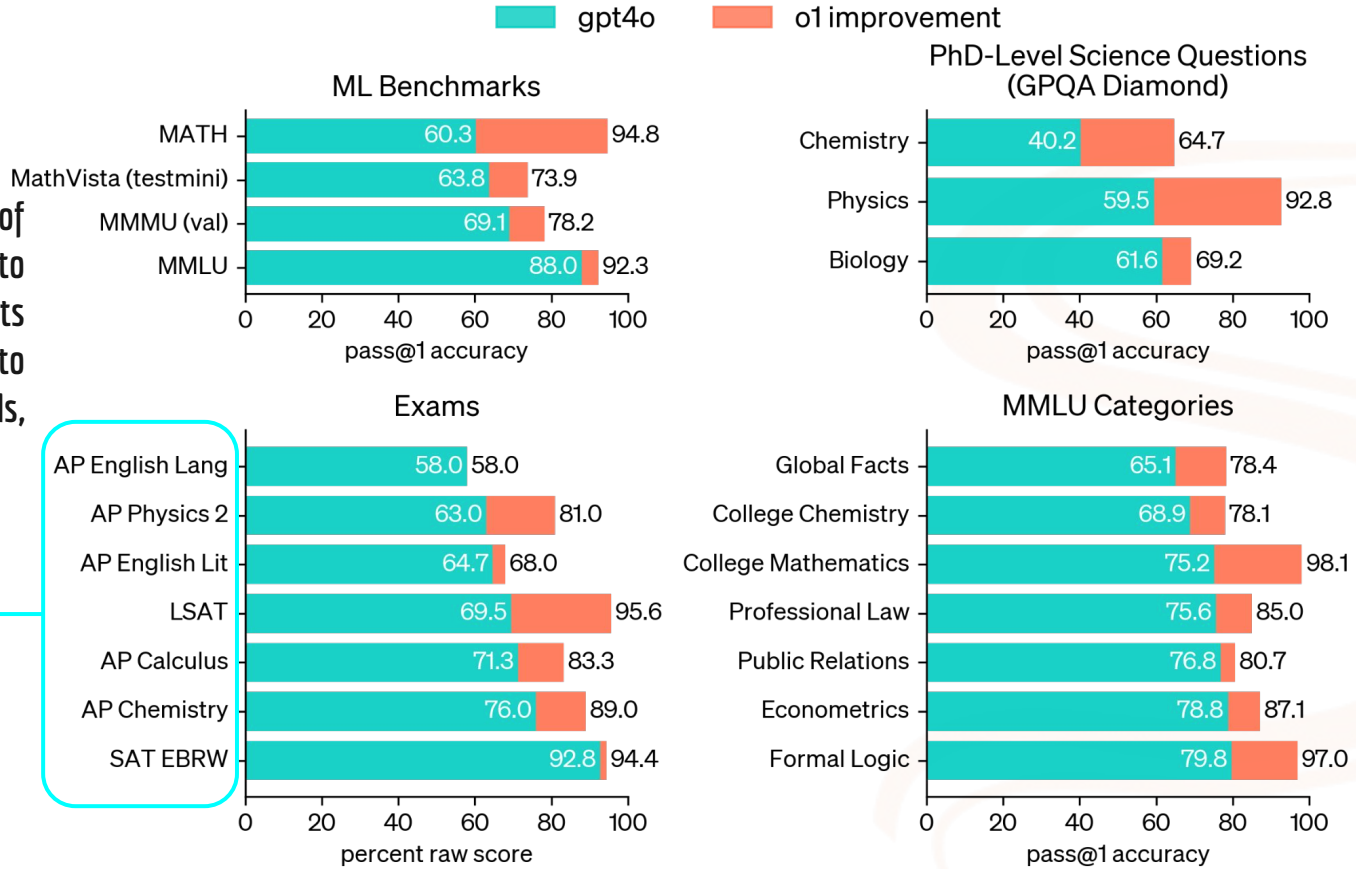
Reinforcement learning with human feedback (RLHF)



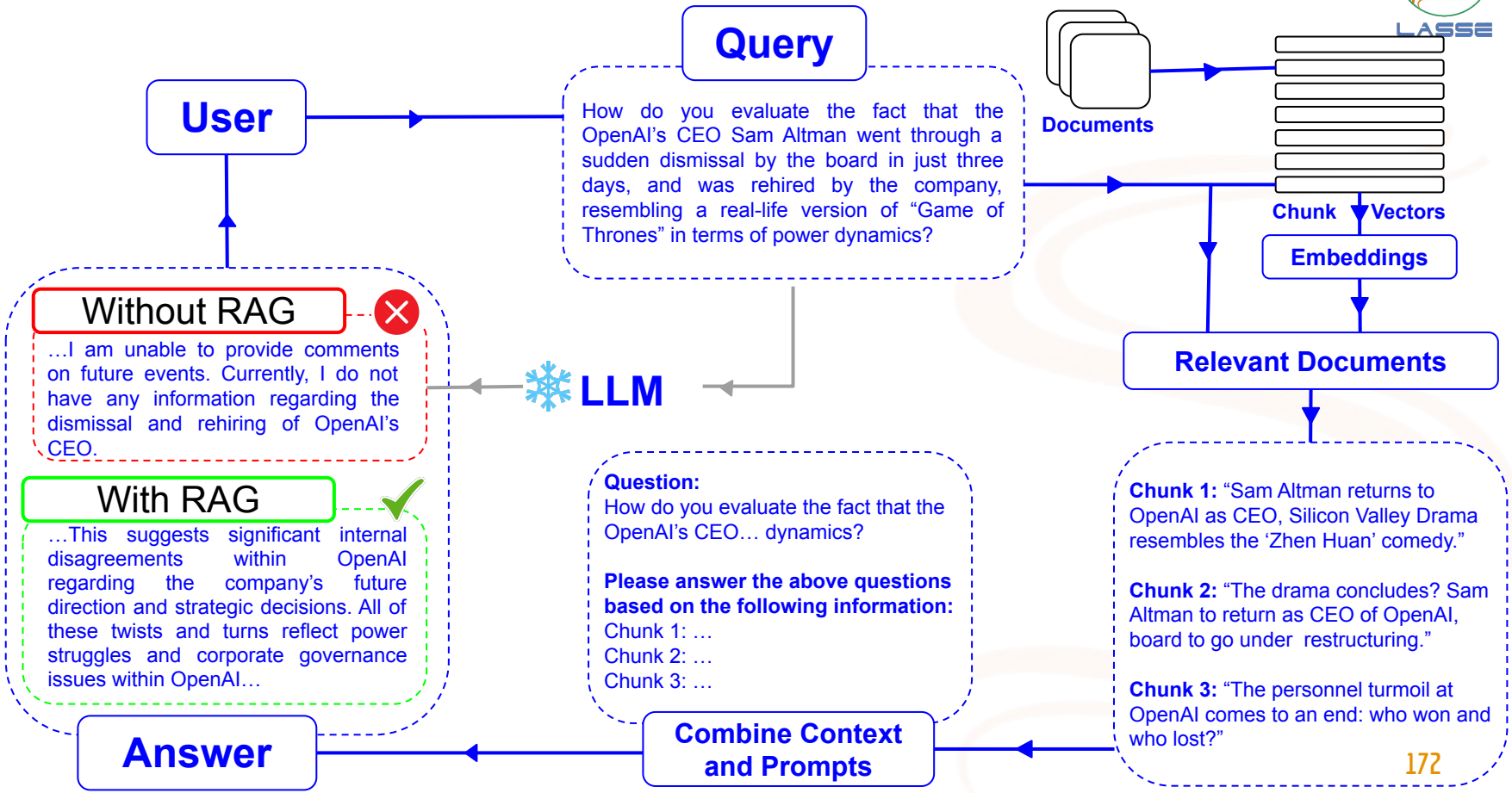
Extra!

Reasoning with OpenAI's o1

A complex process of reasoning is used to achieve results compared to established models, such as GPT4o.



Retrieval-Augmented Generation (RAG)

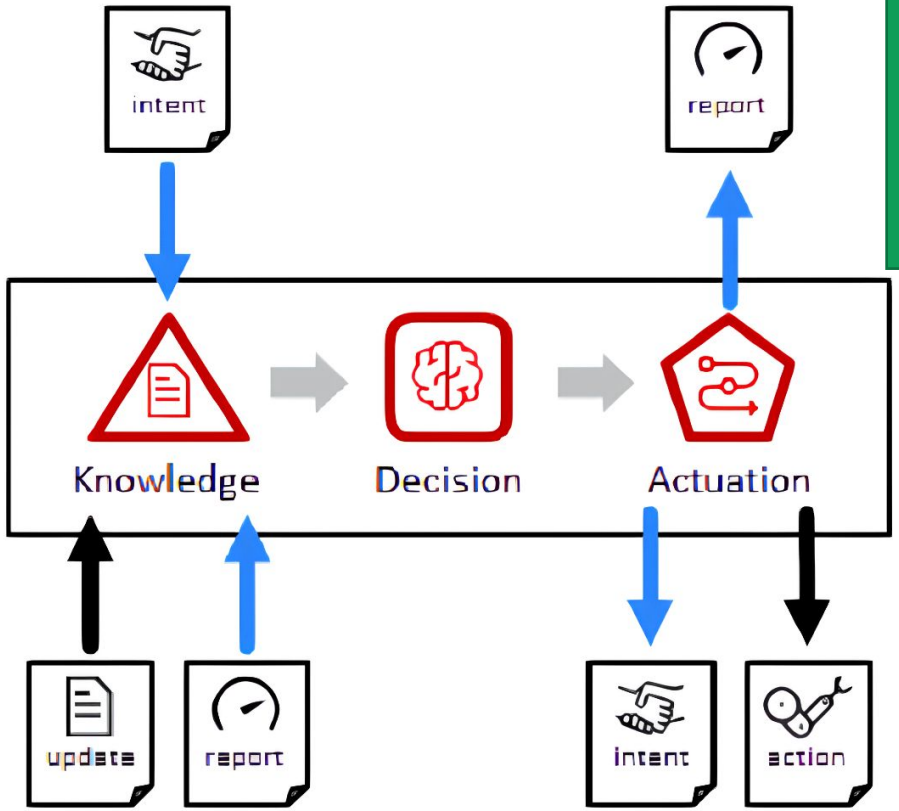


Part IV - Transformers and LLMs Applied to Telecommunications

Application of Large Language Models (LLMs) to Telecom

Usage of LLMs in Intent-Based Networks (IBN) for translation from natural language to configuration file formats.

Intent-Based Networks Overview



In this scenario, the users communicate their expectations (intent) for an application and an autonomous system provides resources to attend the demands.

Intent Handling Function

The basic building block of intent-based operation

Intent API

Other (not intent based) API

[1]"IG1253 Intent in Autonomous Networks v1.3.0," TM Forum. https://www.tmforum.org/resources/standard/ig1253-intent-in-autonomous-networks-v1-0-0/

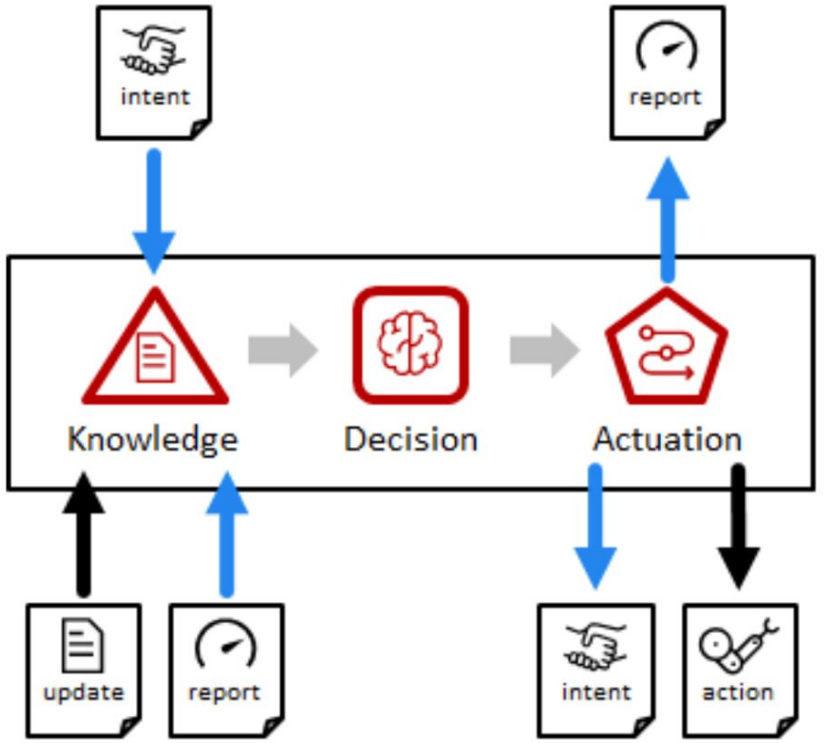
Extra!

Intent-Based Networks Overview

LASSE

"Intent is the formal specification of all expectations including requirements, goals, and constraints given to a technical system"

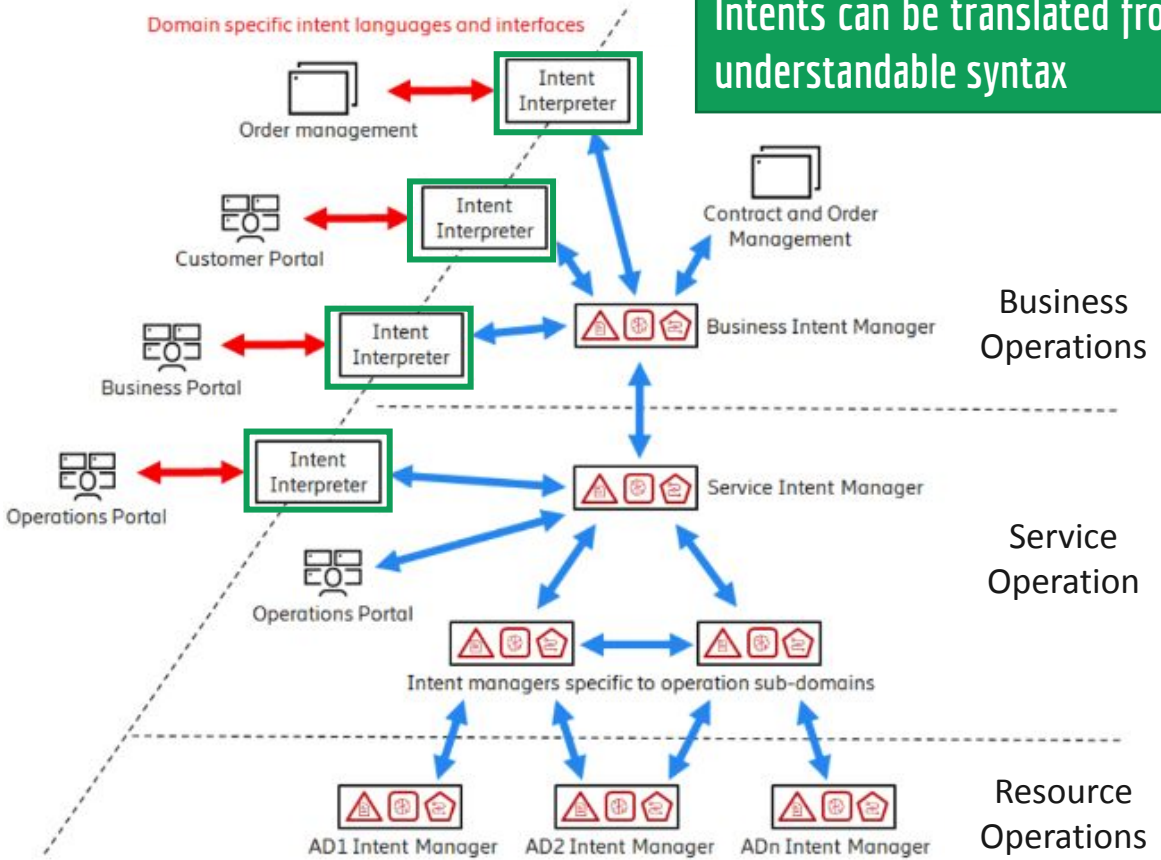
There are several definitions, architectures and languages for expressing intents (UML, JSON, RDF-Resource Description Framework, TURTL, etc)



[1]"IG1253 Intent in Autonomous Networks v1.3.0," *TM Forum*. https://www.tmforum.org/resources/standard/ig1253-intent-in-autonomous-networks-v1-0-0/

Intent-Based Systems

Intents can be translated from natural language to a machine understandable syntax



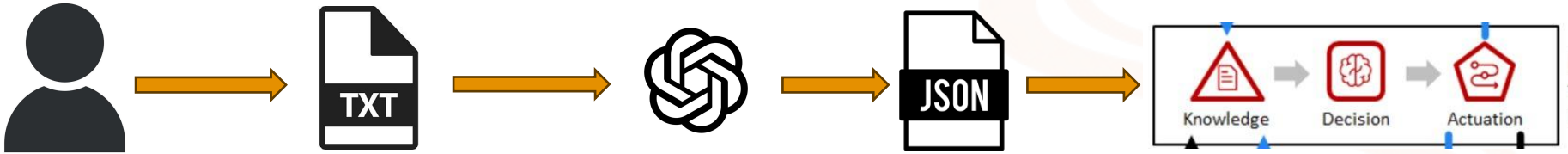
The intent handler communicates with users and other handlers from differents parts of an application

Applications may have different language levels, each with specific intents

[1]"IG1253 Intent in Autonomous Networks v1.3.0," *TM Forum*. https://www.tmforum.org/resources/standard/ig1253-intent-in-autonomous-networks-v1-0-0/

LLM models in intent-based networking

The user's intent goes through a LLM model that converts the input to a machine understandable format



User describes the desired network application in natural language

The new file is provided to the autonomous system to process the user's intent

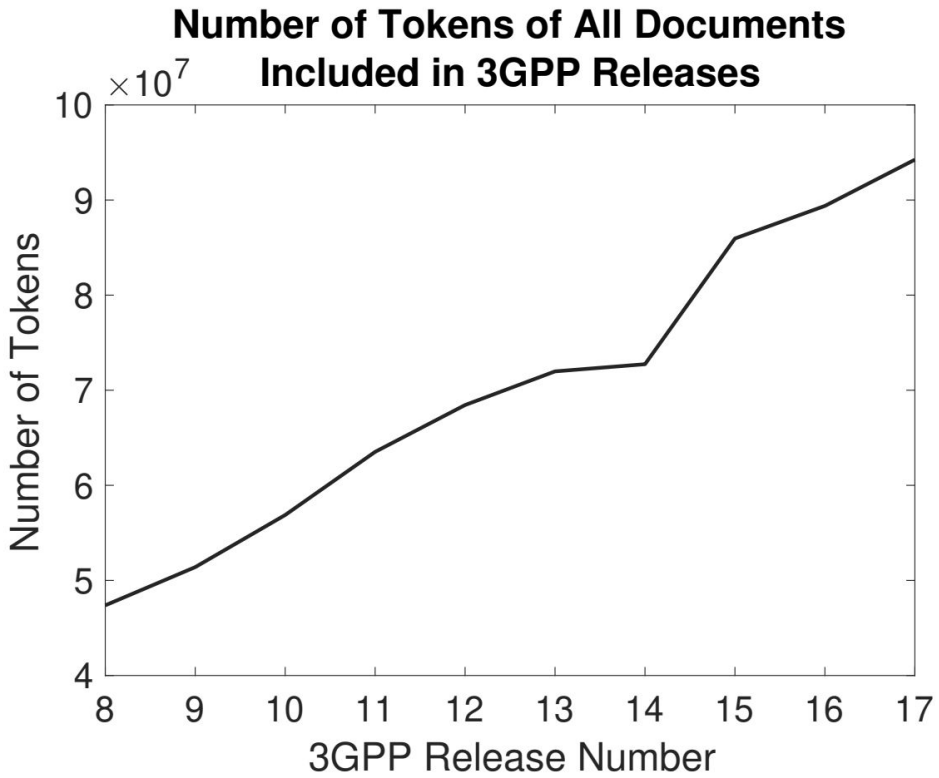
[1] Intent in Autonomous Networks. TM Forum. 2022
[2] https://thenounproject.com/browse/icons/term/json-file/
[3] https://dryicons.com/icon/txt-file-format-icon-9756
[4] https://seeklogo.com/vector-logo/465219/chatgpt

Application of Large Language Models (LLMs) to Telecom

Bisecting K-Means in RAG for Enhancing Question-Answering Tasks Performance in Telecommunications

Pedro Sousa, Cláudio Klautau, Frank Morte and Luis Navarro

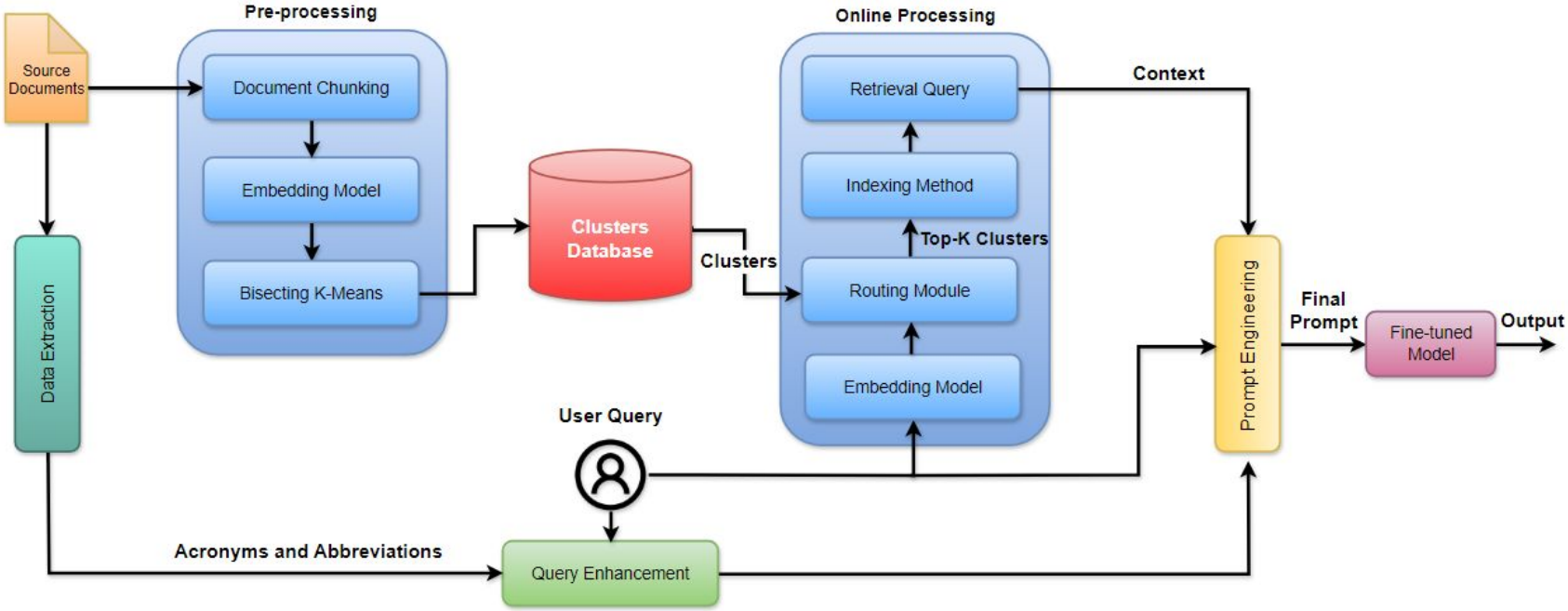
LLMs as digital assistants for 3GPP standards



A. Karapantelakis et al., "Using Large Language Models to Understand Telecom Standards," ICMLCN, 2024

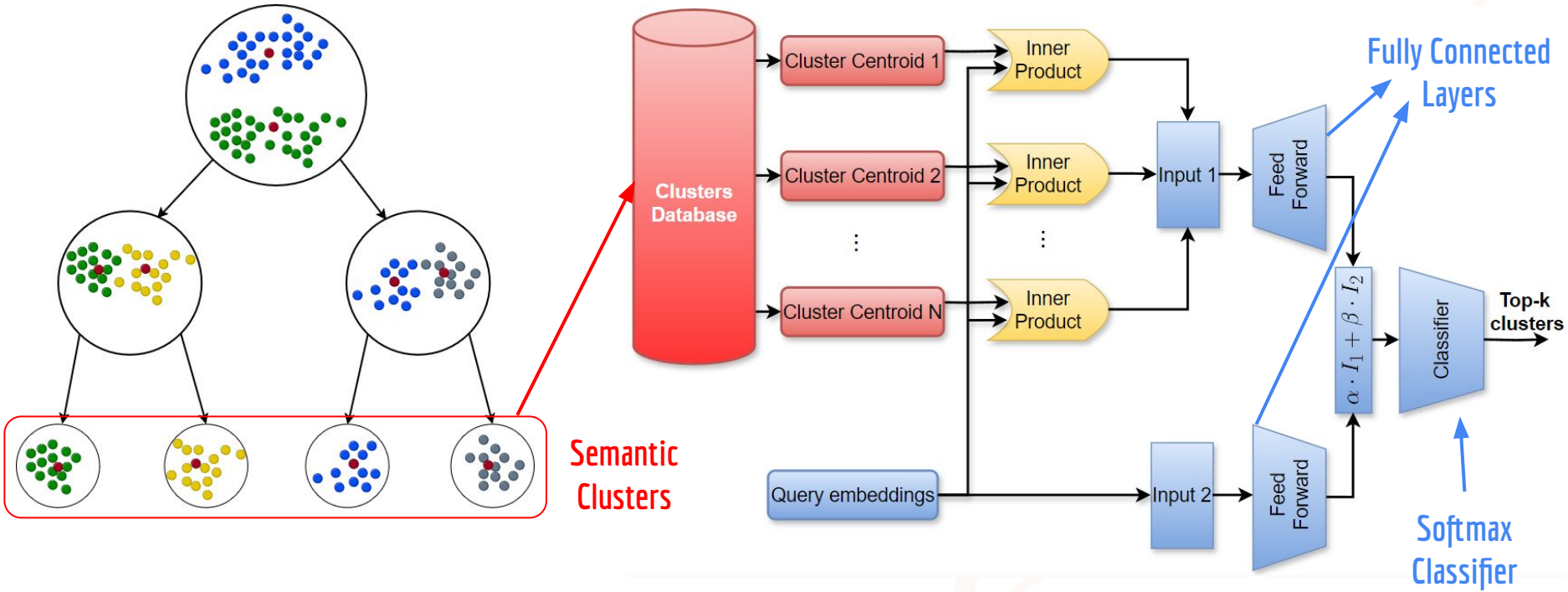
K-Means based RAG framework for telecom documents

3GPP Release 18 docs



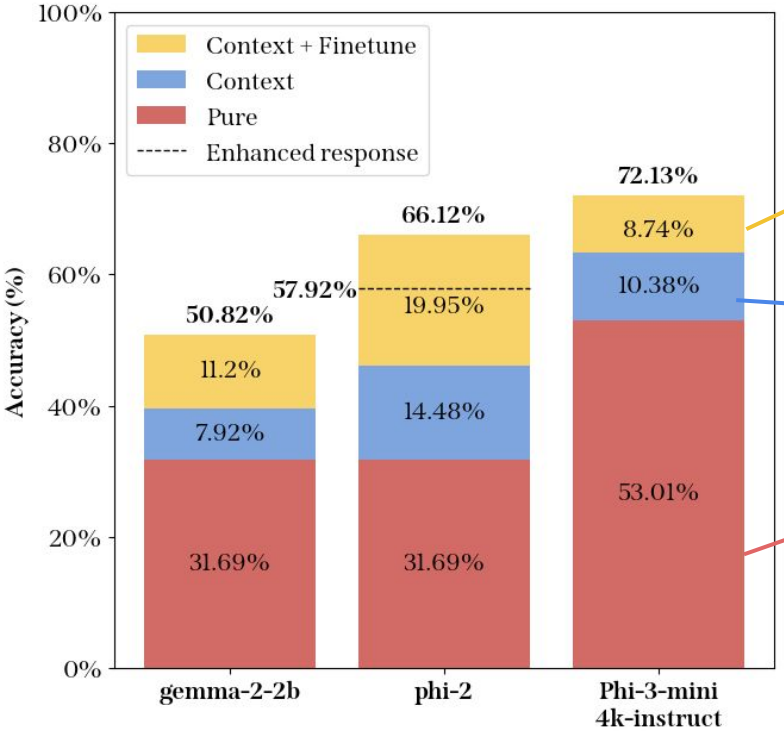
P. Sousa, C. Klautau, F. B. Morte and L. Navarro, "Bisecting K-Means in RAG for Enhancing Question-Answering Tasks Performance in Telecommunications," 2024 IEEE Globecom Workshops (GC Wkshps), Cape Town, South Africa, 2024.

Semantic clusters for optimized searches



P. Sousa, C. Klautau, F. B. Morte and L. Navarro, "Bisecting K-Means in RAG for Enhancing Question-Answering Tasks Performance in Telecommunications," 2024 IEEE Globecom Workshops (GC Wkshps), Cape Town, South Africa, 2024.

Results for Different Small Language Models (SLMs)



New training on the telecom dataset and the framework

Just the framework processing results

Only a new training with the original documents

P. Sousa, C. Klautau, F. B. Morte and L. Navarro, "Bisecting K-Means in RAG for Enhancing Question-Answering Tasks Performance in Telecommunications," 2024 IEEE Globecom Workshops (GC Wkshps), Cape Town, South Africa, 2024.

Jupyter notebook implementation presented by Pedro Sousa

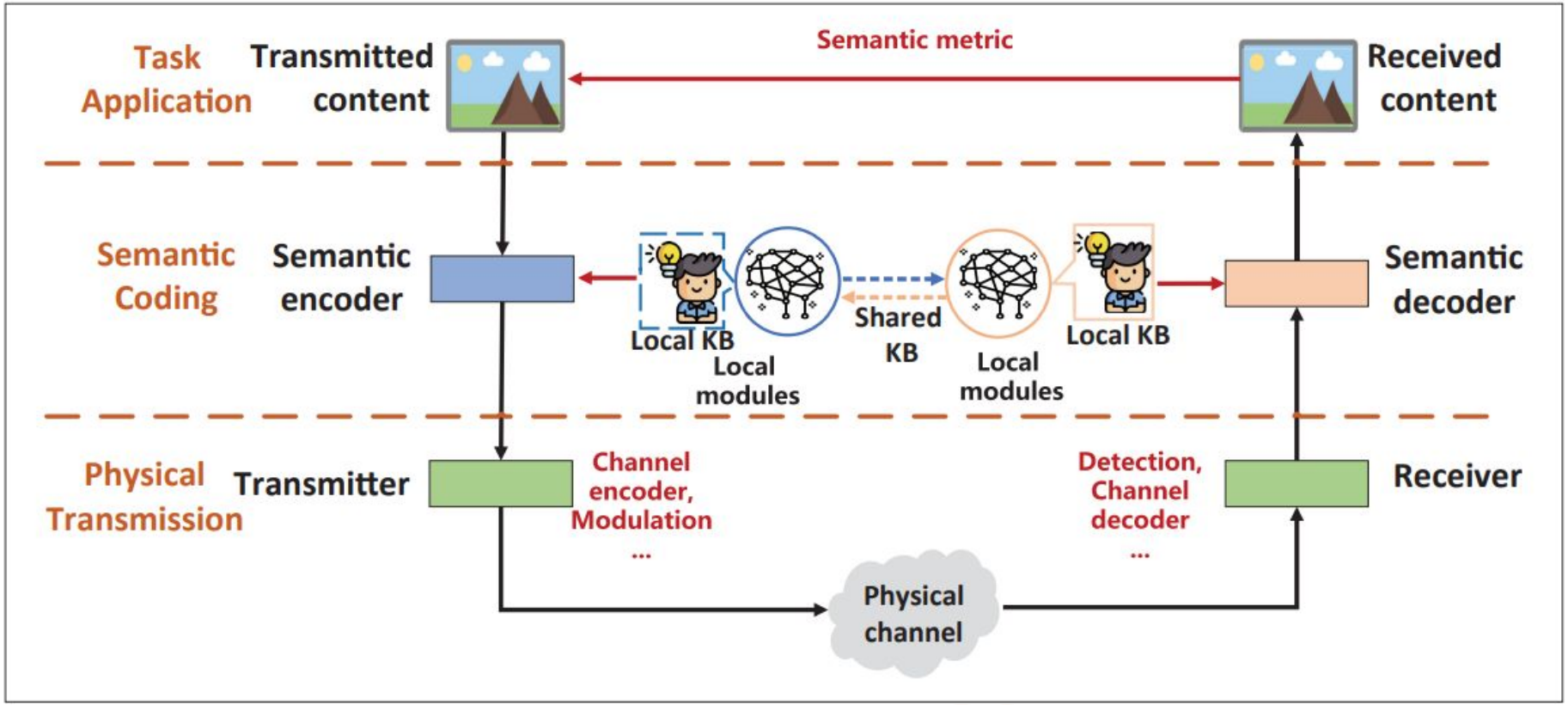
Available at: https://github.com/lasseufpa/sbrt2024_shortcourse_gen_ai

Folder: Clustering_for_RAG_in_Telecom_Domain

Semantic Communications

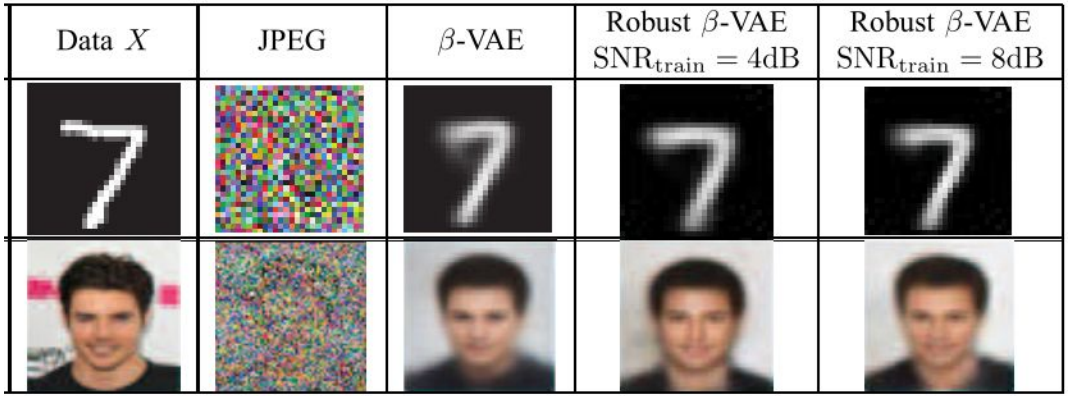
Extra!

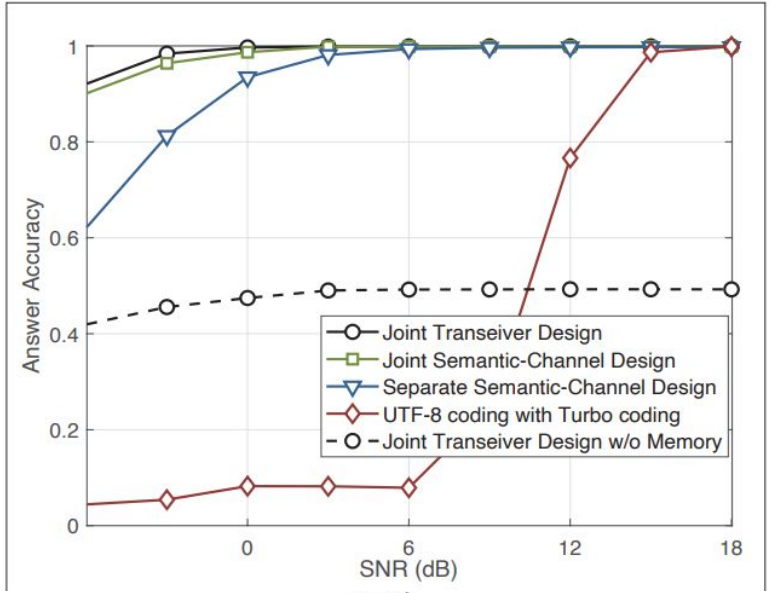
Semantic communications

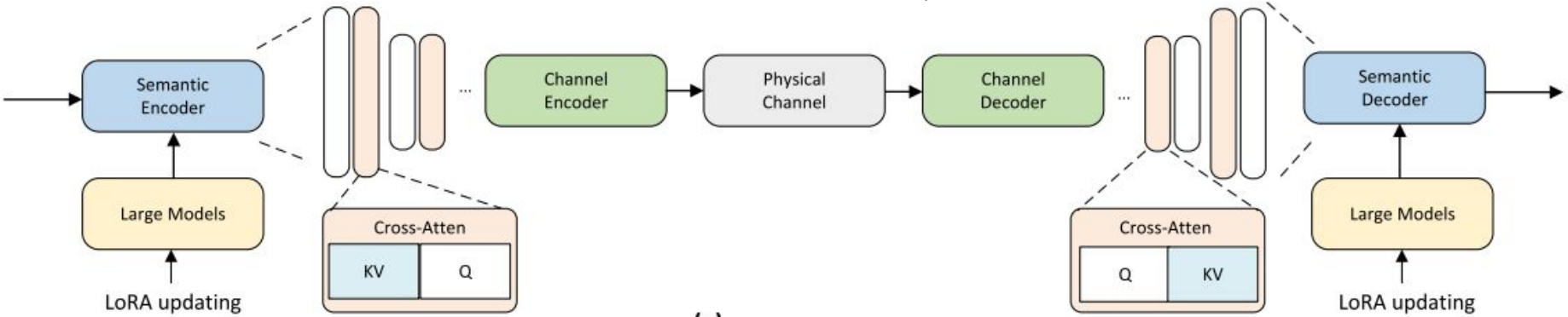


Jiang et al, Semantic communications using foundation models: Design approaches and open issues, 2024

Semantic communications







(e)

[1] Ma et al, Task-Oriented Explainable Semantic Communications, 2023
[2] Xie et al, Towards intelligent communications: Large model empowered semantic communications, 2024
[3] LoRA: Low-Rank Adaptation - https://arxiv.org/pdf/2106.09685, 2021

LLM applied to the RAN

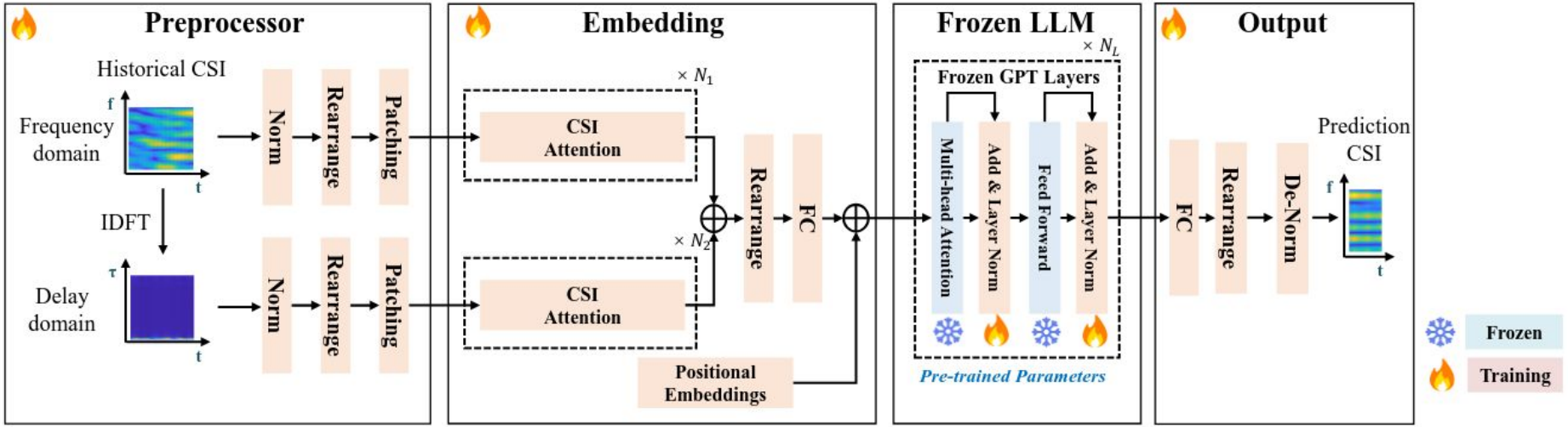
LLM4CP: Adapting Large Language Models for Channel Prediction

- Uses LLMs (**GPT-2**) to predict **future downlink CSI** based on **historical uplink CSI**
- Inspired by [2], which: **fine-tunes frozen pre-trained LLM on time series** datasets and achieves state-of-the-art performance on main time series analysis tasks
- **Cross-modality knowledge transfer**: pre-trained LLMs cannot directly process non-linguistic data. But similar to how text is preprocessed into tokens by the tokenizer, one obtains **embeddings of CSI data** using the proposed preprocessor and embedding module
- During the training process, multi-head attention and feed forward layers are frozen to retain universal knowledge, while addition, layer normalization, and positional embedding are fine-tuned for adapting the LLM to the channel prediction

[1] B. Liu, X. Liu, S. Gao, X. Cheng and L. Yang, "LLM4CP: Adapting Large Language Models for Channel Prediction," in *Journal of Communications and Information Networks*, vol. 9, no. 2, pp. 113-125, June 2024,

[2] Zhou et al, "One Fits All:Power General Time Series Analysis by Pretrained LM", https://arxiv.org/abs/2302.11939, 2023

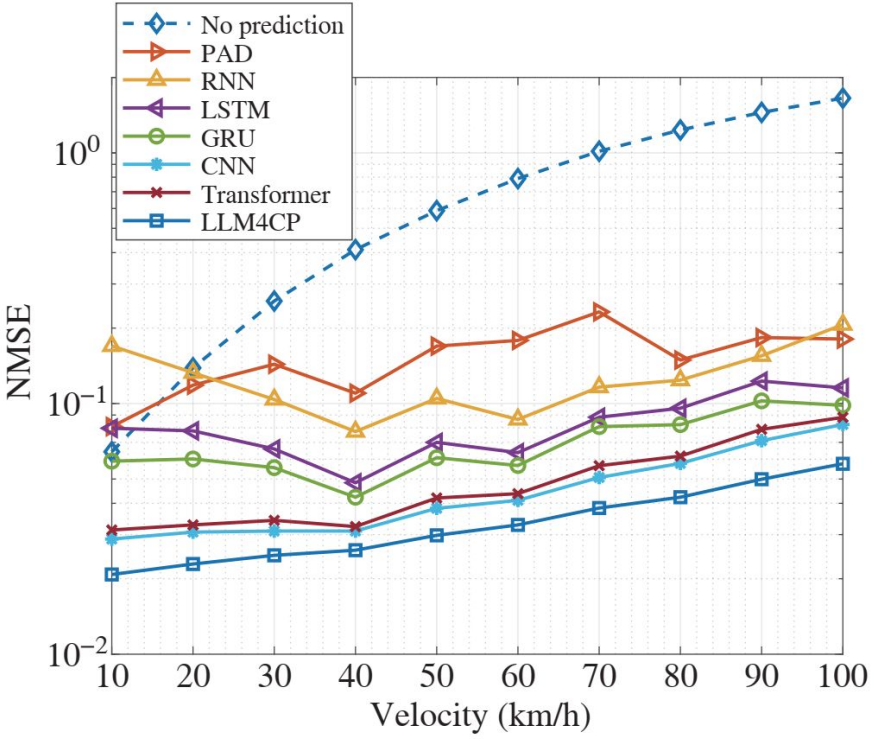
Channel Prediction with LLMs



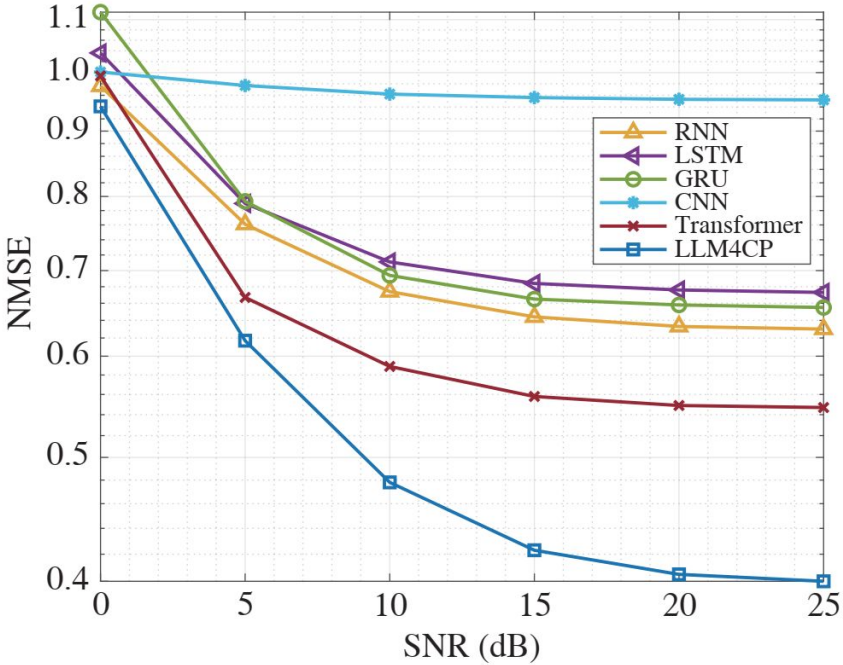
B. Liu, X. Liu, S. Gao, X. Cheng and L. Yang, "LLM4CP: Adapting Large Language Models for Channel Prediction," in *Journal of Communications and Information Networks*, vol. 9, no. 2, pp. 113-125, June 2024

Results using channels generated by Quadriga

Performance for different user velocities assuming time-division duplexing (TDD)



Noisy conditions: performance for different SNRs assuming frequency-division duplexing (FDD)



Computational cost

Network parameters (training parameters/total parameters) and the training/interference cost per batch

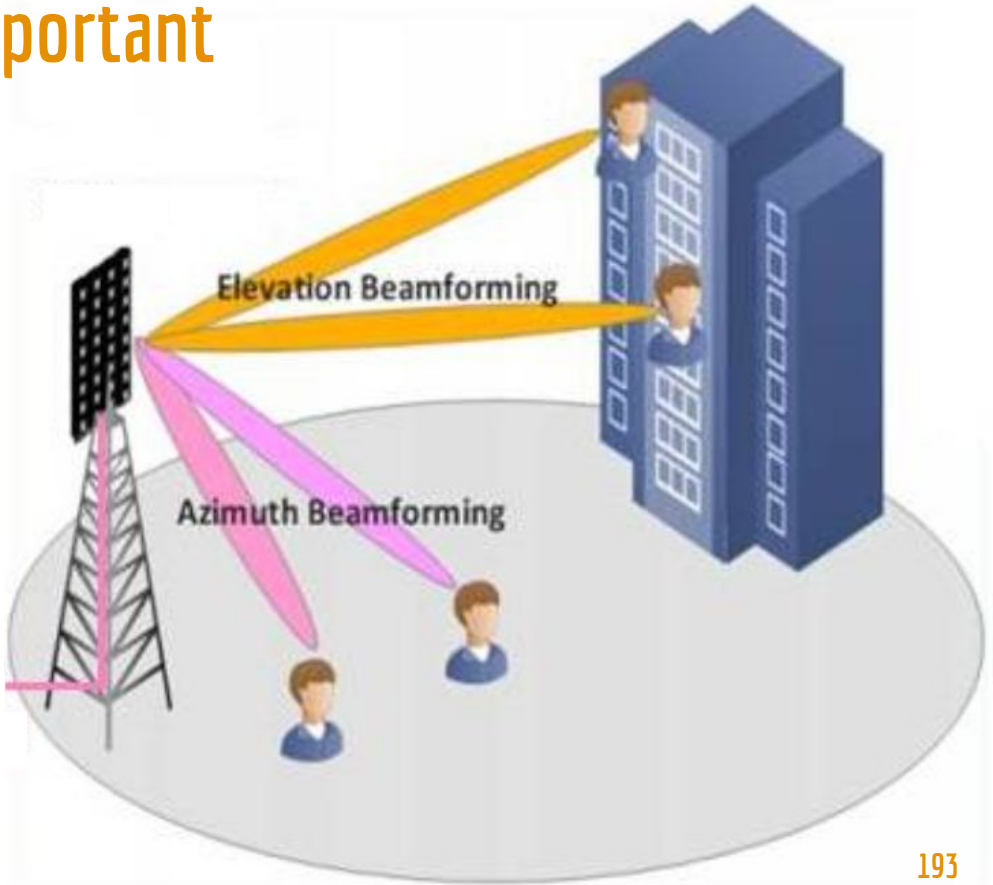
Metric	RNN	LSTM	CNN	Transformer	LLM4CP
Network parameters (1×10^6)	0.30/0.30	1.13/1.13	3.14/3.14	1.76/1.76	1.73/82.87
Training time (ms)	4.59	7.40	2.03	21.84	6.84
Interference time (ms)	3.55	5.19	0.52	18.35	5.84

LLM's inference time is shorter than that of the Transformer thanks to inference acceleration specific to the GPT model

5G and 6G use directional communications ⇒ beam management is important

3D or full dimension (FD) MIMO to improve throughput and/or coverage while minimizing interference

Based on CSI, transmitter can use a "**precoder**" and the receiver can use a "**combiner**"



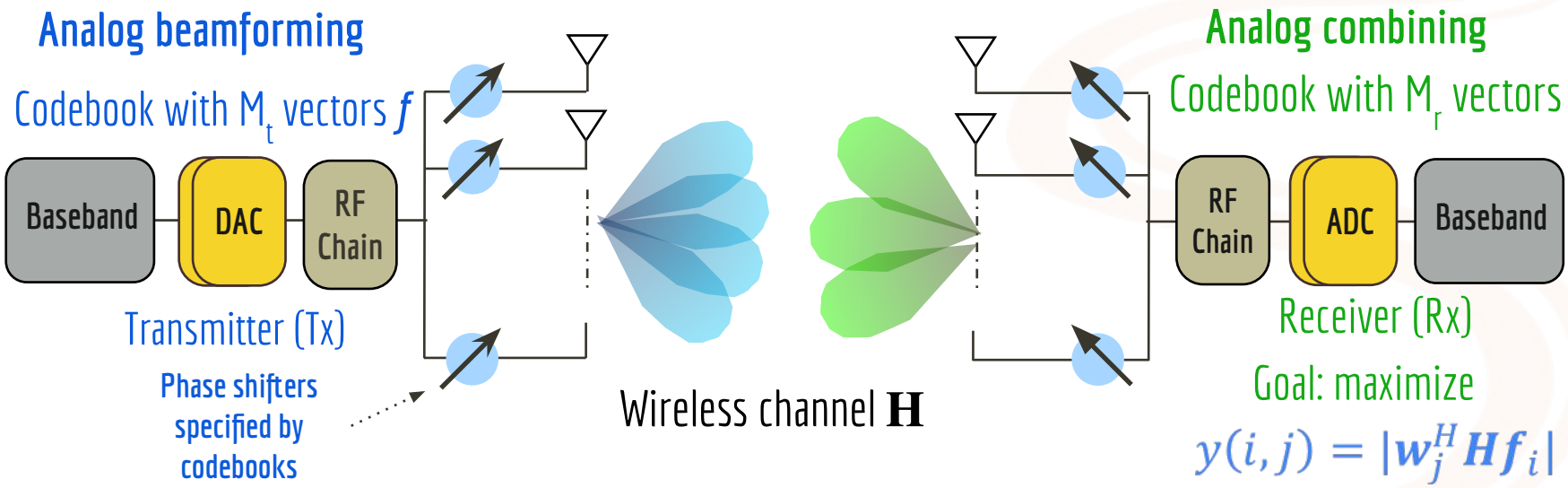
[1] Advanced Antenna Systems for 5G Network Deployments, H. Asplund et al, 2020
[2] https://www.rfwireless-world.com/Terminology/Advantages-of-FD-MIMO.html

Extra!

Beam selection in 5G mmWave using analog (codebook-based) architecture

LASSE

Align the beams of transmitter and receiver



Brute force to find best: try all possible M_t x M_r pairs of indices

[1] R. Heath, N. Gonzalez-Prelcic, S. Rangan, W. Roh and A. Sayeed, An Overview of Signal Processing Techniques for Millimeter Wave MIMO Systems, 2016

Example of ML-based analog beam selection

Example with
$M_t = M_r = 2$ vectors per codebook

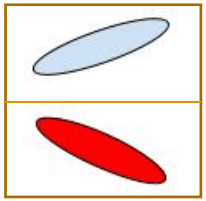
Example:
$\boldsymbol{w}_1$=[0.5, 0.2, -0.4]
assuming 3 antennas at receiver

Tx codebook

Index

j=1

2

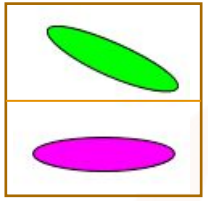


Rx codebook

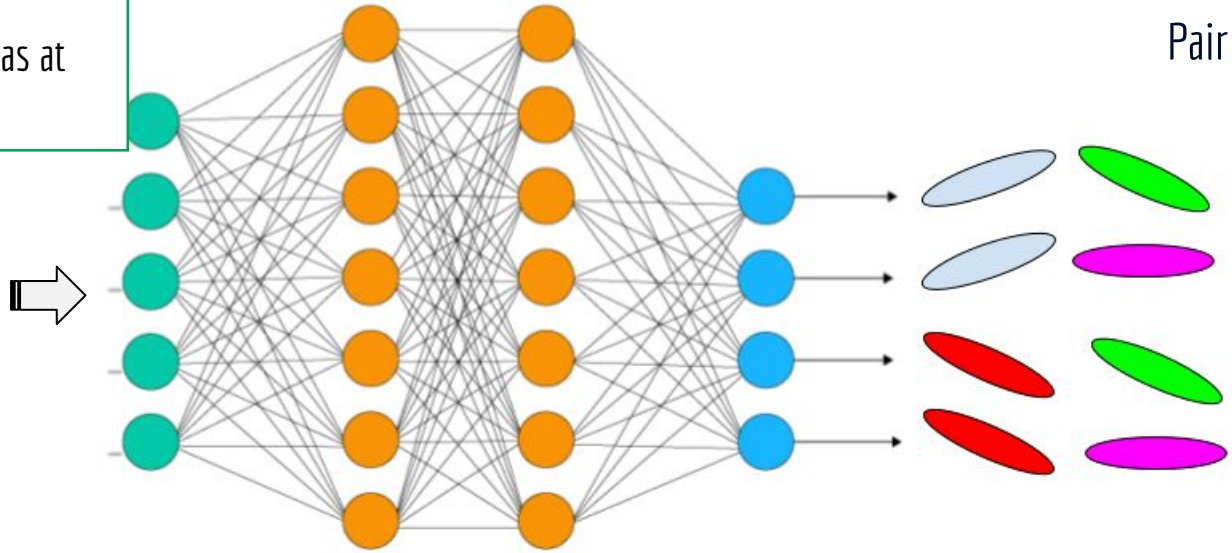
Index

j=1

2



Input



Pair or single index

(1, 1)	0
(1, 2)	1
(2,1)	2
(2,2)	3

Beam Prediction based on Large Language Models

Frame the problem as a LLM-based time series forecasting

System: downlink with base station using *M* antennas to single-antenna users. The figure of merit is the normalized gain:

$$\max_{\hat{q}^\star} G_{\mathrm{N}} = \frac{\left|\mathbf{h}^{\mathrm{T}}\mathbf{f}(\hat{q}^\star)\right|^2}{\left|\mathbf{h}^{\mathrm{T}}\mathbf{f}(q^\star)\right|^2}$$

Experiments: using GPT-2, with a period of 16 ms, use the history of past T=40 steps to predict the future H=10 steps, using a DeepMIMO dataset

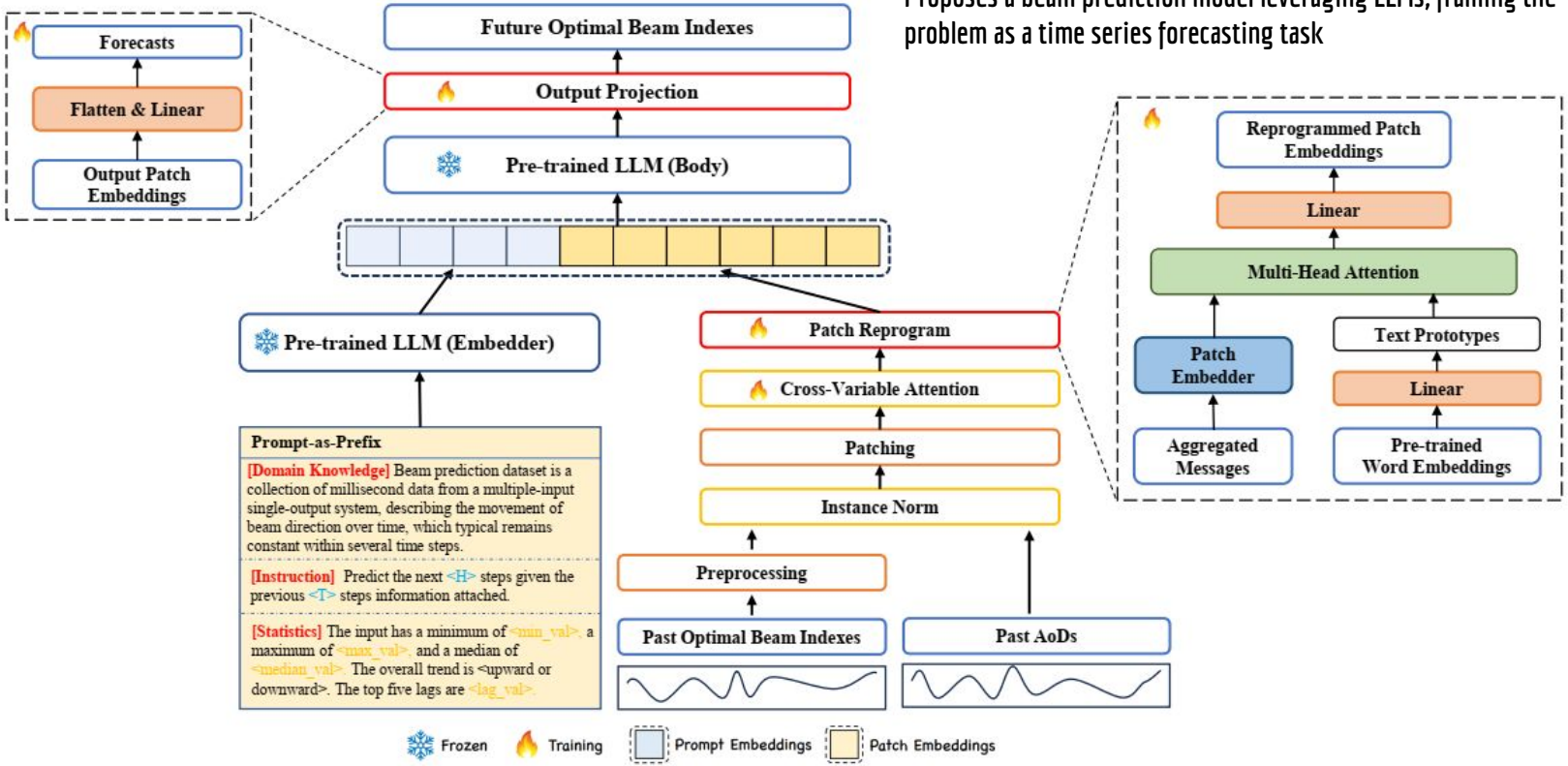
Inputs: are past indices and angle of departure. They are concatenated and the time-series is segmented in "**patches**". Multi-head cross-attention layer is used to align these patches with text, using embeddings representing trends such as "short up" for an upward trend and "steady down" for a downward trend

When the patches are combined, they can represent local information such as "short up then down steadily"

Sheng, Y., Huang, K., Liang, L., Liu, P., Jin, S., & Li, G. Y. (2024). Beam Prediction based on Large Language Models. *arXiv preprint arXiv:2408.08707*.

Beam Prediction based on Large Language Models

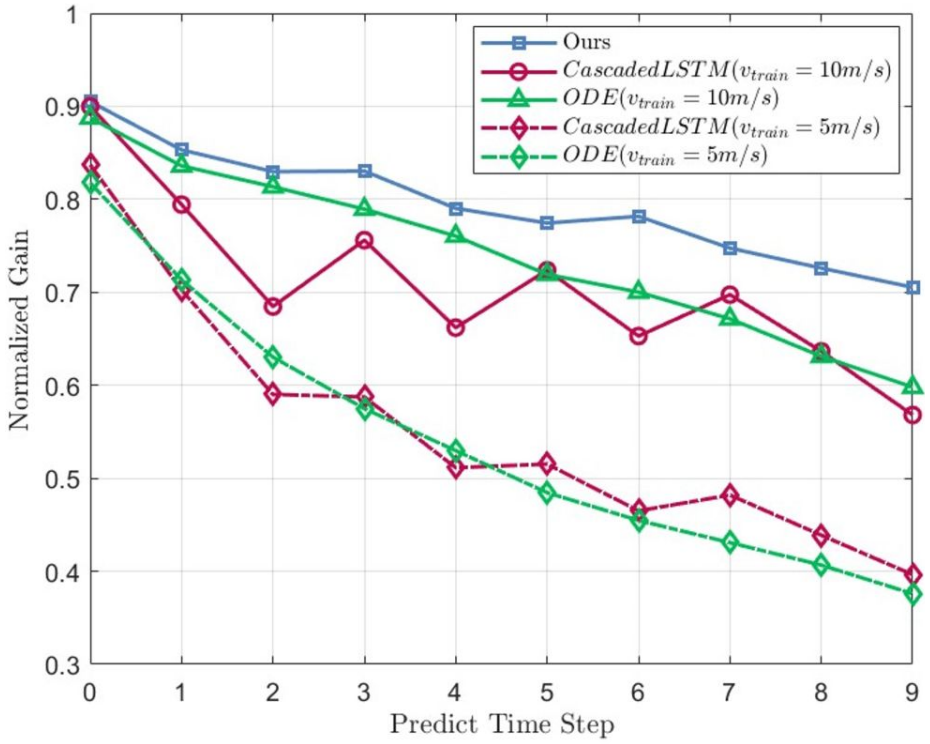
Proposes a beam prediction model leveraging LLMs, framing the problem as a time series forecasting task



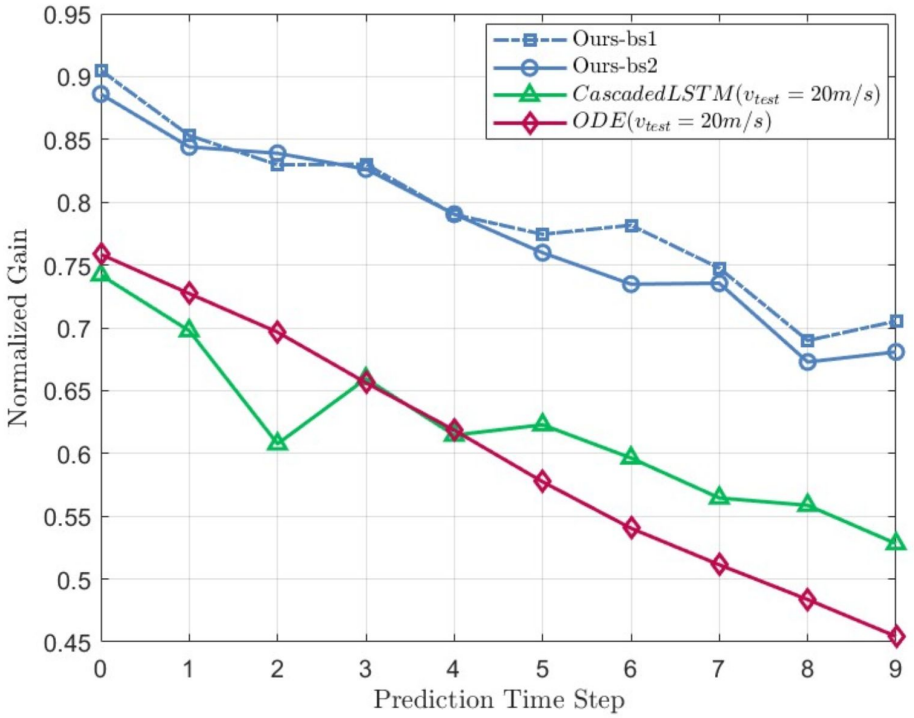
Sheng, Y., Huang, K., Liang, L., Liu, P., Jin, S., & Li, G. Y. (2024). Beam Prediction based on Large Language Models. *arXiv preprint arXiv:2408.08707*.

Results under mismatched conditions

Test with velocity = 20 m/s



Test with "wrong" base station



Conclusions

GenAI in telecom has just started!

There are plenty of opportunities is topics that are inherently “generative” such as channel generation and LLMs applied to the telecommunications domain

It remains to be rigorously proved (proper datasets and evaluation methodologies) that GenAI is the best in tasks such as channel estimation and beam management

Standardized AI, public datasets and benchmarks will play important role, promoting reproducible results and advancing GenAI and AI in telecommunications

To work in GenAI for telecom, a good understanding of the historical development of the main methods is useful. For instance: new chapters of the novel “Discriminative versus Generative” are playing!

Acknowledgements

Our AI work is partially financed by the Innovation Center, **Ericsson** Telecomunicações S.A., Brazil; Universal project (**CNPq** grant 405111/2021-5) and Project Smart 5G Core And MUltiRAn Integration (**SAMURAI**) (MCTIC/CGI.br/**FAPESP** under Grant 2020/05127-2).

Share your thoughts!!

We will be happy to discuss and answer questions about the contents of this tutorial via the e-mail lasse.courses@gmail.com until 11/October/2024

aldebaro@ufpa.br

pedro.sousa@itec.ufpa.br

claudio.barata@itec.ufpa.br

joao.tavares.borges@itec.ufpa.br

felipe.bastos@itec.ufpa.br

cleversonahum@ufpa.br

UFPA

LASSE

Obrigado à UFPA e demais organizadores do SBrT 2024!
Obrigado a você pela participação!
Sucesso nas carreiras!

A SBrT é você!
Inscreva-se no canal!
Participe e torne-se sócio!

SBrT

Extra!

Some useful references

[1] A. Karapantelakis et al., “Using Large Language Models to Understand Telecom Standards,” in 2024 IEEE International Conference on Machine Learning for Communication and Networking (ICMLCN), May 2024, pp. 440–446. doi: 10.1109/ICMLCN59089.2024.10624786.

[2] A. Karapantelakis et al., “A Survey on the Integration of Generative AI for Critical Thinking in Mobile Networks,” Apr. 10, 2024, arXiv: arXiv:2404.06946. doi: 10.48550/arXiv.2404.06946.

[3] A. Karapantelakis et al., “A Survey on the Integration of Generative AI for Critical Thinking in Mobile Networks,” Apr. 10, 2024, arXiv: arXiv:2404.06946. Accessed: Sep. 14, 2024. [Online]. Available: http://arxiv.org/abs/2404.06946

[4] A. Karapantelakis et al., “A Survey on the Integration of Generative AI for Critical Thinking in Mobile Networks,” Apr. 10, 2024, arXiv: arXiv:2404.06946. Accessed: Sep. 14, 2024. [Online]. Available: http://arxiv.org/abs/2404.06946

[5] K. R. M, D. N. Hegde, M. Sarajlic, and A. Sarkar, “Machine Learning (ML)-Assisted Beam Management in Millimeter (mm)Wave Distributed Multiple Input Multiple Output (D-MIMO) Systems,” in 2024 16th International Conference on COMmunication Systems & NETworkS (COMSNETS), Jan. 2024, pp. 457–464. doi: 10.1109/COMSNETS59351.2024.10427216.

[6] M. Xu et al., “Unleashing the Power of Edge-Cloud Generative AI in Mobile Networks: A Survey of AIGC Services,” IEEE Communications Surveys & Tutorials, vol. 26, no. 2, pp. 1127–1170, 2024, doi: 10.1109/COMST.2024.3353265.

[7] H. Du et al., “The Age of Generative AI and AI-Generated Everything,” IEEE Network, pp. 1–1, 2024, doi: 10.1109/MNET.2024.3422241.

[8] A. Celik and A. M. Eltawil, “At the Dawn of Generative AI Era: A Tutorial-cum-Survey on New Frontiers in 6G Wireless Intelligence,” IEEE Open Journal of the Communications Society, vol. 5, pp. 2433–2489, 2024, doi: 10.1109/OJCOMS.2024.3362271.

[9] L. Bariah, Q. Zhao, H. Zou, Y. Tian, F. Bader, and M. Debbah, “Large Generative AI Models for Telecom: The Next Big Thing?,” IEEE Communications Magazine, pp. 1–7, 2024, doi: 10.1109/MCOM.001.2300364.

[10] L. Bariah, H. Zou, Q. Zhao, B. Mouhouche, F. Bader, and M. Debbah, “Understanding Telecom Language Through Large Language Models,” in GLOBECOM 2023 - 2023 IEEE Global Communications Conference, Dec. 2023, pp. 6542–6547. doi: 10.1109/GLOBECOM54140.2023.10437725.

[11] P. Eigenschink, T. Reutterer, S. Vamosi, R. Vamosi, C. Sun, and K. Kalcher, “Deep Generative Models for Synthetic Data: A Survey,” IEEE Access, vol. 11, pp. 47304–47320, 2023, doi: 10.1109/ACCESS.2023.3275134.

[12] J. Liu, M. Nogueira, J. Fernandes, and B. Kantarci, “Adversarial Machine Learning: A Multilayer Review of the State-of-the-Art and Challenges for Wireless and Mobile Systems,” IEEE Communications Surveys & Tutorials, vol. 24, no. 1, pp. 123–159, 2022, doi: 10.1109/COMST.2021.3136132.

[13] Z. Chen, Q. He, L. Liu, D. Lan, H.-M. Chung, and Z. Mao, “An Artificial Intelligence Perspective on Mobile Edge Computing,” in 2019 IEEE International Conference on Smart Internet of Things (SmartIoT), Aug. 2019, pp. 100–106. doi: 10.1109/SmartIoT.2019.00024.

[14] HO, Jonathan; JAIN, Ajay; ABBEEL, Pieter. Denoising diffusion probabilistic models. Advances in neural information processing systems, v. 33, p. 6840-6851, 2020.
FESL, Benedikt et al. Diffusion-based Generative Prior for Low-Complexity MIMO Channel Estimation. arXiv preprint arXiv:2403.03545, 2024.

[15] TSE, David; VISWANATH, Pramod. Fundamentals of wireless communication. Cambridge university press, 2005.