國立臺灣大學
National Taiwan University

Foundation Models for Communication Systems

Davide Buffelli, Chu-Hsiang Huang, Sattar Vakili, Da-shan Shiu

IEEE Global Communications Conference (GLOBECOM) 2025

Presenters

Dr Davide Buffelli

- Senior AI Researcher at MediaTek Research, working on optimization, large language models, and AI applications for communication systems.
- PhD from the University of Padova, on deep learning methods for graph structured data.

Professor Chu-Hsiang Huang

- Assistant professor at National Taiwan University (NTU) working on next-generation wireless communication system design, communication system standardization, artificial intelligence and machine learning, and statistical communication theory.
- Ph.D. degree in Electrical Engineering from the University of California, on Iterative Information Processing on Unreliable Hardware

Material

Link:
https://github.com/mtkresearch/foundation_models_comms_Globecom2025

Contents

Part 1

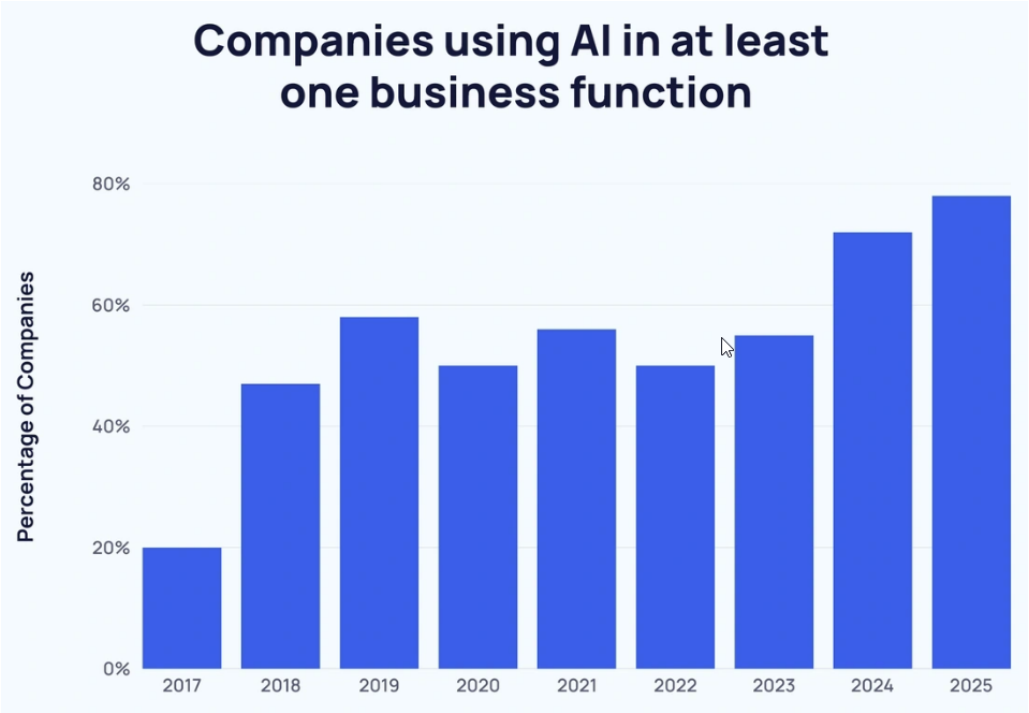
Introduction

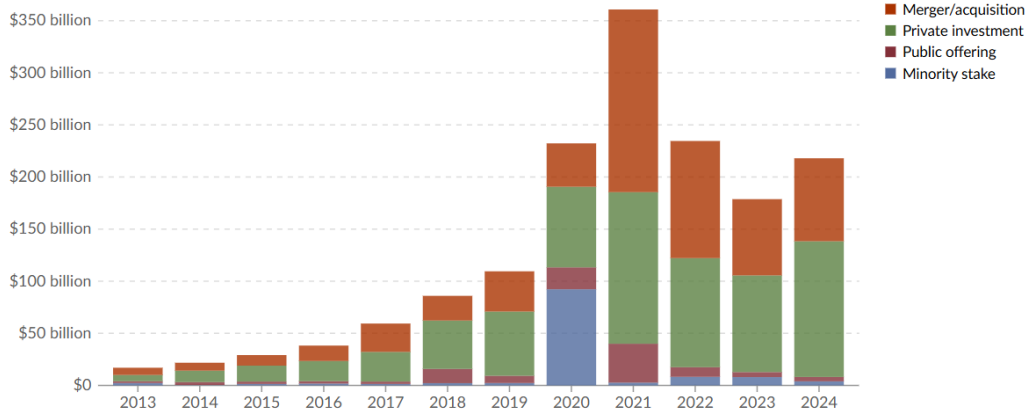
- Current state of AI
- AI in communication systems
- Objectives of this tutorial
- Overview of the topics of this tutorial

AI's Recent Growth

- 2012 AlexNet – deep learning model training on GPUs on large amounts of data outperforms complex traditional methods
- 2013 Word2Vec – first big success of deep learning in natural language processing, outperforming existing approaches
- 2014 Generative Adversarial Networks – first model able to generate new images with high quality
- 2017 Transformer – deep learning models become the standard for tasks like machine translation
- 2018 GPT – first model able to generate new text
- Present: LLMs, Agentic systems, specialised chips – AI adoption is growing rapidly

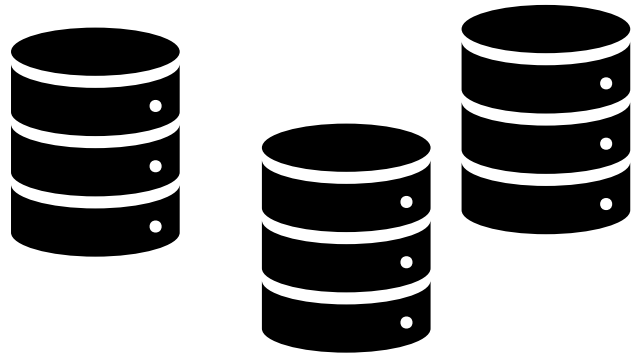
https://explodingtopics.com/blog/ai-statistics





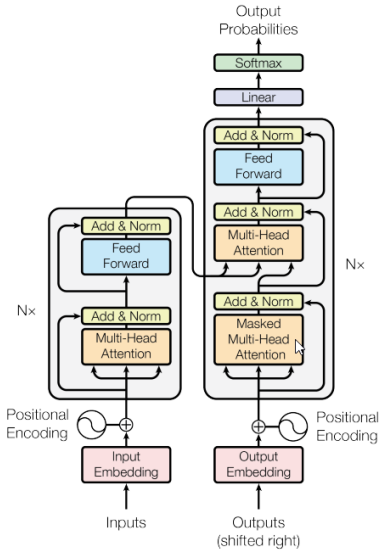
https://ourworldindata.org/ai-investments

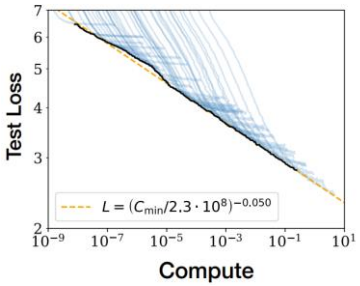
Behind the growth

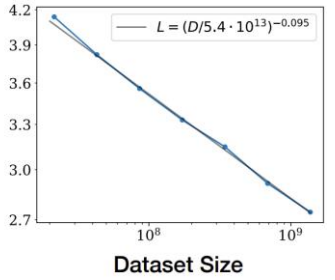


Convergence of several key factors

- Massive data availability
  - Models like ChatGPT, DeepSeek, Qwen, etc. are trained on internet scale amounts of data
- Scalable architectures
  - Tranformers were the first architecture to be designed with a focus on efficiency, and with studied scaling laws
- Specialized hardware
  - GPUs, TPUs, etc. An increasing amount of hardware (and surrounding software infrastructure) is being specifically designed for AI
- Industry investment
  - The global AI market is valued at approximately $391 billion







Data Explosion

Large datasets in 2012

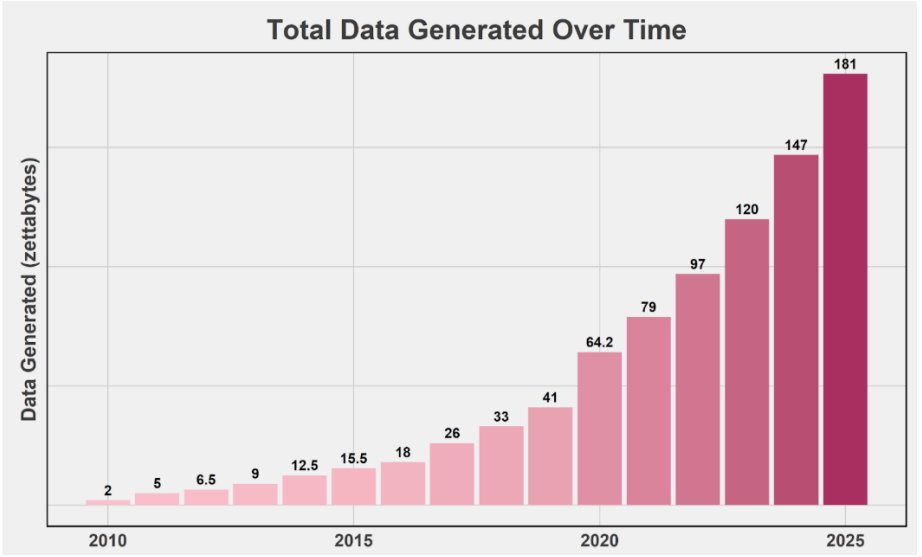
- ImageNet: 1.2M labelled images
- Reuters Corpus Volume 1, 6-7 GB text data
- CSR-I (WSJ0), 141 hours of audio clips
- UCF101 ~13,000 videos

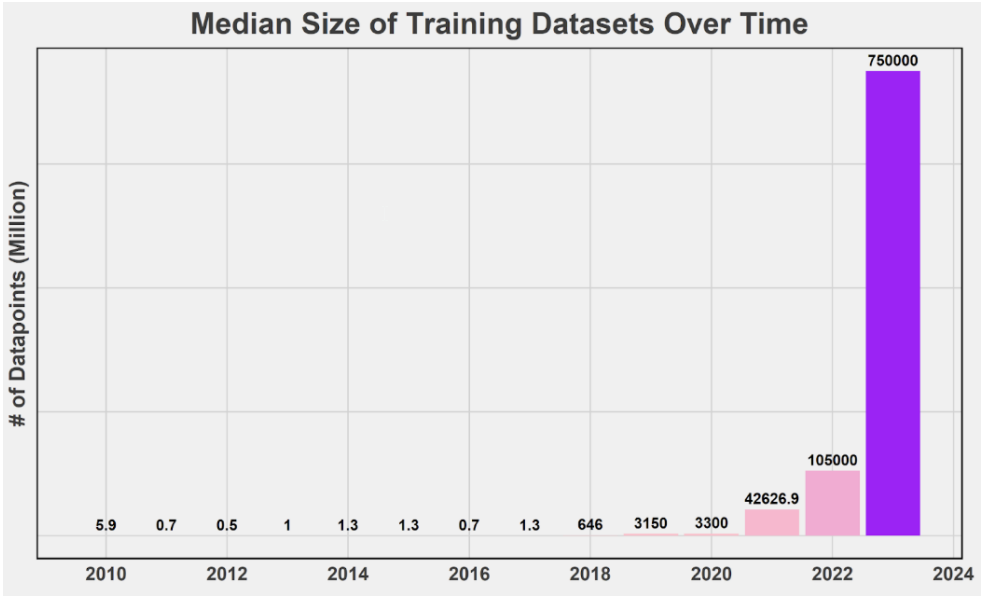
Large datasets in 2025

- LAION-5B: 5B images scraped from the web
- Common Crawl: ~60TB text data
- Common Voice thousands of hours, millions of audio clips
- YouTube-8M, 8 million videos

https://futuretech.mit.edu/news/what-drives-progress-in-ai-trends-in-data

The amount of available data has exploded. In just 10 years, from 2010 to 2020, the total amoun of new data generated per year grew 32x, from 2 zettabytes created in 2010 to over 64 zettabyte created in 2020 alone. Figure 2 draws on the Statista research dataset to illustrate this.

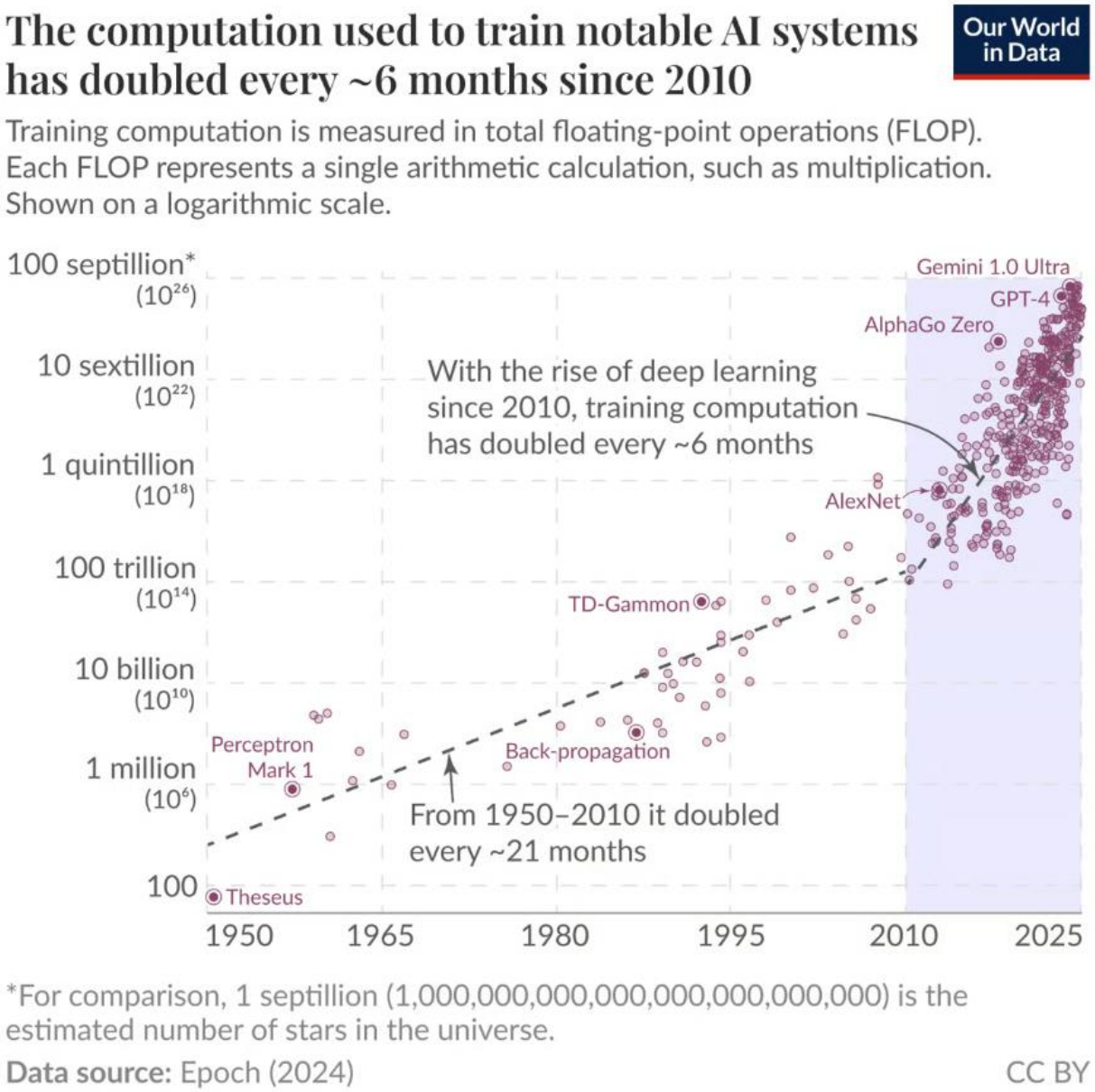




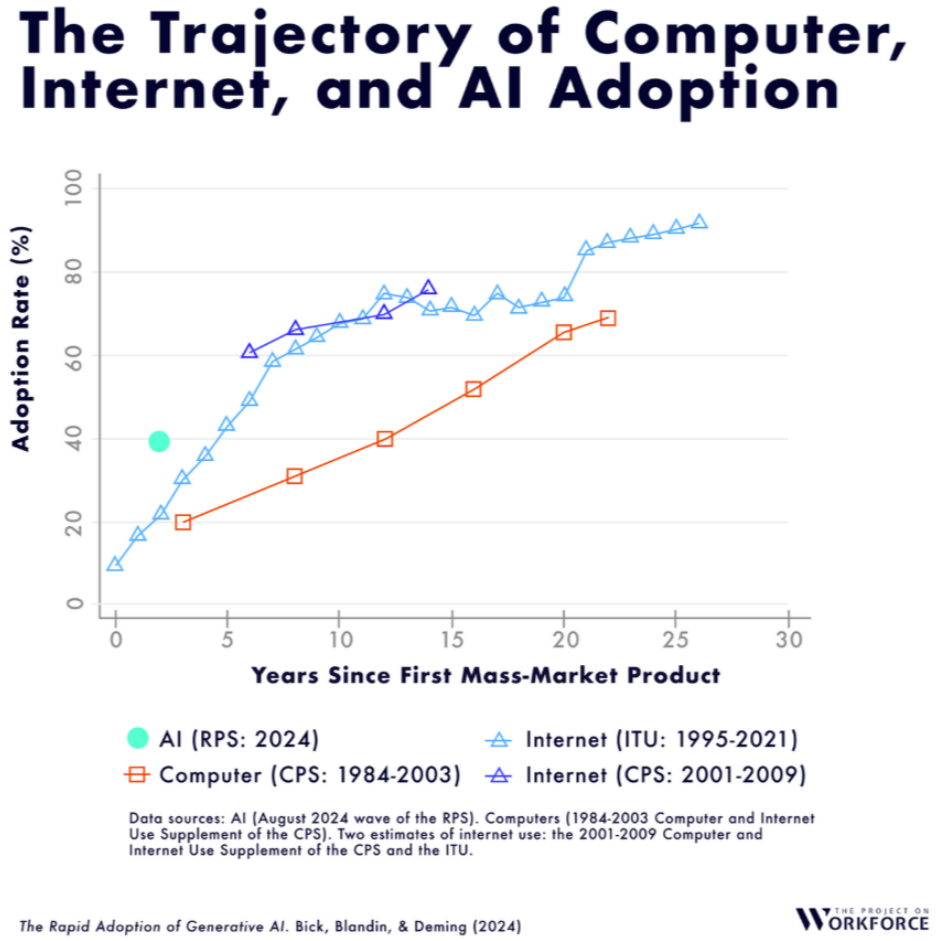
Compute Explosion

- 2012 - AlexNet: ~7x10^15 FLOPs training
- 2020 - GPT-3: ~3.14×10^{23} FLOPs
- 2024 - Training GPT-4 estimated 5e25 FLOPs training

https://ourworldindata.org/data-insights/since-2010-the-training-computation-of-notable-ai-systems-has-doubled-every-six-months

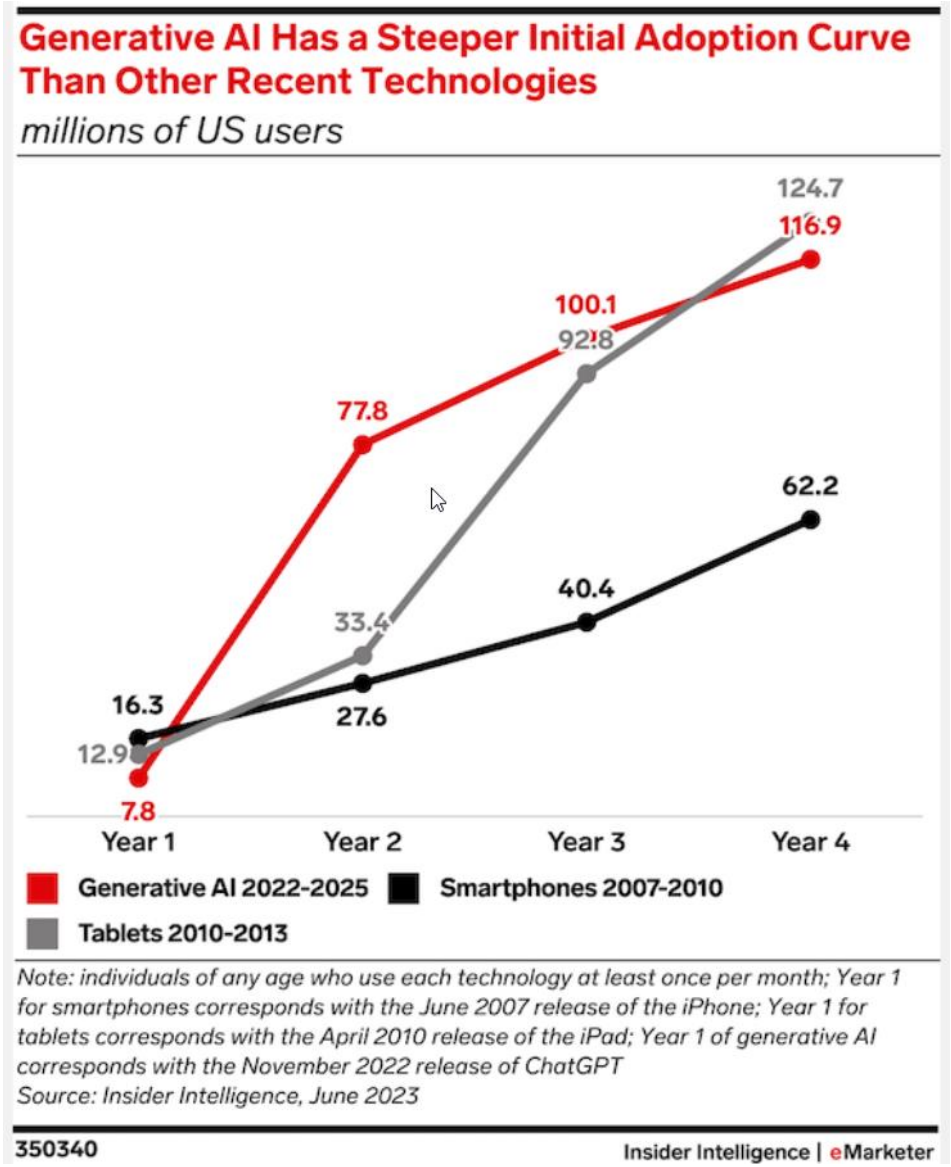


Public Adoption



https://www.pw.hks.harvard.edu/post/the-rapid-adoption-of-generative-ai

https://www.emarketer.com/content/generative-ai-adoption-climbed-faster-than-smartphones-tablets



The Foundation Model Paradigm Shift

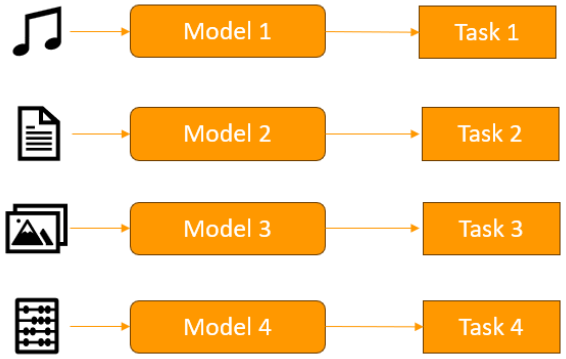
Traditional AI Workflow

- Labelled Dataset → model → task-specific training → deployment

- Classic supervised learning approach

Limitations

- Requires labelled (expensive) data
- Poor generalization to new tasks
- Expensive retraining for each scenario



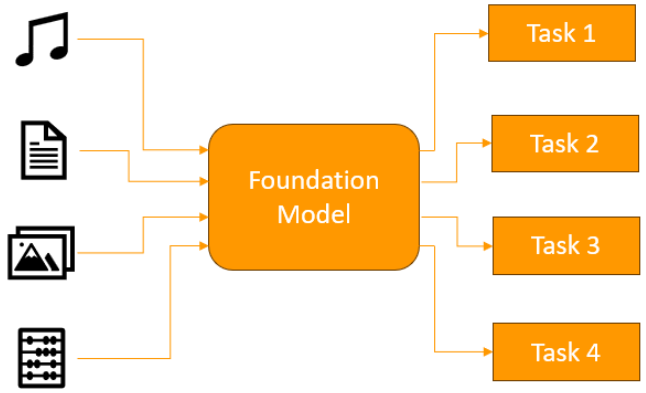
Foundation Model Workflow

- Huge unlabelled Dataset → model → task-agnostic training → deployment

- Foundation models are trained once, used many times

Properties

- Train on broad, unlabeled (cheap) data.
- Adaptable to many downstream tasks



Example: Large Language Models (LLMs)

- LLMs are a Foundation model for textual data
- Trained on vast amounts of text data
  - Books, magazines, websites, articles, technical reports, papers, etc.
- Adaptable to new tasks with minimal additional training
  - Simply prompt the model to perform the task
- Can be fine-tuned/aligned to specific specialized tasks with minimal labelled data
  - *Requires access to weights of the model

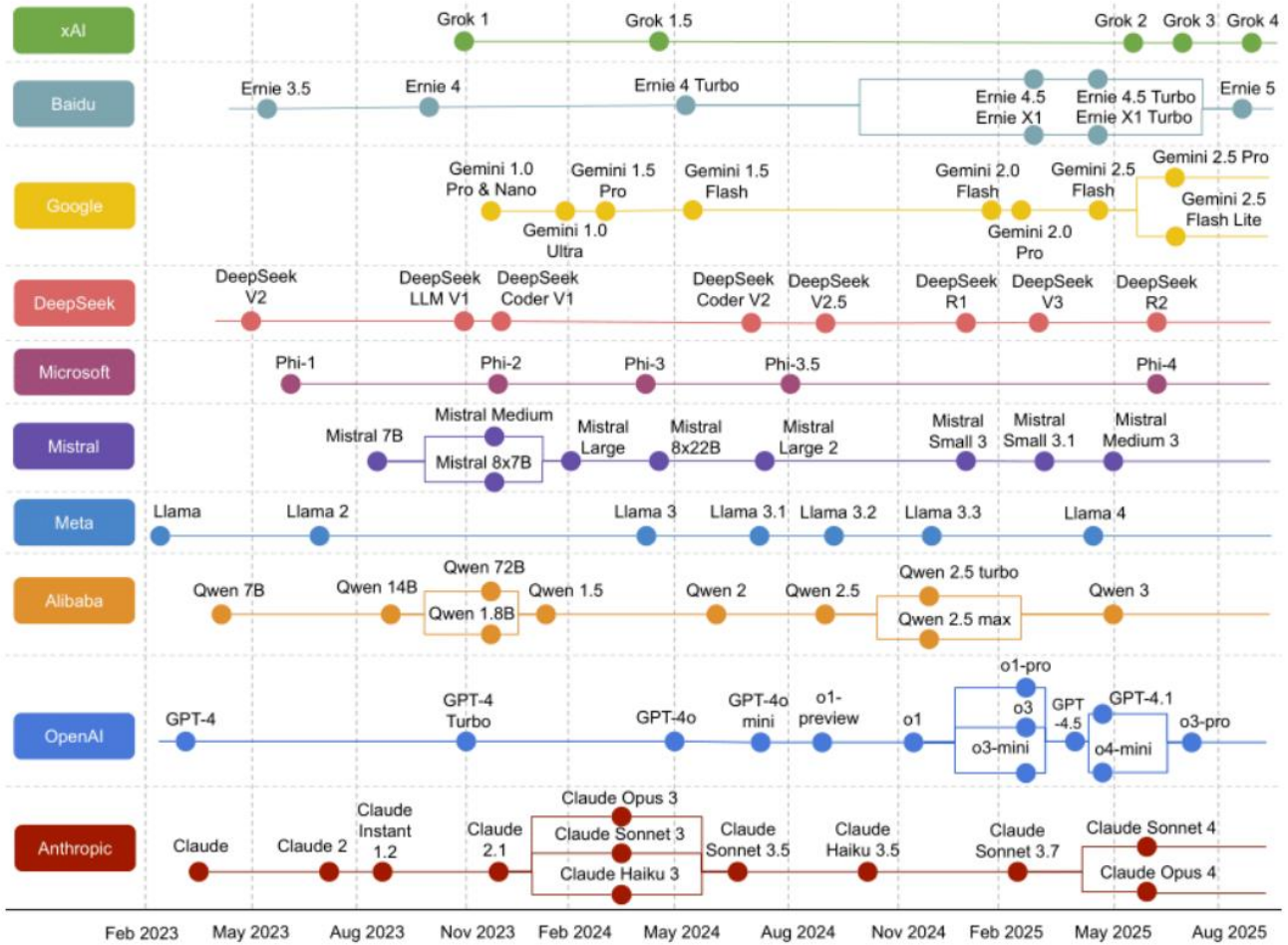


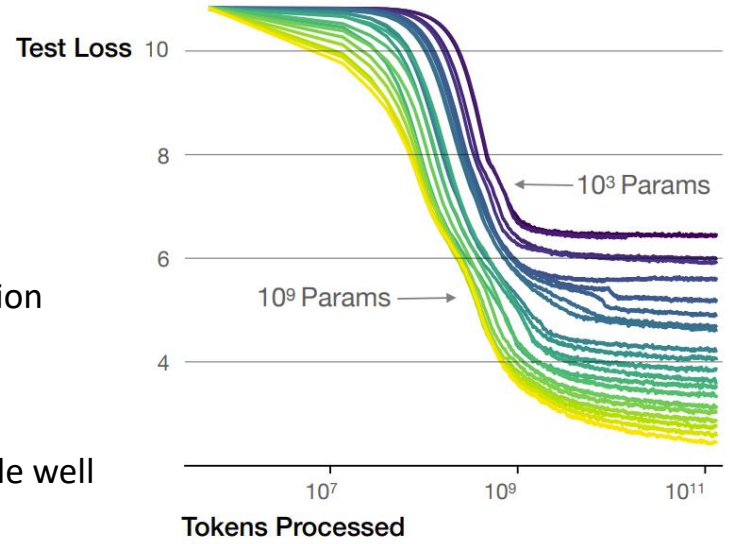
Image from “How Well Do LLMs Predict Prerequisite Skills? Zero-Shot Comparison to Expert-Defined Concepts“, Luyen et al., 2025

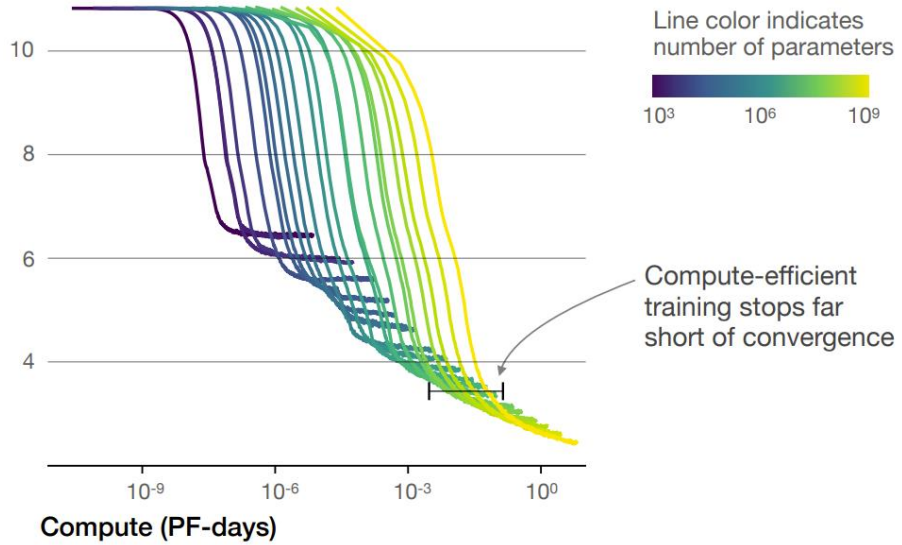
Scaling Laws

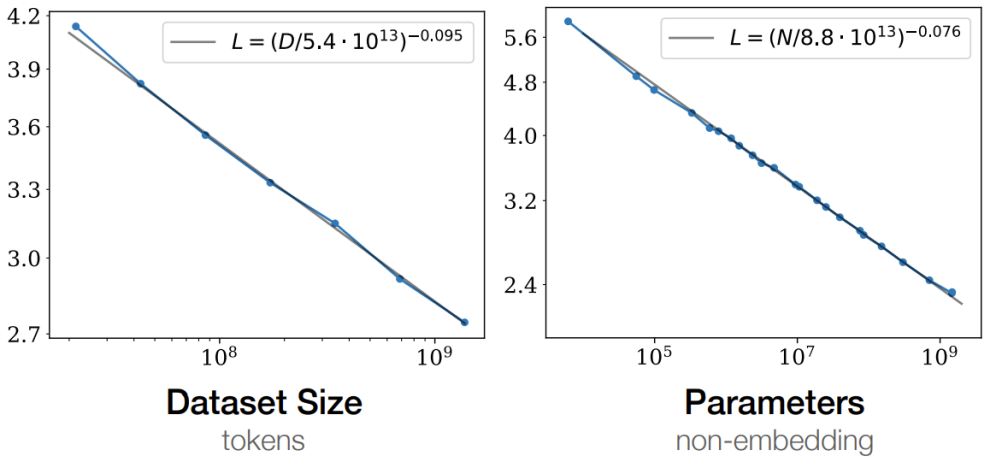
- A key motivation behind the shift to the Foundation Model paradigm is given by scaling laws

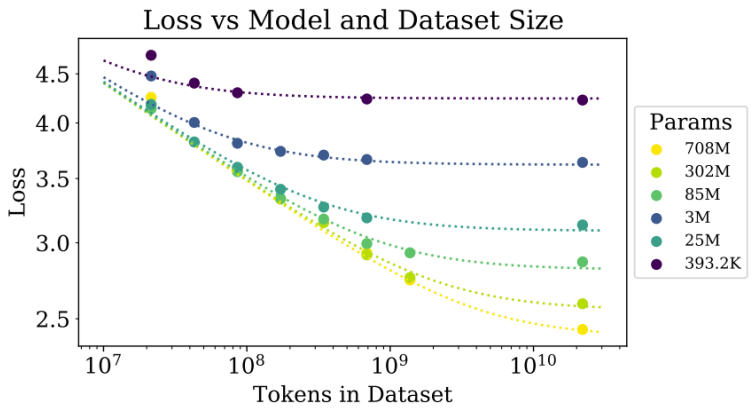
- Empirically, Foundation Models are shown to scale well with more data, compute, and parameters

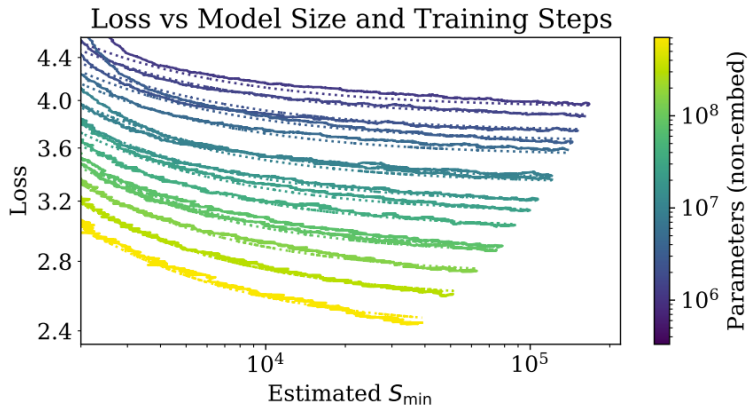
- TLDR: the bigger the better







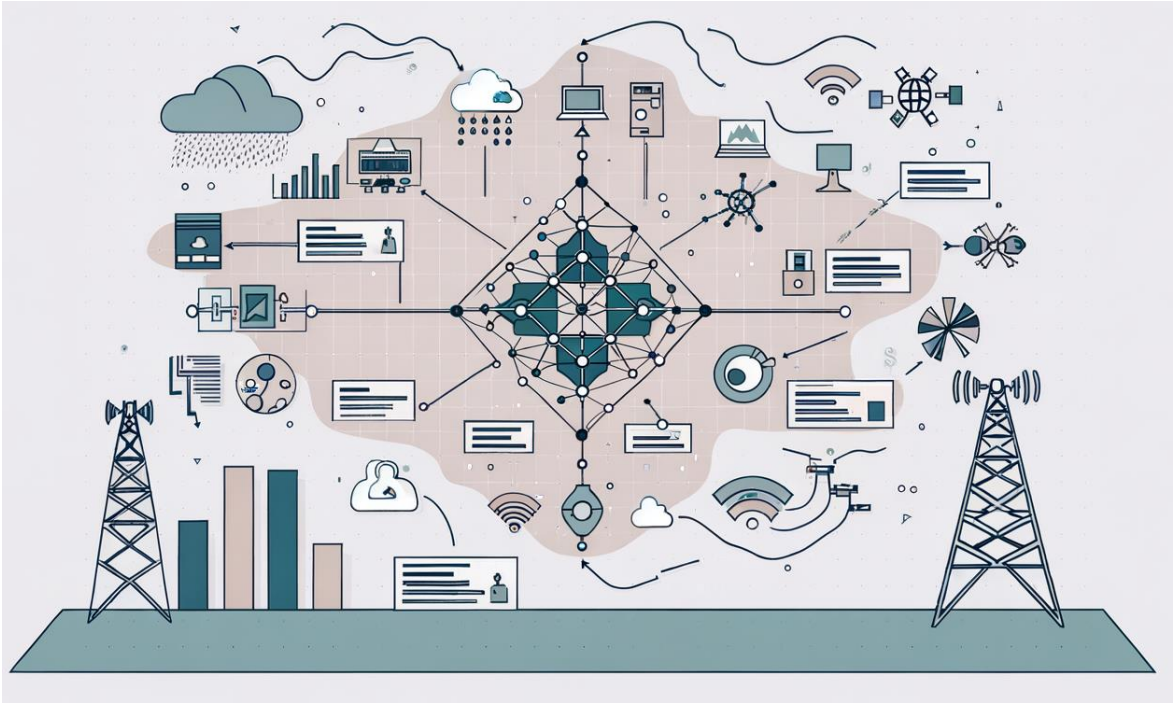




"Scaling Laws for Neural Language Models", Kaplan et al., 2020

From Task-Specific AI to Foundation Models: A New Paradigm for Communication Systems

- AI is transforming every industry
  - New paradigm: from task-specific models to Foundation Models
- AI has entered Communication Systems
  - It is now entering into the Foundation Model Paradigm
- This tutorial will bridge AI's latest advances with communication applications
  - How to design a foundation model for communication data
  - Possible applications
  - Open challenges
  - Future directions



AI generated image

AI in Communication Systems

- Modern communication systems require numerous estimation and optimization tasks
- Deep Learning methods are based on the idea of learning from data
- Current research has focused on designing task-specific solutions

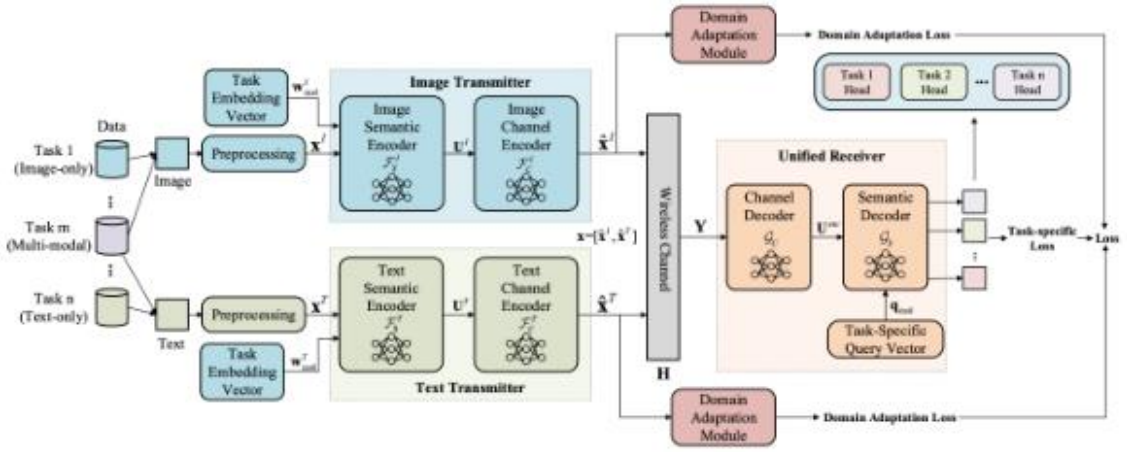


Image from "A Unified Multi-Task Semantic Communication System with Domain Adaptation", G. Zhang, Q. Hu, Z. Qin, Y. Cai and G. Yu, GLOBECOM 2022

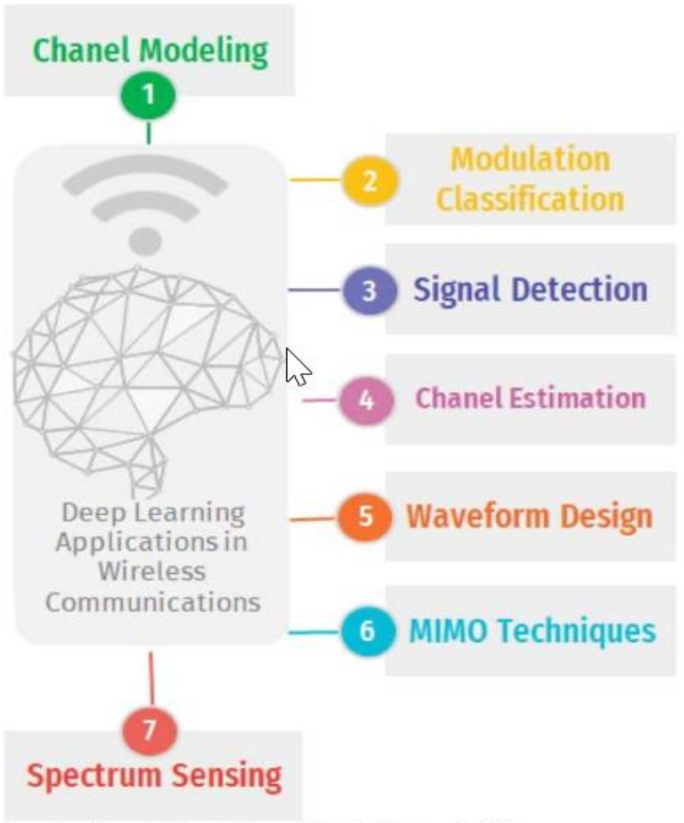


Image from "Towards Evaluating Adversarial Attacks Robustness in Wireless Communication", Fatima and Mazri, (IJACSA) 2021

AI in Communication Systems

- Modern communication systems require numerous estimation and optimization tasks
- Deep Learning methods are based on the idea of learning fr
- Current research has focused on designing task-specific solu

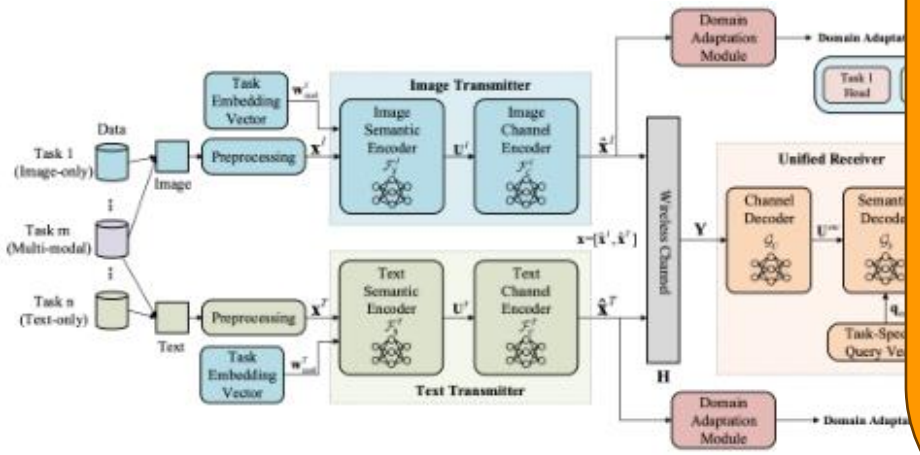
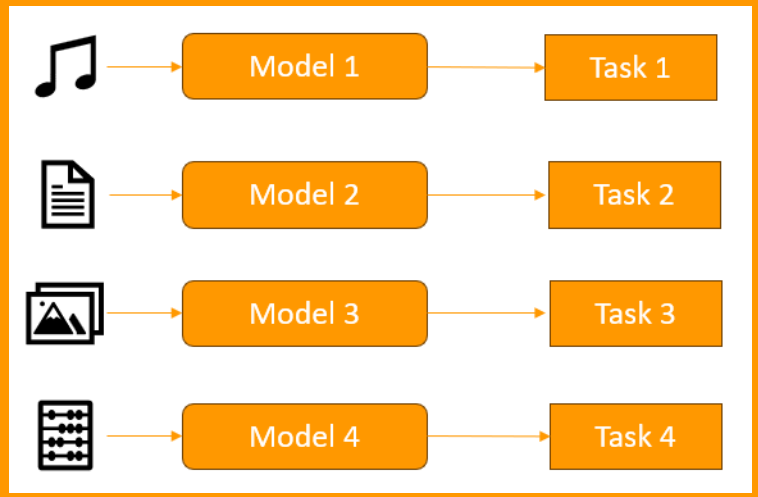


Image from "A Unified Multi-Task Semantic Communication System with Domain Adaptation", G. Zhang, Q. Hu, Z. Qin, Y. Cai and G. Yu, GLOBECOM 2022

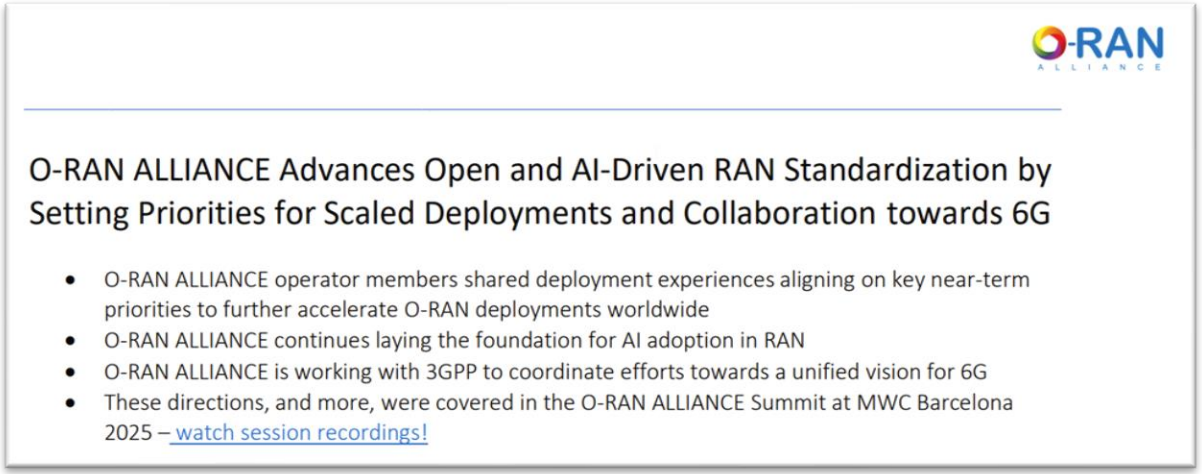
Chanel Modeling

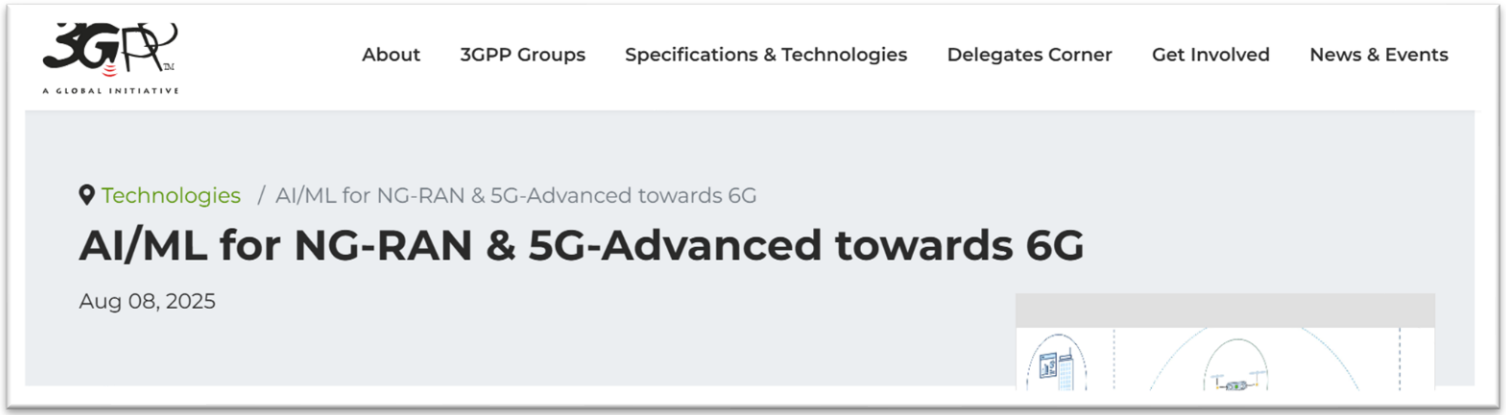
The area of communication systems is mainly still in the "standard supervised learning" phase



Foundation Models in Communication Systems

- Communication systems are entering the Foundation Model era
  - Current AI models applied to communication systems require ad-hoc design and data
  - Some recent works are exploring moving to the Foundation model paradigm
- Foundation models have a huge potential in communication systems
  - Single model for multiple tasks
  - Lower data labelling costs
- Still in a very preliminary phase with many unexplored directions!





Objective of this Tutorial

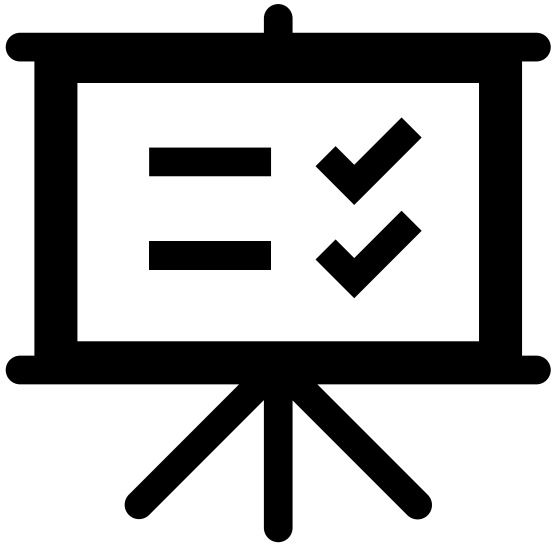
Our goal is to encourage the community to explore the area of Foundation Models applied to communication systems

- Overview of background knowledge on Machine Learning and Deep Learning

- Presentation of fundamentals of Foundation Models
  - Design
  - Training
  - Data
  - Applications

- Showcase recent state-of-the-art studies in this area

- Highlight open challenges and possible future directions

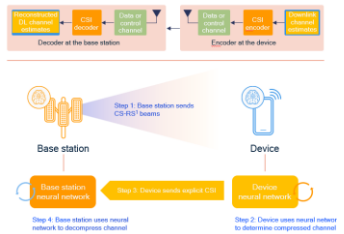


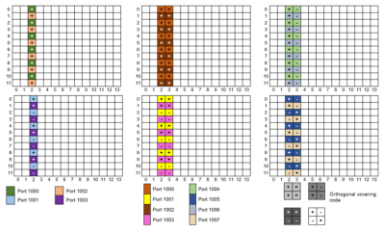
Organization (1)

Part 2

Communication System and Channel Mode

- 5G mmWave systems
- Channel Estimation
- Channel State Information
- 3GPP AI/ML air-interface

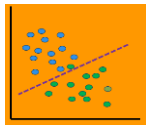


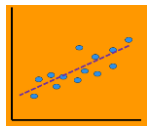


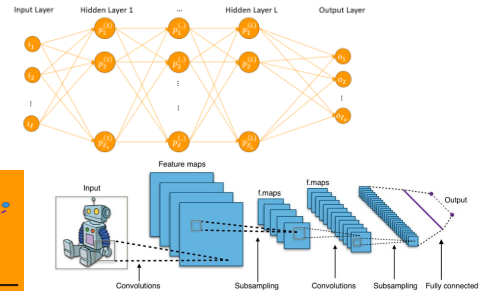
Part 3

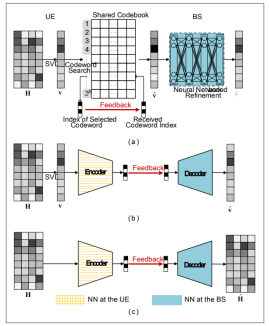
Introduction to Deep Learning

- Machine Learning basics
- Deep Learning basics
  - Multi-Layer Perceptron
  - Overview of Popular Architectures
  - Model Training
- Application Example
  - Deep Learning for CSI Feedback







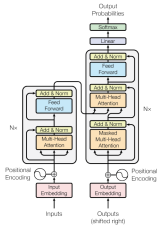


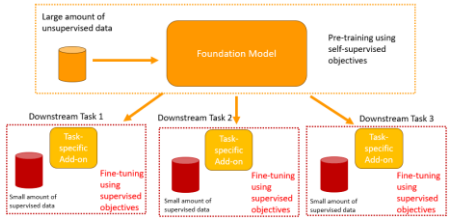
Organization (2)

Part 4

Foundation Model Design

- The Transformer Architecture
  - Tokenization
  - Positional encodings
  - Attention
- Foundation Model Paradigm
  - Input design
  - Multi-modal transformers
  - Training procedures

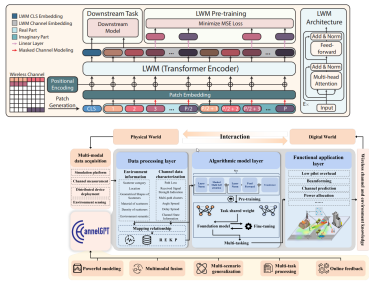


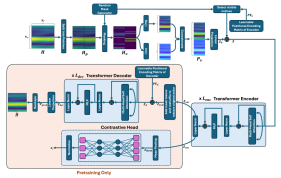


Part 5

Case Studies and Experimental Evaluation

- Analysis of 7 state-of-the-art approaches
  - Data design
  - Architecture design
  - Training procedure
  - Inference procedure
  - Downstream evaluations
  - Experimental results

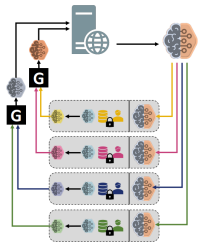




Part 6

Future Directions and Open Challenges

- Large scale datasets
- Standardized benchmarkc
- On-device inference
- Federeated Learning
- AI Air interface



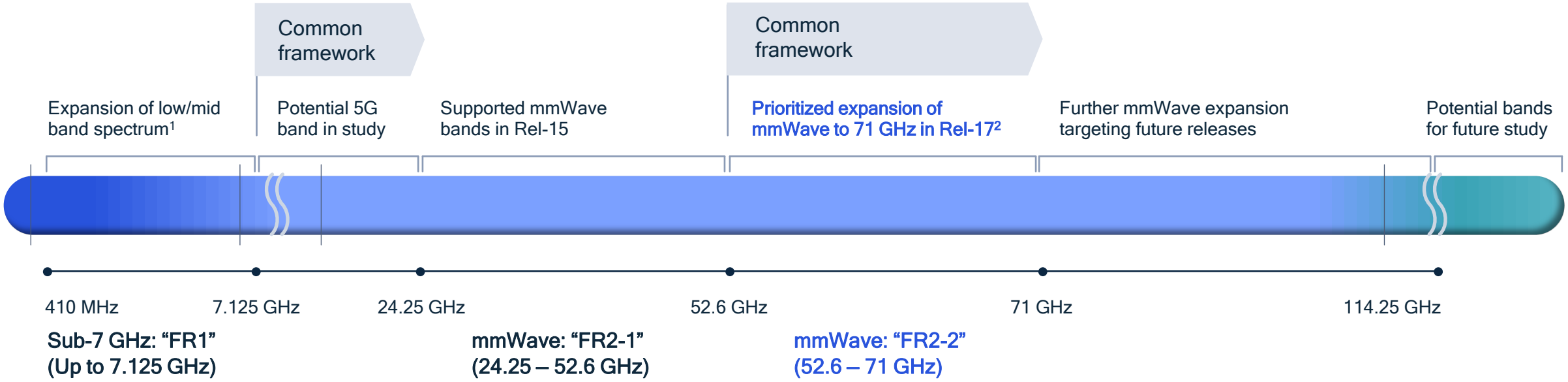
Part 2

Communication System and Channel Mode

- 5G mmWave systems
- Reference signal and channel estimation
- Channel State Information (CSI)
- 3GPP AI/ML air-interface design and use cases

Spectrum beyond 4G: Millimeter Wave

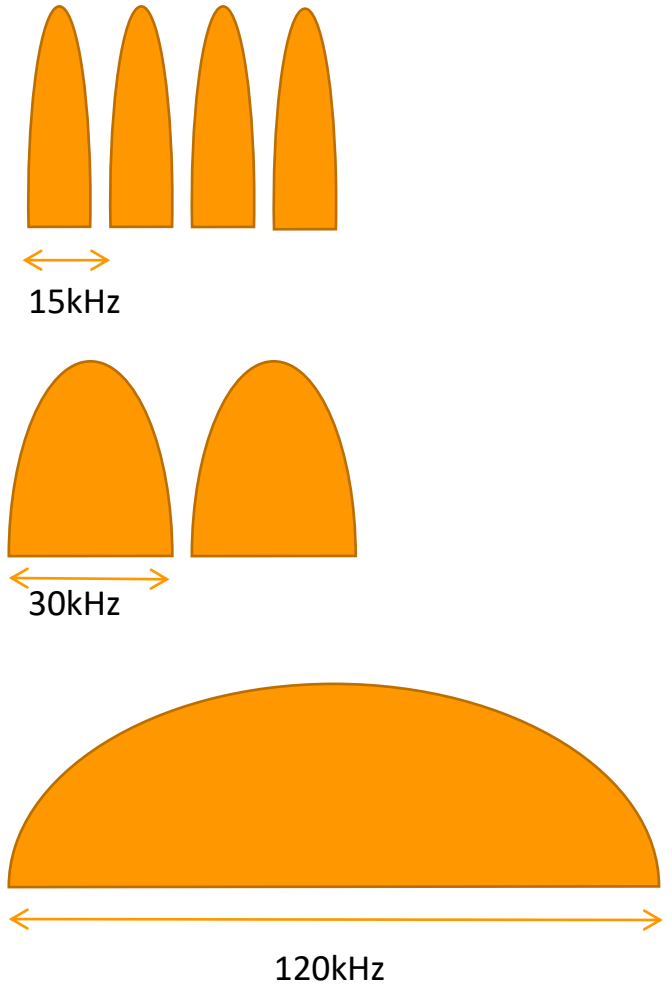
- 5G explores higher frequency bands and faces additional challenges
  - Higher center frequency leads to higher Doppler shift (in the same speed)
  - Inferior link budget due to higher pathloss and more sensitive to phase noise



Scalable Numerologies in OFDM System

- Subcarrier spacing in OFDMC system: to maintain the orthogonality of subcarriers and avoid inter-carrier interference
  - –Inter-symbol interference is guarded by cyclic-prefix
- Multiple options for subcarrier spacing are supported in 5G NR to efficiently address 5G diverse spectrums, deployment scenarios and different service applications

	SCS	Frequency Ranges
FR1	15, 30, 60kHz	Up to 7.125GHz
FR2-1	60, 120, 240kHz	24.25-52.6GHz
FR2-2	120, 480, 960kHz	52.6-71GHz

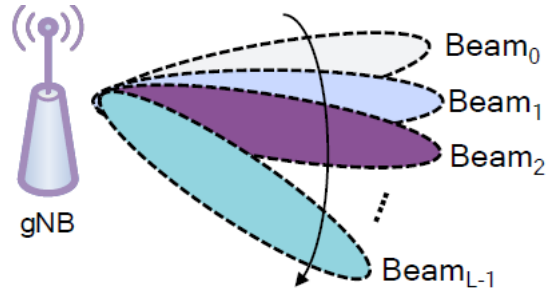


Beam Management

- Antenna array to conquer the pathloss
  - –Large antenna array can fit into a small form factor due to smaller wavelength
  - –Beamforming on more antenna elements can overcome pathloss and lower receive power in mmW
  - –Design considerations
    - Number of elements in horizontal and vertical directions
    - Antenna spacing
    - Precoder codebook design: adjust phase and amplitude to achieve a better antenna gain in certain directions
- Beam management
  - Beamforming is adjusting phase and amplitude of antenna elements to achieve coherent combining when transmitting/receiving a signal to/from **a certain direction**
  - Beam sweeping: sweep wide beams to cover the entire area before knowing device's (UE) direction
  - Beam refinement: refine to narrower beams after locate UEs on wide beams
  - Beam switching: switch to the best beam when UE is moving

SSB Transmission and Beam Sweeping

- SSB burst
  - –Each SSB in the burst has an index to help UE identifying the associated beam
- SSB burst period
  - –Configurable from 5 to 160ms
  - –UE assumes 20ms in initial acquisition
- Beam sweeping
  - –Allows gNB to cover entire cell area by sweeping beams in SSB transmission
  - –Maximum number of SSBs in a burst depends on th carrier frequency



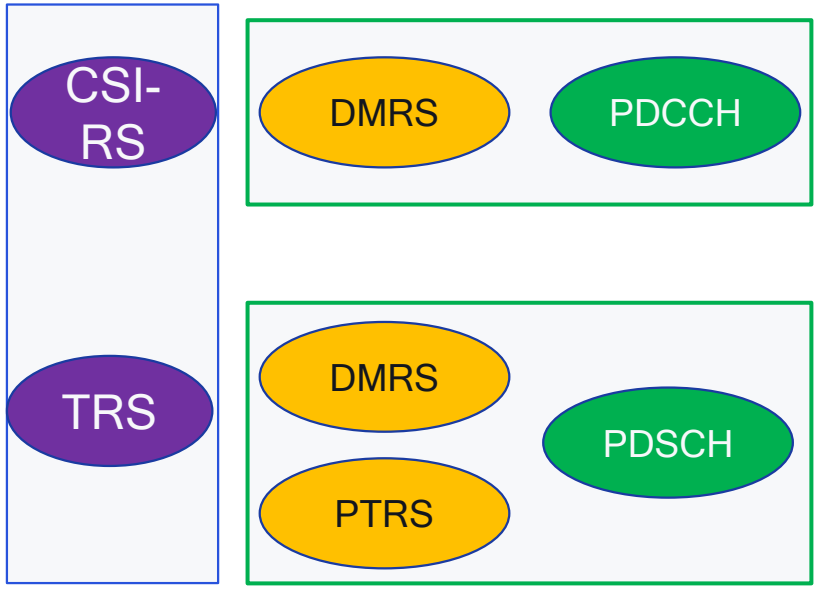
5G DL Reference Signals and Physical Layer Signals

- 5G has a more flexible reference signal scheduling framework
- SS Block
  - –PSS/SSS
  - –DMRS
  - –PBCH
- Control Channel
  - –DMRS
  - –PDCCH
- Data Channel
  - –DMRS
  - –PTRS
  - –PDSCH
- Other reference signals
  - –CSI-RS
  - –TRS

Initial acquisition /Beam management

Beam management /CSI report /tracking

Data transmission



Always on CRS is replaced by different RSs for different purposes

Channel Estimation

- We estimate the channel through the pilots/reference signals to provide channel information for the demodulation/decoding blocks.
- Received signal model in multipath in time domain

$$y(t) = \sum_{\tau} p(\tau)h(t-\tau)$$

where $p(\tau)$ is the reference signal.

- The frequency domain signal after FFT is

$Y(f) = P(f)H(f)$, where $H(f) = \sum_t h(t)e^{-2\pi ft}$

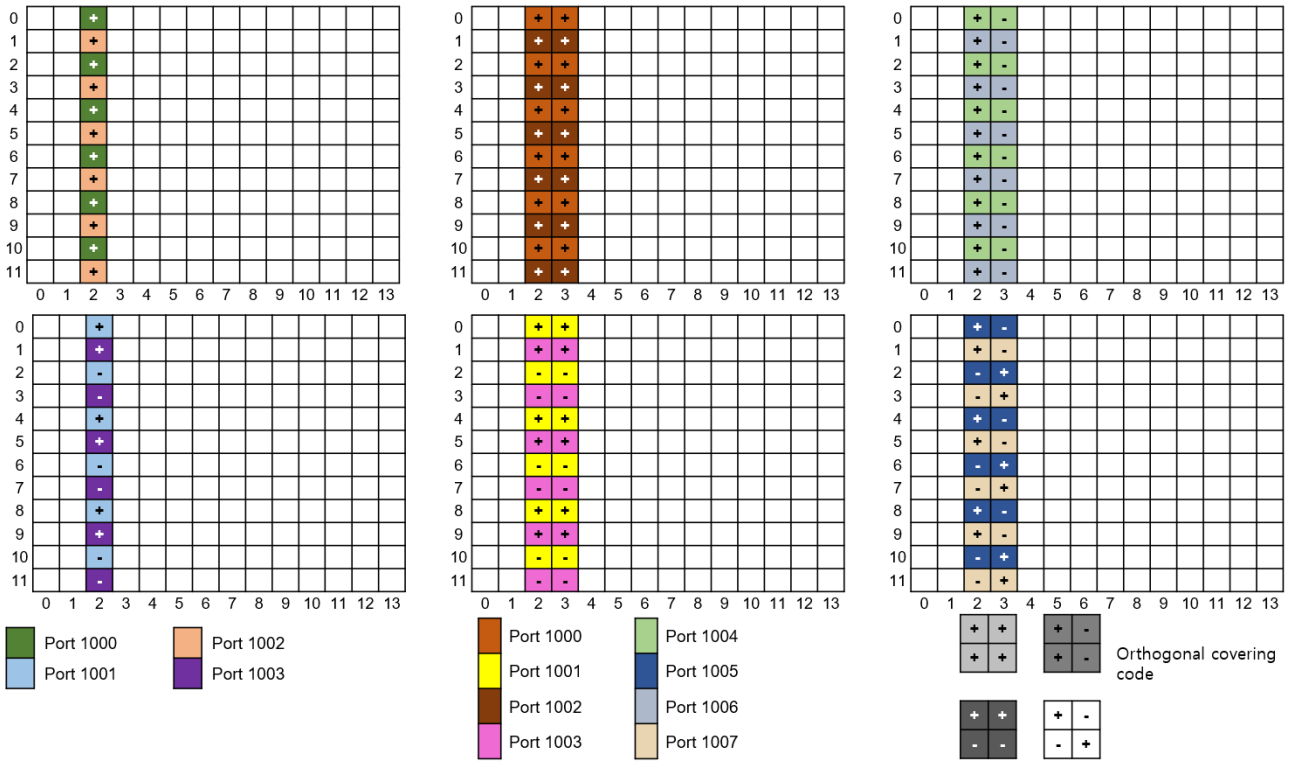
when we sample $H(f)$ at nf_0, the IFFT of sampled $H(f)$ repeated in time domain as

$$\sum_n h(t)\delta(n\frac{1}{f_0} - t)$$

As long as $t > \frac{1}{f_0}$, we get the correct channel $h(t)$ in time domain

- We use interpolation or extrapolation through different techniques to estimate the channels on non-pilot REs

Reference signal (DMRS) patterns for 5G without additional symbols in time domain



CSI Feedback and Precoding with ML

- In MIMO system, we can “diagonalize” the channel by precoding derived by SVD

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x};\ \ \boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T; \boldsymbol{U}^T\boldsymbol{y} = \boldsymbol{U}^T\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T(\boldsymbol{V}\boldsymbol{x}) = \boldsymbol{\Sigma}\boldsymbol{x}$$

- Precoder matrix indication (PMI): Receiver can estimate the channel to get $\boldsymbol{U}^T$, but transmitter needs to know $\boldsymbol{V}$ to transmit $\boldsymbol{V}\boldsymbol{x}$
  - How do transmitter get $\boldsymbol{V}$ with low overhead?
  - We can have a fixed options of precoder, and the receiver report the one that is closest to $\boldsymbol{V}$
  - Alternatively, ML can find good representations of the $\boldsymbol{V}$ under a given overhead constraint
- Channel quality indication (CQI): Ideally, transmitter also wants to know $\boldsymbol{\Sigma}$ to decide the MIMO channel capacity on each layer
  - In practice, we have limited options of code rate and modulation order combination to achieve certain spectrum efficiency
  - Therefore, instead of sending $\boldsymbol{\Sigma}$ back, the receiver sends the suggestions of code rate and modulation order to the transmitter, that is CQI
  - ML can be applied to predict the CQI

3GPP AI/ML Study Item Objectives

- Most of the AI/ML application to wireless communications are implementation based solution leveraging existing air-interface design
  - –The protocol/air-interface and physical signal design are not tailored to AI/ML algorithms
- In R18 (roughly 2022), 3GPP starts to Evaluate the benefits of augmenting the air-interface to support machine learning based algorithms for enhanced performance and/or reduced complexity/overhead
  - –Identifying what is required for an adequate AI model characterization and description establishing pertinent notation for discussions and subsequent evaluations.
  - –Identifying various levels of collaboration between user equipment (UE) and the base station relevant to the selected use cases.
  - –Understanding the attainable gains and the associated complexity requirement based on the identified KPIs.
  - –Characterizing life-cycle management of AI model: e.g., model training, model deployment, model inference, model monitoring, model updating

Use Cases Studied by 3GPP (5G-Advance)

- Channel feedback
  - –Traditional channel state information (CSI) feedback protocol is based on a pre-determined basis of channel/precoder representation
  - –More efficient, predictive CSI feedback based on AI models can improve user downlink throughput and reduce uplink overhead
- Beam management
  - –Beam prediction in time/spatial domain for overhead and latency reduction, improving beam selection accuracy by AI models
  - –Important for mmW systems
- Precise positioning
  - –Positioning accuracy enhancement for different indoor and outdoor scenarios, improve handling of non-line-of-sight path by AI models
- Mobility
  - –AI/ML based RRM measurement and event prediction, including neighboring cell measurement, HO failure/RLF prediction, etc

6G: AI Use Cases Scope Discussion in Progress

Use cases		Primary agendas/topic category
Low overhead CSI-RS or CSI prediction with AI/ML		CSI-RS and CSI acquisition
Low overhead DMRS with AI/ML receiver		UL & DL DMRS associated with PUSCH/PDSCH Note: DMRS-free may be related to modulation
CSI compression and feedback		CSI acquisition Note: this may be related to uplink control
AI/ML for beam management and extension		From initial access and beam management perspective Note: maybe related to mobility
AI/ML for SRS	Low overhead SRS with AI/ML	SRS
	Low PAPR SRS sequence design	
	AI/ML based SRS power imbalance compensation	
AI-enabled UL precoder indication		UL MIMO
AI-based non-linearity handling at transmitter or receiver		Related to DMRS, SRS, Power control
AI/ML for (de)modulation		Modulation, may be related to MIMO assuming no change to DMRS
AI/ML based waveform for PAPR reduction		Waveform
AI/ML based HARQ-ACK feedback		Channel coding Note: this may be related to uplink control
PDCCH related	Prior-Information-Aided DCI Decoding	DL control
	Lossless DCI Compression	
Power control related	UL closed-loop power control	Power control
	Pathloss prediction	
RACH related design	Early contention resolution in RACH	Random access/PRACH
	Low PAPR PRACH sequence design	
Site Specific Learning for AI/ML using RAN Digital Twin		Depending on corresponding use case where site specific learning is applicable, e.g., DMRS
Digital twin construction related use cases	AI/ML-enabled RAN digital twin with distributed model	ISAC
	Sensing based RAN digital twin construction with NW-side AI/ML model	ISAC
AI for positioning		Positioning related agenda, if any

Extension of 5GA cases

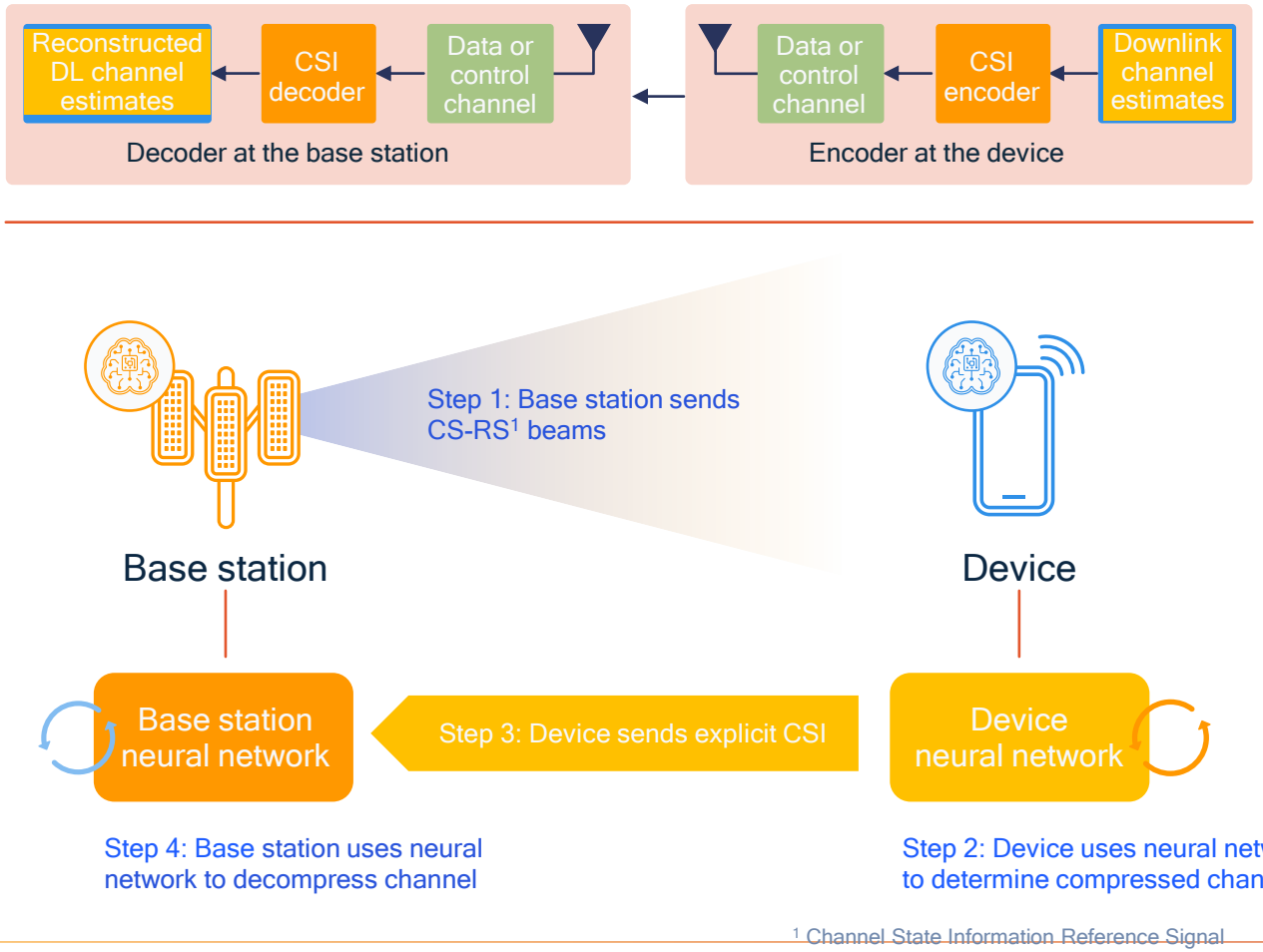
Digital twin related

Use case to accommodate AI/ML service, e.g., token traffic: if any impact, most likely scheduling/HARQ

Source: 3GPP RAN1#123 chair notes

Cross-node machine learning based channel state information

Explicit channel feedback framework for CSI compression and prediction utilizing domain knowledge and neural networks



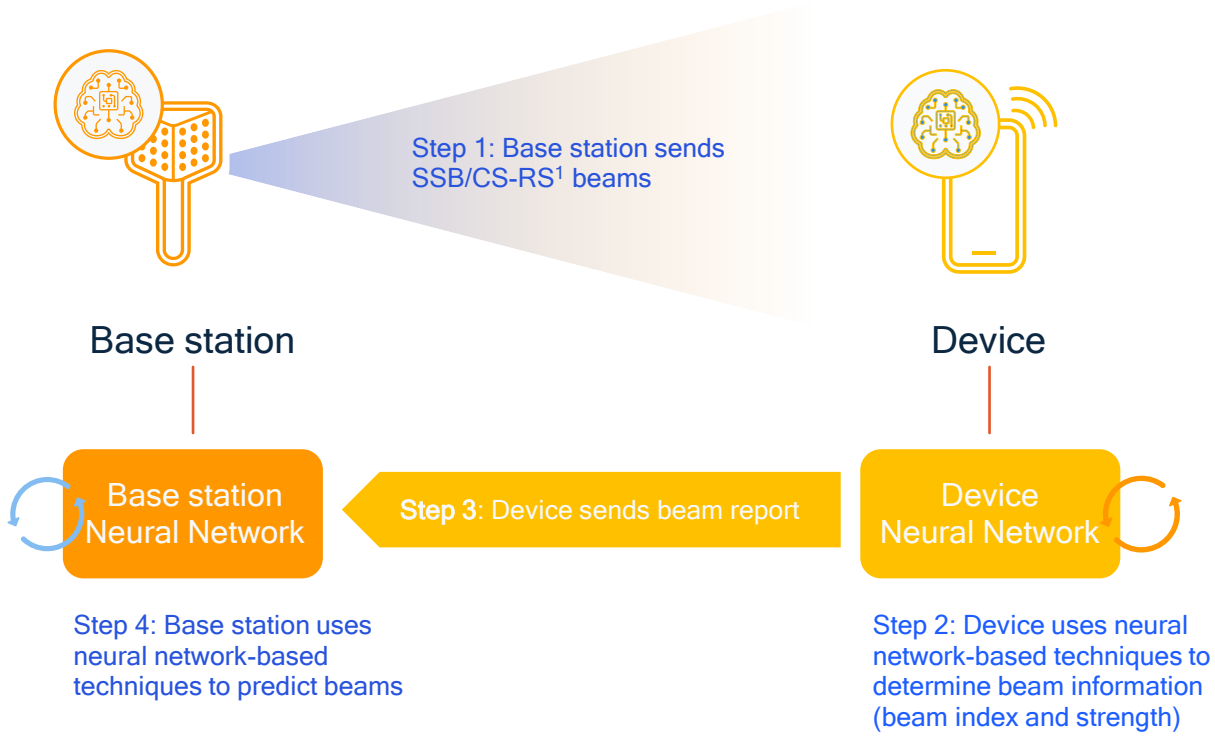
[1] Channel State Information Reference Signal

- CSI feedback: precoder, CQI and RI
- Traditional precoder indication: based on a general and fixed basis. Report index or co-efficients
- Improve system efficiency with neural network framework for CSI on non-linear encoding and decoding
  - Different models finding different representations for different propagation and environment conditions
- Customized, lower overhead feedback based on individual device
- Challenging issue: model matching when full information is not shared between vendors

*Figure and material source of reference: “Towards an AI native communications system design”, Qualcomm Technologies, Inc.

Machine learning based mmWave beam management

Beam prediction utilizing domain knowledge, location, velocity, other aspects of environmental and application awareness to improve robustness and throughput



- Reduced feedback overhead and transmission from device
  - Predict beam in time domain by recognizing the movement/rotation
- Reduced transmission of reference signal from base station
  - Predict beam in spatial domain by recognizing/infering codebook
- Efficient beam tracking and beam search for higher performance
- Challenge: variation of beam codebook design across network vendors

[1] Synchronization Signal Block / Channel State Information Reference Signal

*Figure and material source of reference: “Towards an AI native communications system design”, Qualcomm Technologies, Inc.

MEDIATEK

Part 3

Introduction to Deep Learning

- Machine Learning Fundamentals
- Neural Networks
- Training a model
- Case study: AI for CSI

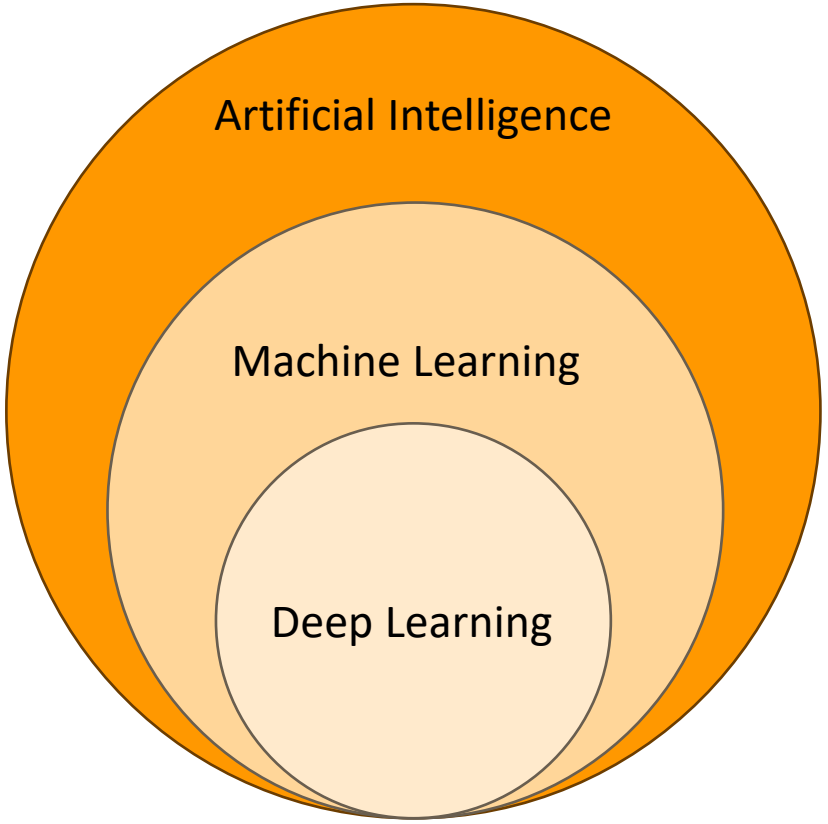
Machine Learning 101

- Field concerned with the development and study of statistical algorithms that can learn from data and generalise to unseen data

- Three main categories of learning algorithms:
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning

- Two main tasks for supervised learning
  - Classification
  - Regression

- Examples
  - Neural networks
  - Decision Trees
  - Support Vector Machines
  - Clustering



Machine Learning 101

- Field concerned with the develo… learn from data and generalise to …
- Three main categories of le…
  - **Supervised Learning**
  - Unsupervised Learning
  - Reinforcement Learning
- Two main tasks for supervi…
  - Classification
  - Regression
- Examples
  - Neural networks
  - Decision Trees
  - Support Vector Machines
  - Clustering

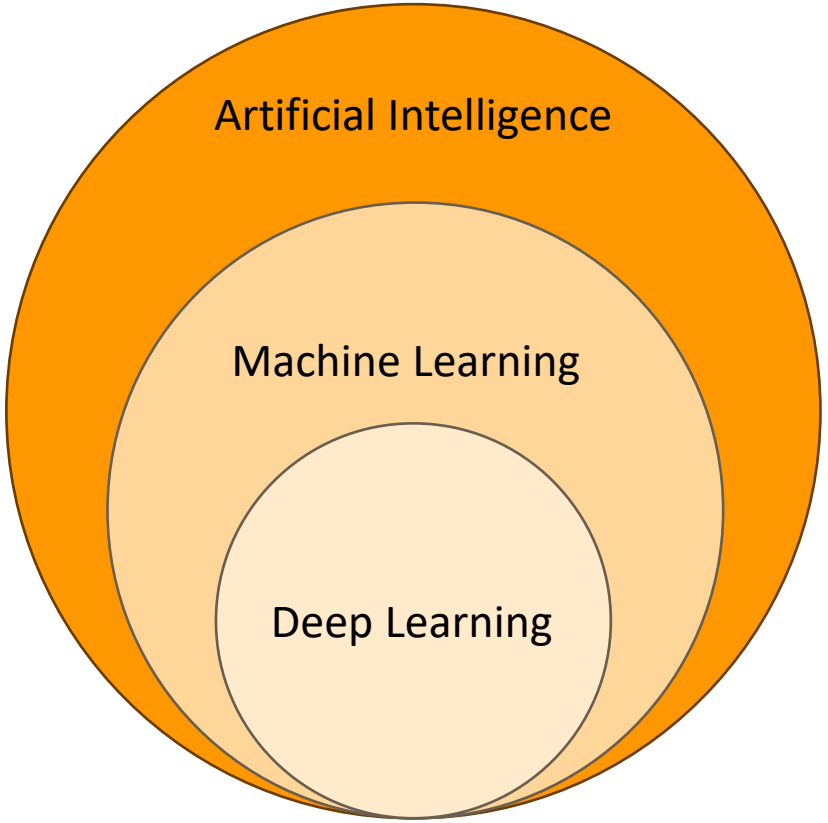
Supervised Learning uses labelled data to train a model

Dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

The objective is to learn a function $f: X \to Y$

Training is done by minimizing a loss $\sum_i \mathcal{L}(f(x_i), y_i)$

Obtaining the label for each datapoint may be difficult/expensive



Machine Learning 101

- Field concerned with the d
  from data and generalise
- Three main categories o
  - Supervised Learning
  - **Unsupervised Learning**
  - Reinforcement Learning
- Two main tasks for supe
  - Classification
  - Regression
- Examples
  - Neural networks
  - Decision Trees
  - Support Vector Machines
  - Clustering

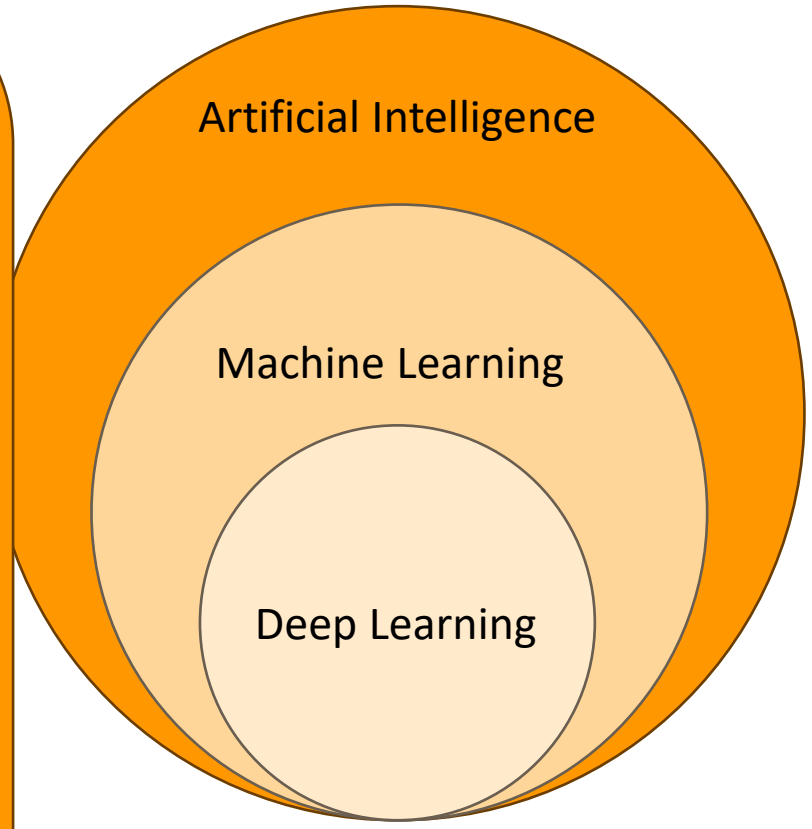
Unsupervised Learning aims at finding patterns and structures in unlabeled data

Dataset $\mathcal{D} = \{x_1, \dots, x_n\}$

The objective is to learn a function
$f: X \to Z$
(Z is a latent representation)

Training is done by minimizing a loss
$\sum_i \mathcal{L}(f(\tilde{x}_i), x_i)$
the loss depends on the input data itself and $\tilde{x}_i$ can be a transformation of the input(e.g., the latent representation should be reflect some properties of the data or should allow reconstruction of missing parts of the data)

Obtaining unlabelled data is usually cheap



Machine Le

- Field concerned with the
  from data and generalise

- Three main categories of
  - Supervised Learning
  - Unsupervised Learning
  - **Reinforcement Learning**

- Two main tasks for super
  - Classification
  - Regression

- Examples
  - Neural networks
  - Decision Trees
  - Support Vector Machines
  - Clustering

Reinforcement Learning is a framework in which an agent learns to make sequential decisions by interacting with an environment to maximize cumulative reward

Modeled as a Markov Decision Process

(S, A, P, R, γ)

S – set of states

A – set of actions

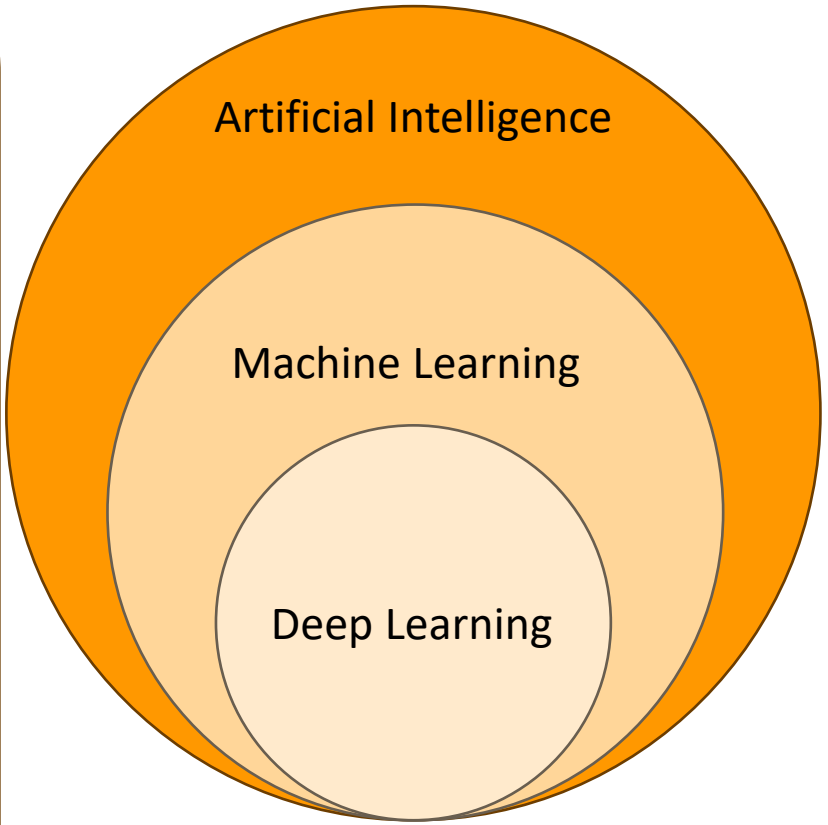
$P(s'|s, a)$ – transition probability

$R(s, a)$ – reward function

γ - discount factor for future rewards

The goal is to learn a policy

$$\pi(a|s) = argmax \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$



Machine Learning 101

- Field concerned with the development an
  from data and generalise to unseen data

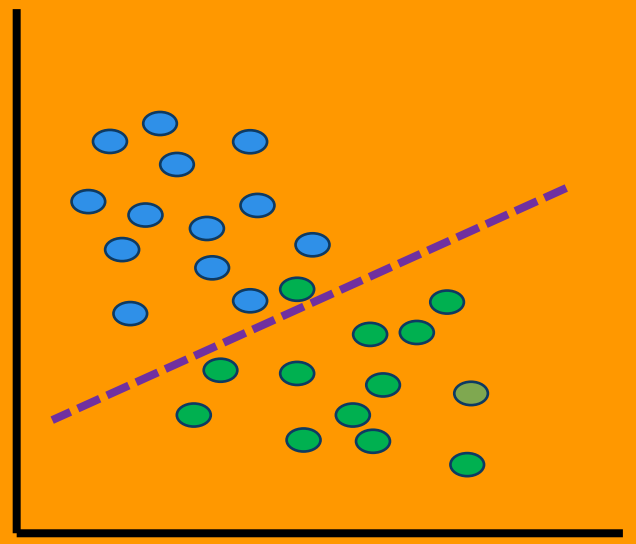
- Three main categories of learning algorith
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning

- Two main tasks for supervised learning
  - **Classification**
  - **Regression**

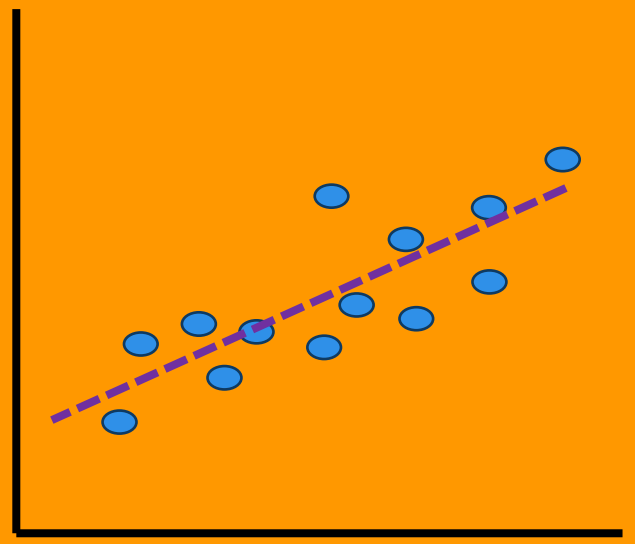
- Examples
  - Neural networks
  - Decision Trees
  - Support Vector Machines
  - Clustering

Classification is the task of assigning objects to some pre-existing classes

Regression is the task of predicting a continuous numerical value



Classification



Regression

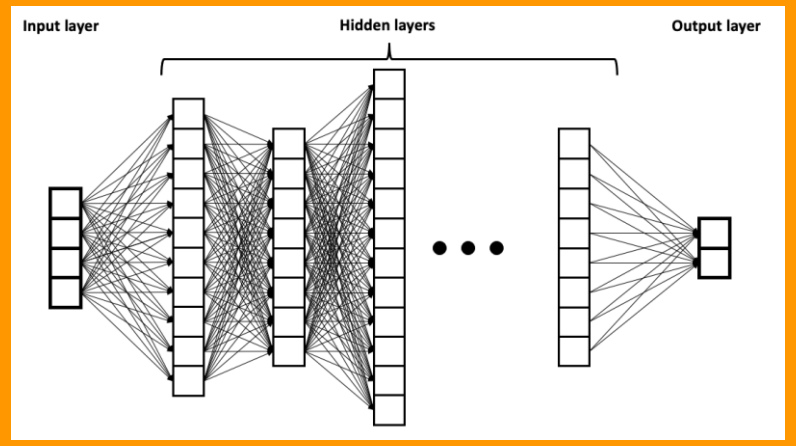
Machine Learning 101

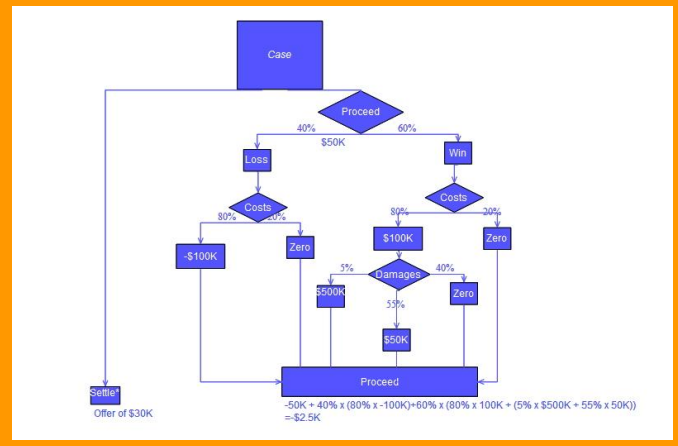
- Field concerned with the development an
  from data and generalise to unseen data

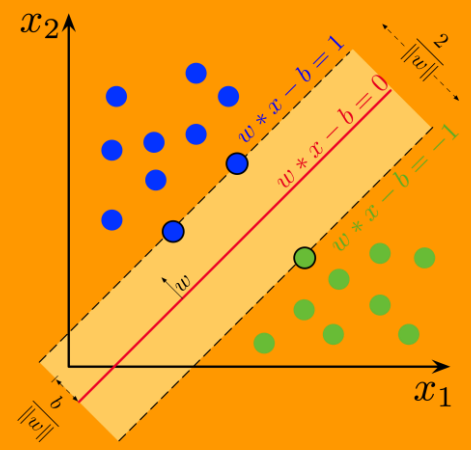
- Three main categories of learning algorith
  - Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning

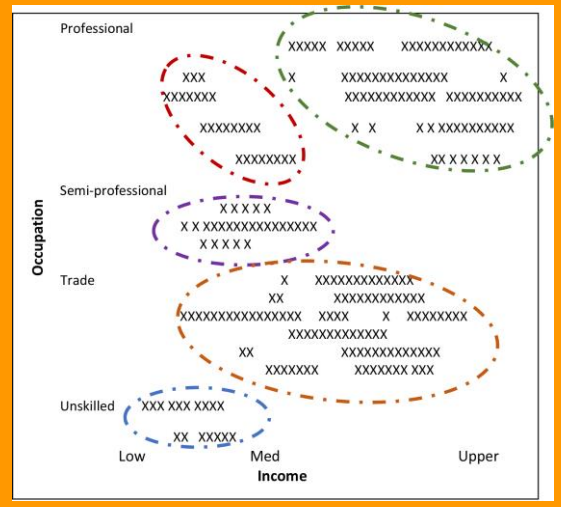
- Two main tasks for supervised learning
  - Classification
  - Regression

- **Examples**
  - **Neural networks**
  - **Decision Trees**
  - **Support Vector Machines**
  - **Clustering**





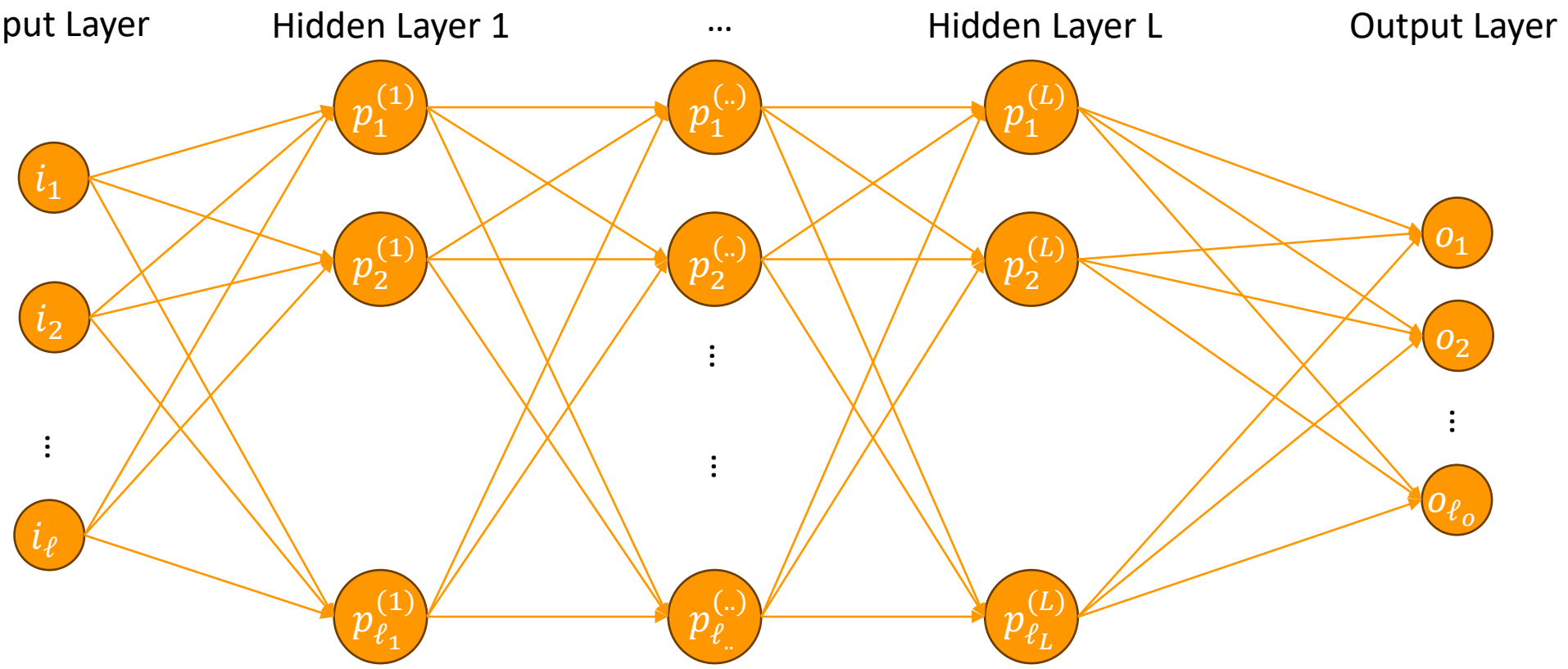




Images from Wikimedia Commons

Our focus: Neural Networks

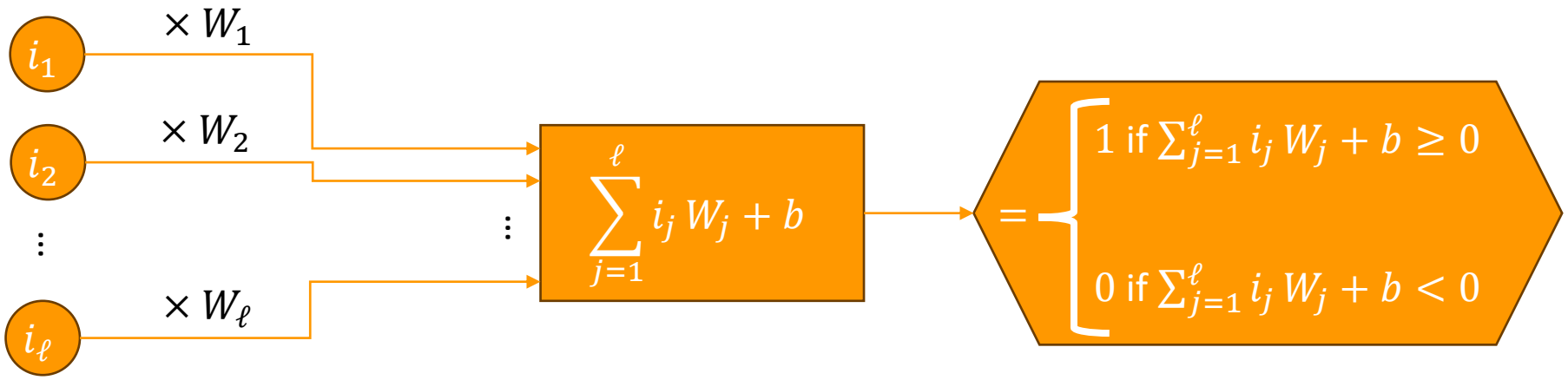
- Machine learning model inspired by the human brain that uses interconnected nodes, or **artificial neurons**, to process data



The Perceptron Model

- Binary classifier representing an "artificial neuron"

$W_1, W_2, \ldots, W_\ell$ are learnable weights

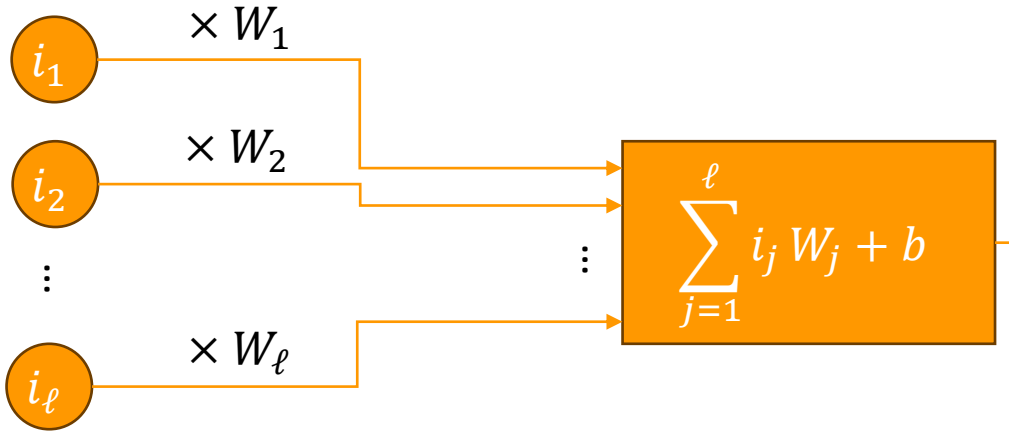


- In vector form: $f(\boldsymbol{i}) = h(\boldsymbol{W}\boldsymbol{i} + b)$
  - $\boldsymbol{i} \in \mathbb{R}^\ell$
  - $\boldsymbol{W} \in \mathbb{R}^\ell$
  - $b \in \mathbb{R}$

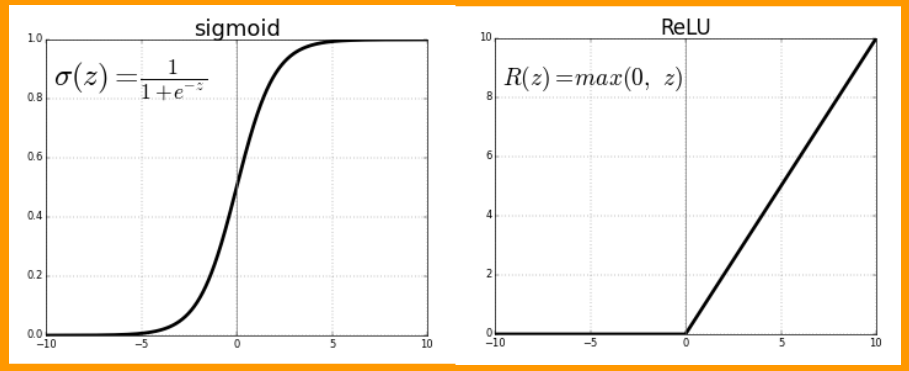
The Perceptron Model

- Binary classifier representing an "artificial neuron"

$W_1, W_2, \ldots, W_\ell$ are learnable weights



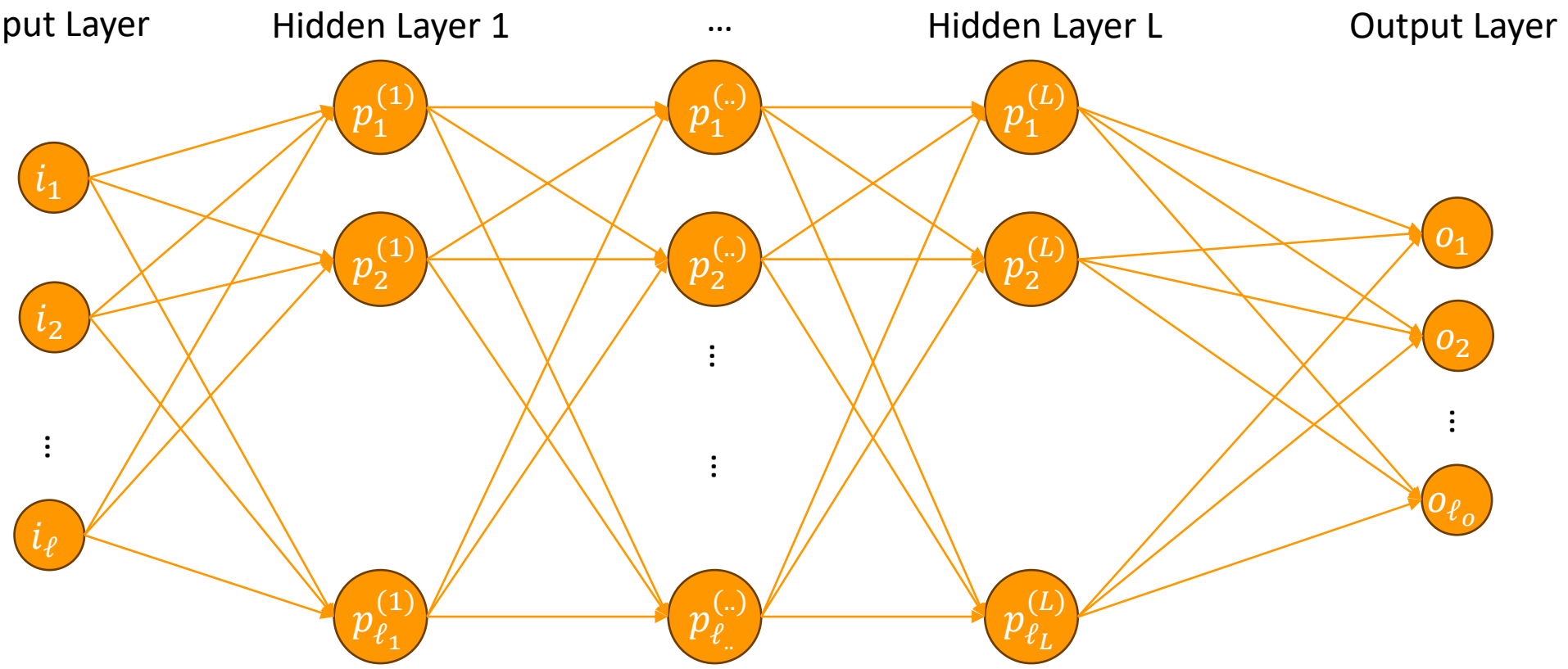
In practice, we use different "activation functions", so that the output of each neuron looks like a "smooth step function":



- In vector form: $f(\boldsymbol{i}) = h(\boldsymbol{W}\boldsymbol{i} + b)$
  - $\boldsymbol{i} \in \mathbb{R}^\ell$
  - $\boldsymbol{W} \in \mathbb{R}^\ell$
  - $b \in \mathbb{R}$

The Multilayer Perceptron (MLP) Model

- Feedforward neural network model constructed by having fully connected layers of perceptrons



Training

Dataset $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

- Training deep learning models is done using gradient descent methods

$$\theta(t+1) = \theta(t) - \gamma \nabla_\theta L$$

γ is the **learning rate**

- Defining the Loss function

Classification

Cross entropy (CE)

$$L(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}) = -\sum_{c=1}^{M} y_c \log f(x)_c$$

$f(x)_c$ is predicted probability for class c
y_c is 1 or 0

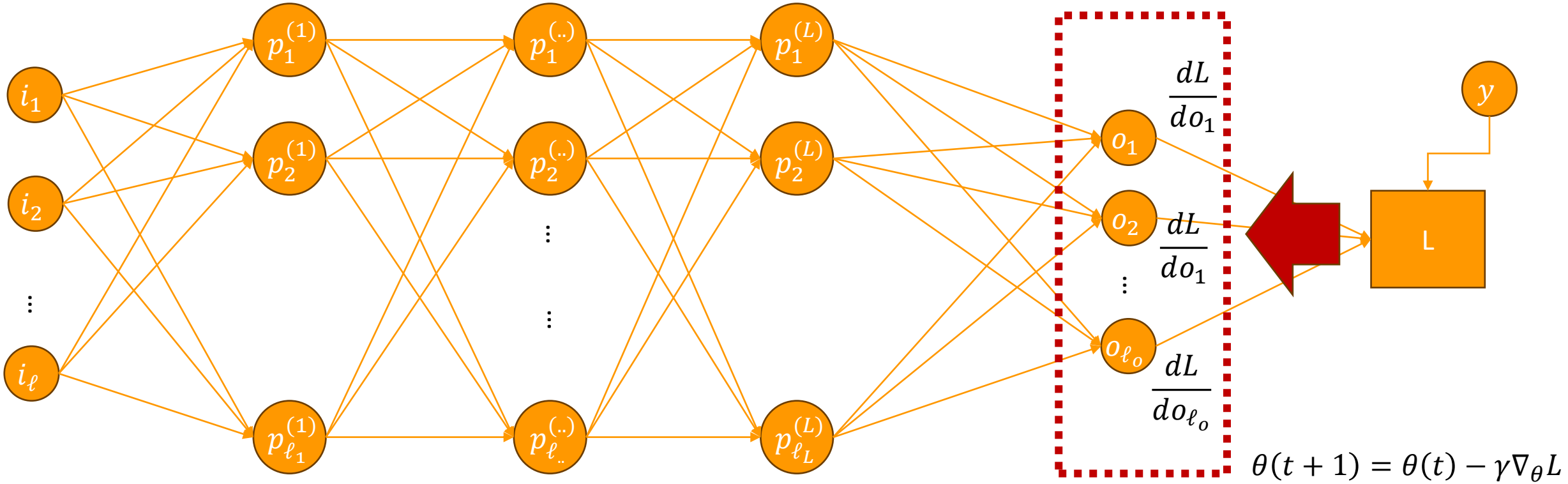
Regression

Mean squared error (MSE)

$$L(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}) = \|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y}\|^2 = \sum_i (f(x_i) - y_i)^2$$

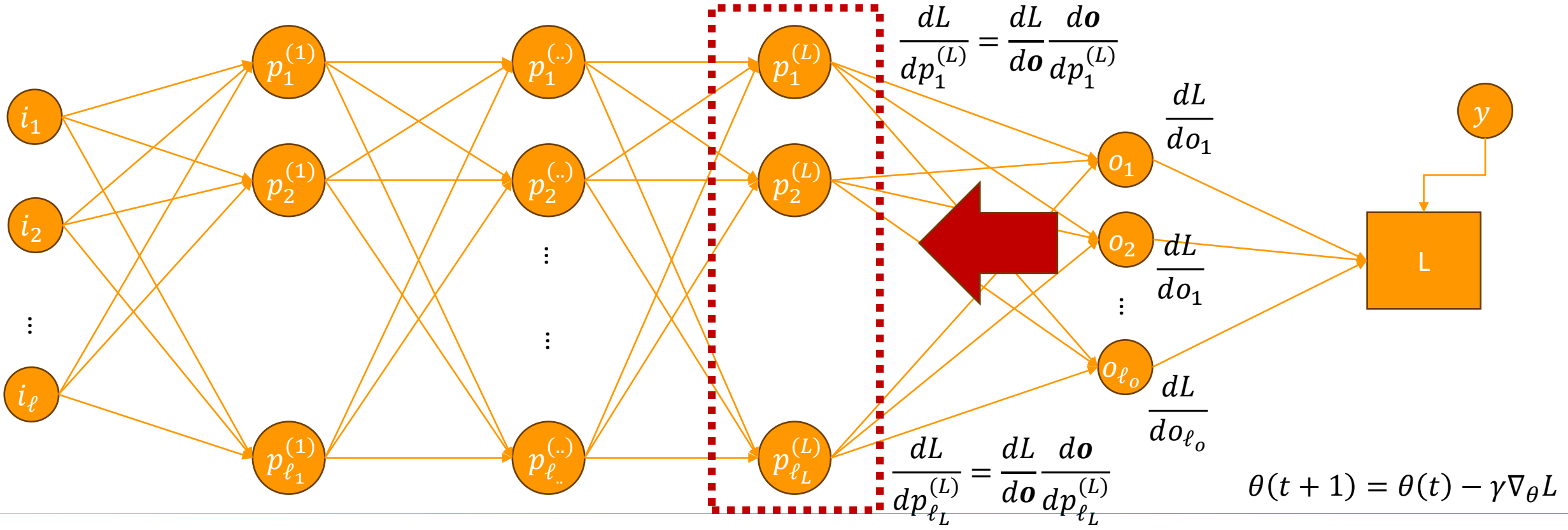
The Backpropagation Algorithm

- Efficient application of the chain rule to neural networks
- Computes the gradient of a loss function with respect to the weights of the network for a single input–output example, computing the gradient one layer at a time, starting from the last layer and going backwards towards the first



The Backpropagation Algorithm

- Efficient application of the chain rule to neural networks
- Computes the gradient of a loss function with respect to the weights of the network for a single input–output example, computing the gradient one layer at a time, starting from the last layer and going backwards towards the first



Backpropagation in Practice

- Modern libraries for deep learning have a built in implementations of backpropagation
  - PyTorch
  - Jax
  - Tensorflow

- Example in Pytorch

Let's consider function $f(x, y) = 3x^2 + 2xy + y^2$

We want to compute the derivative at $x = 1, y = 2$

Let's compute the partial derivatives: $\frac{df}{dx} = 6x + 2y$ $\frac{df}{dy} = 2x + 2y$

Plugging in $x = 1, y = 2$ we get $\nabla f(1,2) = (10, 6)$

Backpropagation in Practice

- Modern libraries for deep learning have a built in implementations of backpropagation
  - PyTorch
  - Jax
  - Tensorflow

- Example in Pytorch

Let's consider function $f(x, y) = 3x^2 + 2xy + y^2$

We want to compute the derivative at $x = 1, y = 2$

Let's compute the partial derivatives: $\frac{df}{dx} = 6x + 2y$ $\frac{df}{dy} = 2x + 2y$

Plugging in $x = 1, y = 2$ we get $\nabla f(1,2) = (10, 6)$

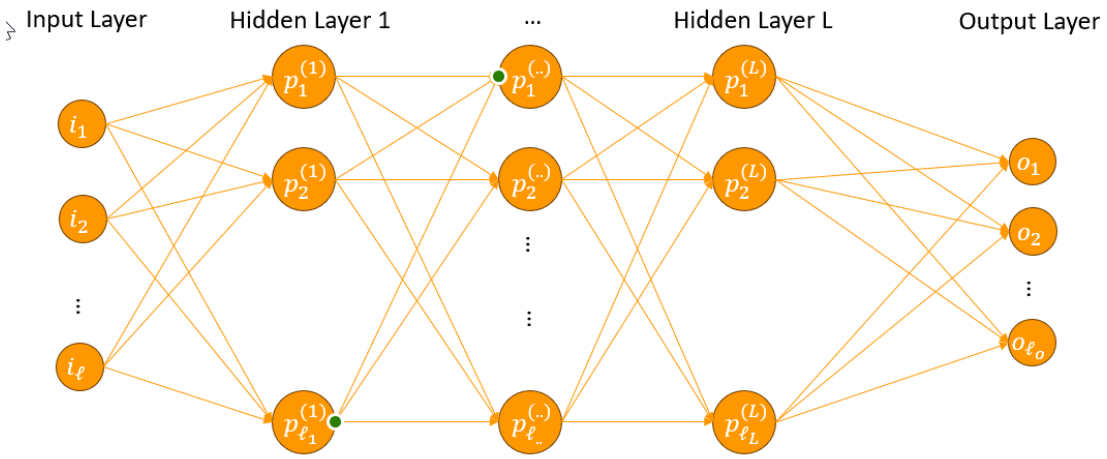
```python
import torch

# Create tensors with requires_grad=True
x = torch.tensor(1.0, requires_grad=True)
y = torch.tensor(2.0, requires_grad=True)

# Define the function
f = 3 * x**2 + 2 * x * y + y**2

# Compute gradients
f.backward()

# Print the gradients
print("df/dx:", x.grad)  # Should print 10.0
print("df/dy:", y.grad)  # Should print 6.0
```

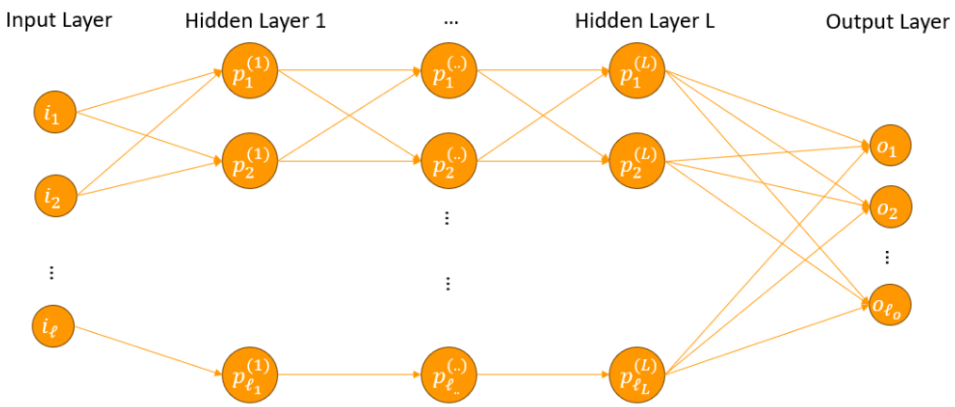
Output:

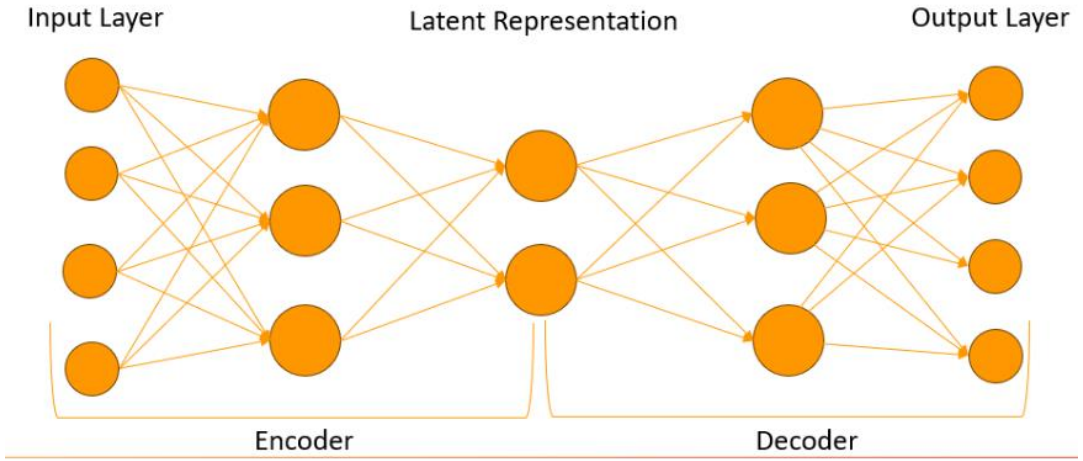
```
df/dx: tensor(10.)
df/dy: tensor(6.)
```

Neural Network Architecture

- The architecture of a neural network defines its computational graph



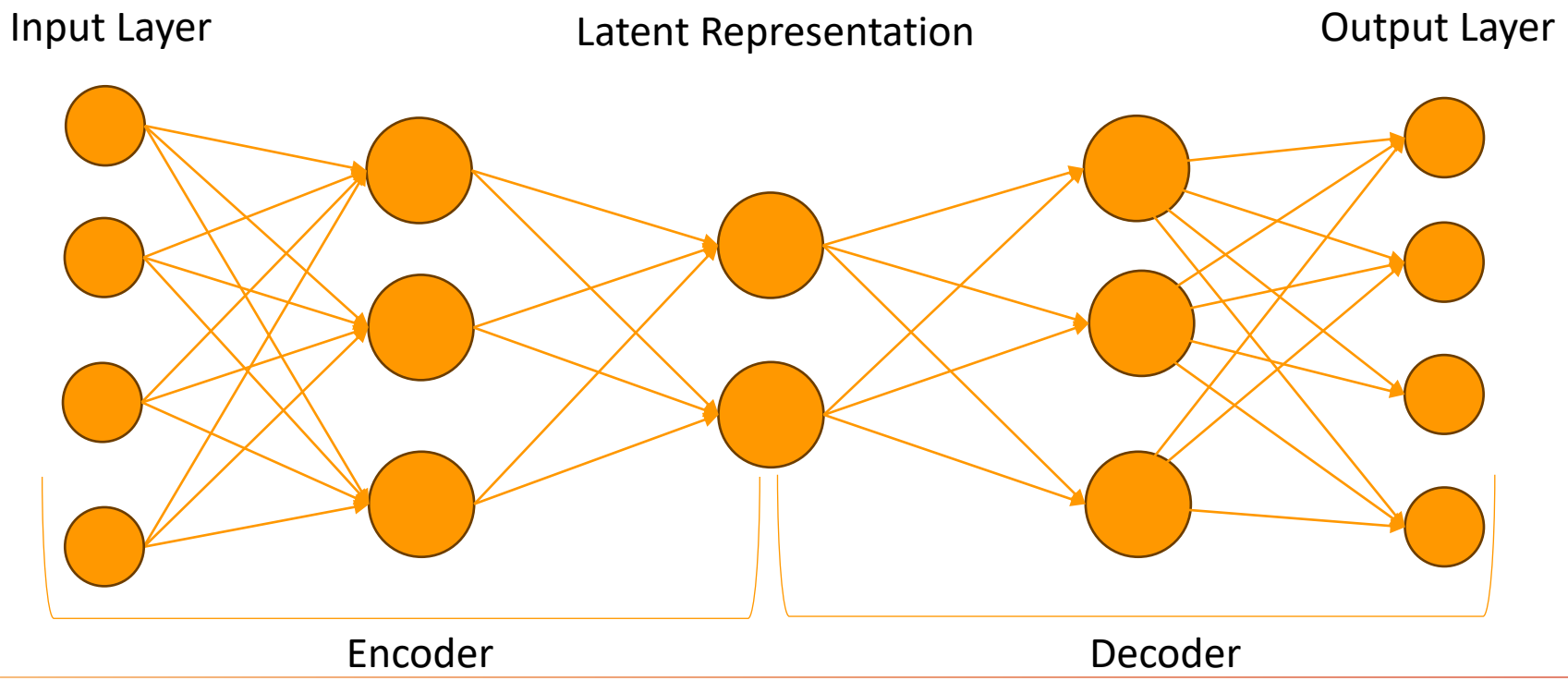




Autoencoder

Composed of

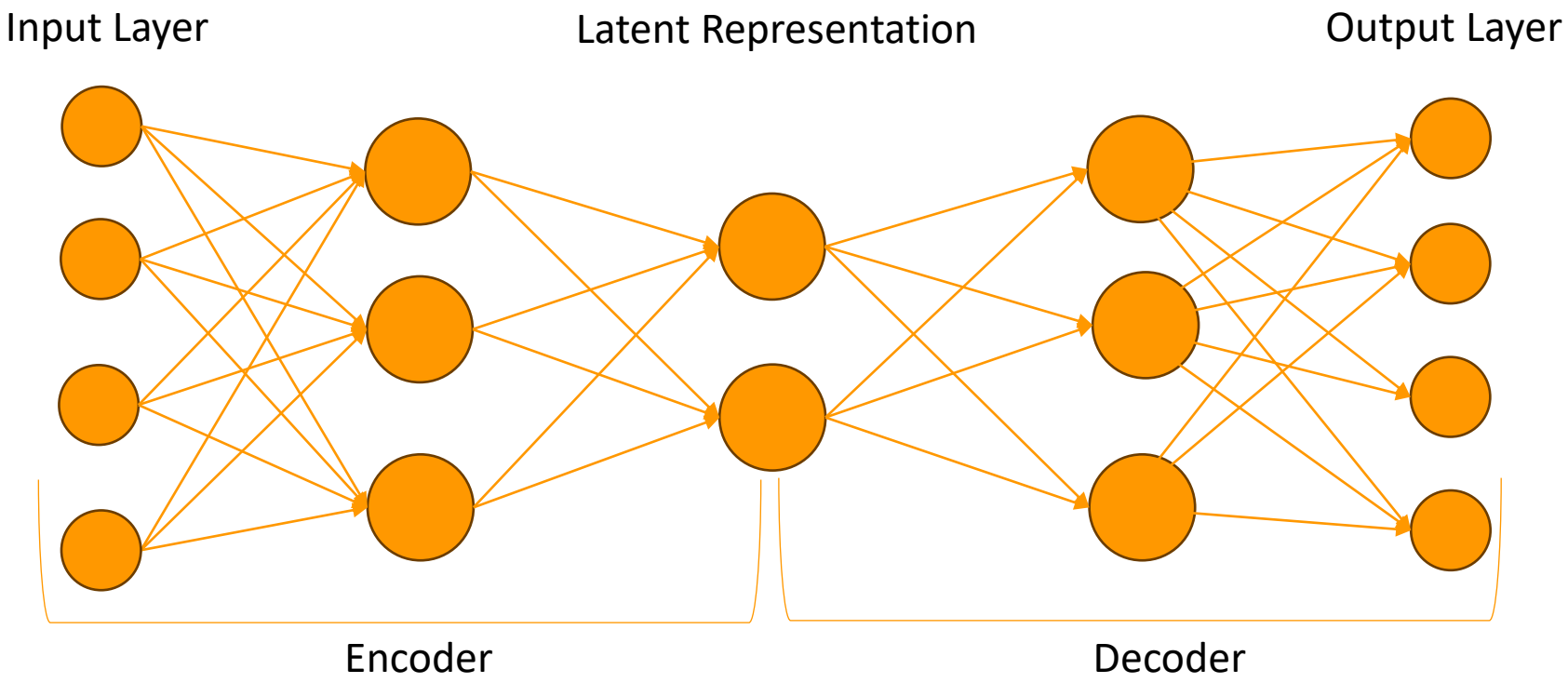
- Encoder: maps input to lower dimensional latent space
- Decoder: maps lower dimensional representation back to input space



Autoencoder

Composed of

- Encoder: maps input to lower dimensional latent space
- Decoder: maps lower dimensional representation back to input space



Trained to reconstruct the input

$$L = \left| x - D\left(E(x)\right)\right|^2$$

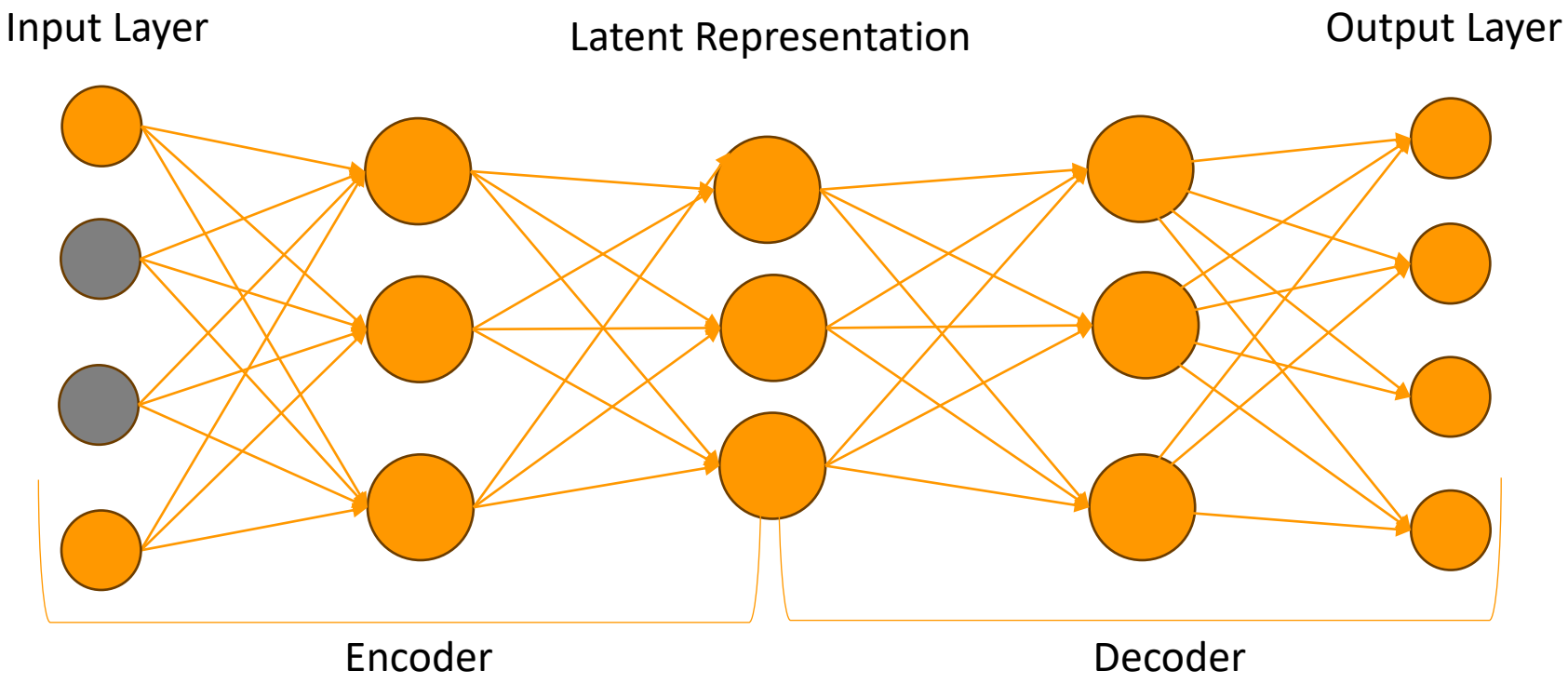
The model learns to produce lower dimensional representations that preserve as much as possible of the input information

Can be used as a dimensionality reduction tool

Denoising Autoencoder

Composed of

- Corruption strategy
- Encoder: maps input to lower dimensional latent space
- Decoder: maps lower dimensional representation back to input space



Trained to reconstruct the input from a randomly corrupted version of it

$$L = \left| x - D\left(E(\tilde{x})\right)\right|^2$$

Every training input is corrupted in a random manner.
The model learns to “clean” the input data

Can be used as a noise reduction tool

Other Popular Architectures

Convolutional Neural Networks (CNNs)

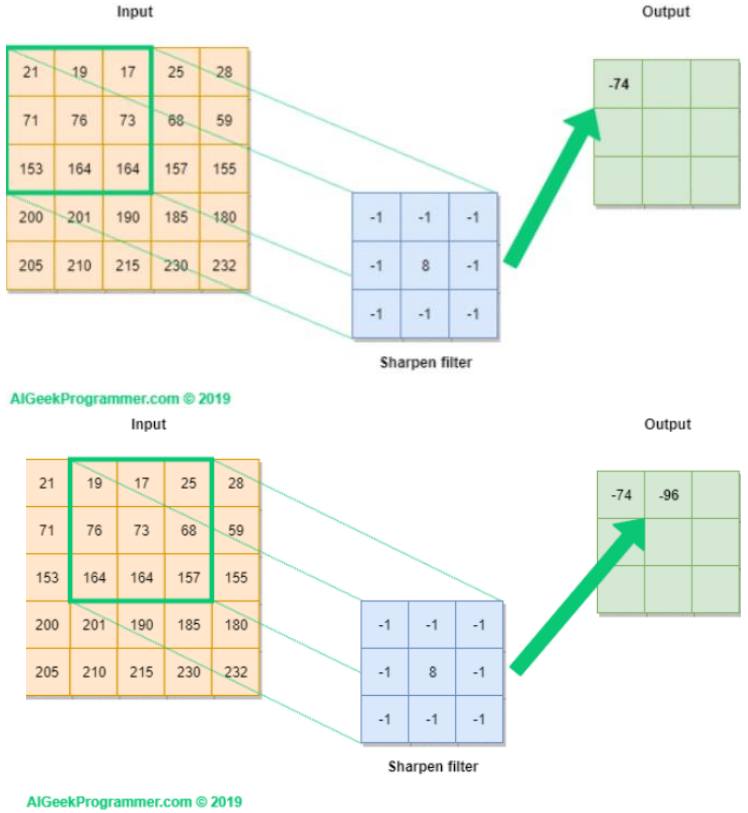
- Designed to process data with a grid-like topology (e.g., images, video frames).
- Use convolutional layers to automatically learn spatial hierarchies of features.
- Reduce parameters compared to fully connected networks by sharing weights.

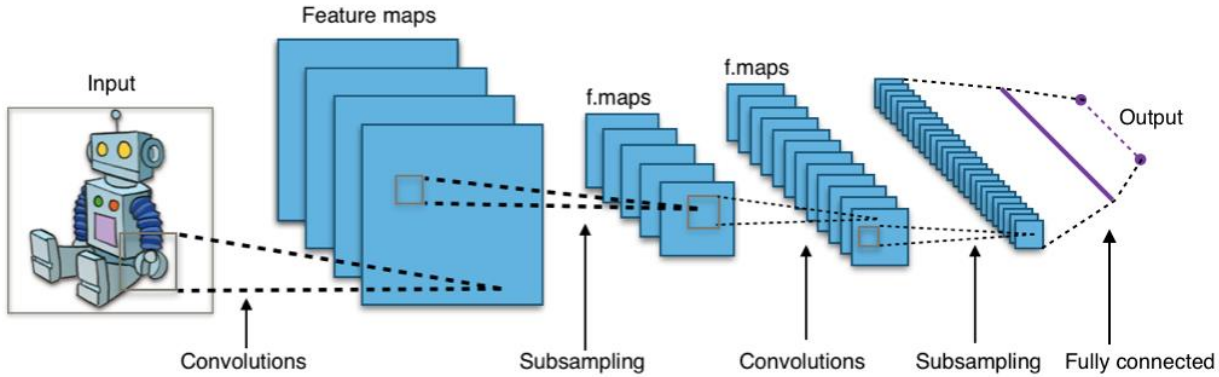
Strengths:

- Excellent at capturing local patterns (edges, textures, shapes).
- Translation invariance — features recognized regardless of position.

Typical use cases:

- Image classification (e.g., object recognition).
- Image segmentation (e.g., medical imaging).
- Video analysis.
- Some applications in audio and time-series (spectrograms).



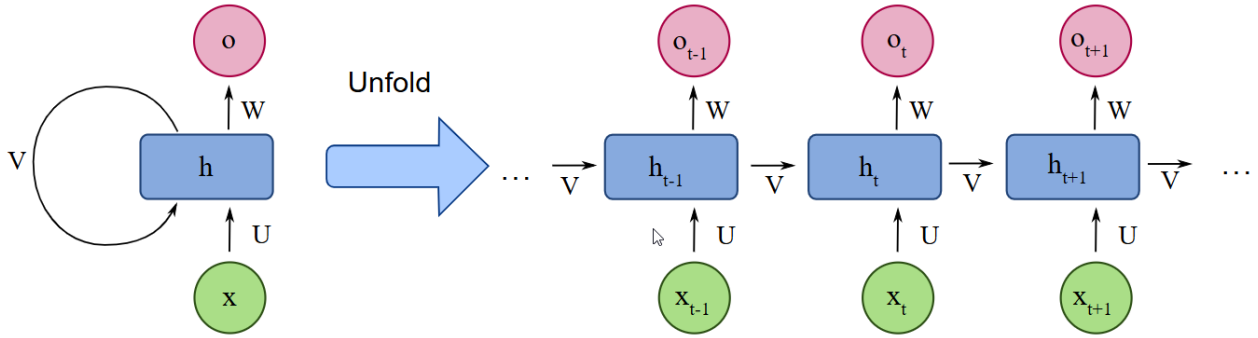


Images from Wikimedia Commons

Other Popular Architectures

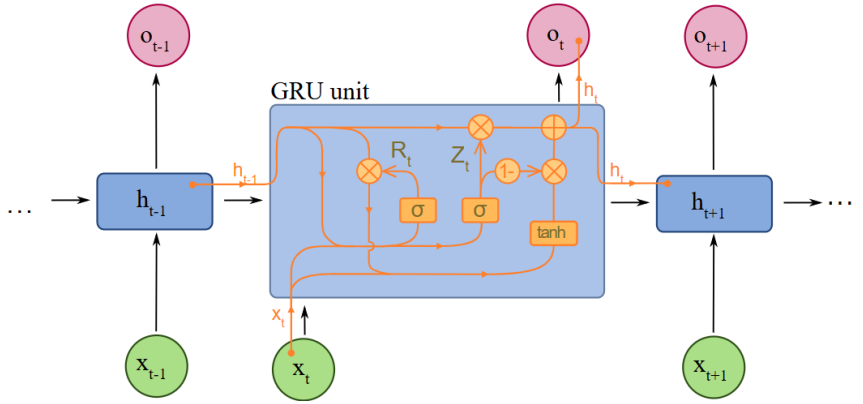
Recurrent Neural Networks (RNNs)

- Designed for sequential data — process inputs step-by-step while maintaining a hidden state.
- LSTMs and GRUs are improved RNN variants that handle long-term dependencies better.



Strengths:

- Capture temporal relationships and context over time.
- Can model variable-length sequences.



Typical use cases:

- Natural language processing (text generation, sentiment analysis).
  - Now Transformers are used instead
- Speech recognition.
- Time-series forecasting.
- Sequential event prediction.

Images from Wikimedia Commons

Other Popular Architectures

Transformer (we'll present it in detail later)

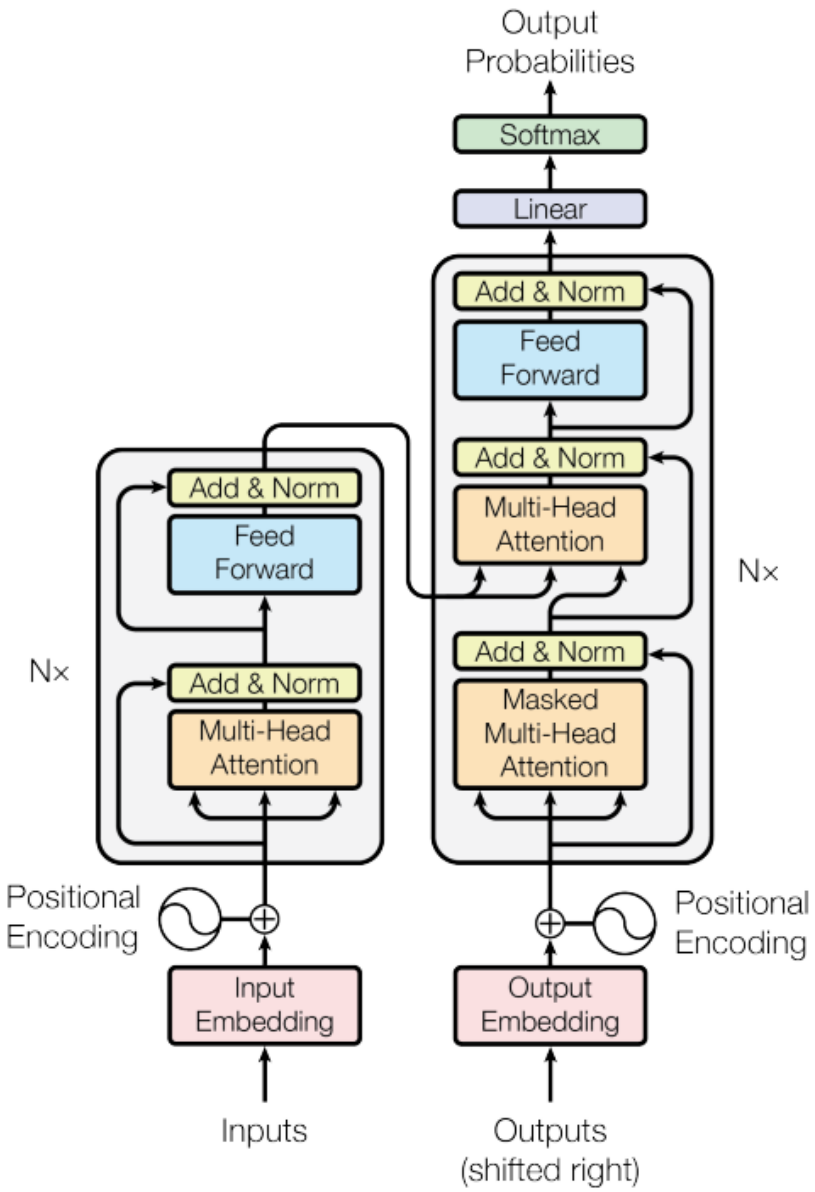
- Use self-attention mechanisms to model relationships between all elements in a sequence simultaneously.
- Highly parallelizable compared to RNNs.
- Scales well to large datasets and long sequences.

Strengths:

- Capture global context efficiently.
- State-of-the-art in many NLP tasks.
- Flexible — can be adapted to vision, audio, and multimodal tasks.

Typical use cases:

- Large language models (e.g., ChatGPT).
- Foundation Models
- Nowadays, almost anything ☺



Training a Neural Network

- Define Task
  - Classification
  - Regression
  - Clustering/embedding
- Collect Data
  - General good practices
- Define Neural Network Architecture
  - MLP
  - CNN
  - Transformer
- Define Loss
  - MSE
  - Cross Entropy
- Monitor Training

Training a Neural N

- **Define Task**
  - Classification
  - Regression
  - Clustering/embedding
- Collect Data
  - General good practices
- Define Neural Network Architecture
  - MLP
  - CNN
  - Transformer
- Define Loss
  - MSE
  - Cross Entropy
- Monitor Training

Defining the Task

Understand what you want the neural network to do.

- Identify the type of task (classification, regression, etc.)
- Define success metrics (Accuracy, F1-score, RMSE, MAE, etc.)
- Understand constraints (Latency, model size, interpretability, deployment environment)
- Identify domain-specific challenges

Good Practices:

- Start with a baseline model to set expectations.
- Definee goal: e.g., “Reach 90% accuracy on validation set.”
- Consider feasibility: Is the data available? Is the problem solvable with current resources?
- Think about deployment early: Will it run on mobile, cloud, or embedded devices?

Training a Neural N

- Define Task
  - Classification
  - Regression
  - Clustering/embedding
- **Collect Data**
  - General good practices
- Define Neural Network Architecture
  - MLP
  - CNN
  - Transformer
- Define Loss
  - MSE
  - Cross Entropy
- Monitor Training

Collecting Data

Gather and prepare data that represents the problem well

Data Gathering

- Sources: Public datasets, internal logs, sensors, APIs.
- Quantity vs. quality: More data helps, but clean, representative data is critical.
- Labeling: Manual annotation, crowdsourcing, semi-supervised methods.
- Balance: Try to avoid class imbalance or prepare counter-measures

Data Preparation

- Cleaning: Handle missing values, duplicates, errors.
- Normalization/Standardization
- Splitting: Train/validation/test
- Augmentation: For images/audio/text, apply transformations to increase diversity.
- Bias: Identify potential biases in the data

Training a Neural

- Define Task
  - Classification
  - Regression
  - Clustering/embedding
- Collect Data
  - General good practices
- **Define Neural Network Architecture**
  - MLP
  - CNN
  - Transformer
- Define Loss
  - MSE
  - Cross Entropy
- Monitor Training

Defining the Architecture

Choose a model structure that fits the task and constraints

- Match to task:
  - CNNs for images
  - RNNs/LSTMs/Transformers for sequences
  - MLPs for tabular data
  - Graph Neural Networks for relational data
- Prepare a hyperparameter tuning plan
- Pretrained models: (if available) Use transfer learning to save time and resources.

Good Practices

- Start with proven architectures: ResNet, BERT, Qwen, etc., then adapt.
- Implement probing strategy to check behaviour
- Experiment systematically: Change one thing at a time to understand impact.

Training a Neu

- Define Task
  - Classification
  - Regression
  - Clustering/embedding
- Collect Data
  - General good practices
- **Define Neural Network Architecture**
  - MLP
  - CNN
  - Transformer
- Define Loss
  - MSE
  - Cross Entropy
- Monitor Training

Defining the Architecture - Inductive Biases

Inductive bias is the built-in assumptions a model uses to guide learning toward certain patterns in the data.

Based on Data type & structure

- CNNs: Inductive bias toward local spatial patterns and translation invariance.
  - Great for images because they assume nearby pixels are related.
- RNNs/LSTMs/GRUs: Inductive bias toward temporal order and sequential dependencies.
  - Great for time-series and text because they assume order matters.
- Transformers: Inductive bias toward global relationships via attention.
  - Great for tasks where any part of the input can relate to any other part.

Based on Amount of data

- Strong inductive biases (CNNs, RNNs) can work well with less data because they “bake in” assumptions about the problem.
- Weak inductive biases (Transformers) are more flexible but often need more data to learn patterns from scratch.

Based on Computational resources

- Consider trade-offs: stronger inductive biases can mean smaller models and faster training.

Training a Neura

- Define Task
  - Classification
  - Regression
  - Clustering/embedding
- Collect Data
  - General good practices
- Define Neural Network Architecture
  - MLP
  - CNN
  - Transformer
- **Define Loss**
  - MSE
  - Cross Entropy
- Monitor Training

Defining the Loss

Select a loss function that aligns with your objective

Common Loss Functions

- Classification: Cross-entropy loss, focal loss (for imbalance).
- Regression: Mean Squared Error (MSE), Mean Absolute Error (MAE).
- Generative models: adversarial loss, reconstruction loss.
- Add special losses based on your task. E.g.:
  - Object detection: combination of classification + bounding box regression losses.
  - Ranking: pairwise or triplet loss.

Good Practices:

- Match loss to metric: If optimizing F1-score, consider surrogate losses that correlate well.
- Handle imbalance: Weighted loss or class-specific penalties.
- Monitor multiple metrics: Loss alone may not reflect real-world performance.
- Stability: Some losses require careful initialization or learning rate tuning.

Training a Neura

- Define Task
  - Classification
  - Regression
  - Clustering/embedding
- Collect Data
  - General good practices
- Define Neural Network Architecture
  - MLP
  - CNN
  - Transformer
- Define Loss
  - MSE
  - Cross Entropy
- **Monitor Training**

Monitor Training

You want to monitor both training progress and generalization

Core metrics:

- Training loss – How well the model fits the training data.
- Validation loss – How well the model generalizes to unseen data.
- Task-specific metrics – Accuracy, F1-score, precision/recall, RMSE, etc.
- Gradient norms – Detect exploding/vanishing gradients.
- Throughput – Samples/sec, GPU utilization (for efficiency tracking)

In-training evaluation

- Evaluate on a separate validation set
- Track Multiple Metrics (accuracy, F1, entropy of predictions, etc.)
- Save checkpoints based on performance
- Include qualitative checks (e.g., visualize predictions/errors)

Training a Neura

- Define Task
  - Classification
  - Regression
  - Clustering/embedding
- Collect Data
  - General good practices
- Define Neural Network Architecture
  - MLP
  - CNN
  - Transformer
- Define Loss
  - MSE
  - Cross Entropy
- Monitor Training

Popular Libraries

General Deep Learning

- **PyTorch https://pytorch.org/**
- **TensorFlow https://www.tensorflow.org/**
- **Keras https://keras.io/**
- **Jax https://github.com/jax-ml/jax**

Higher-level APIs with training/evaluation helpers

- **Lightning https://lightning.ai/docs/pytorch/stable/**
- **FastAI https://github.com/fastai/fastai**
- **HuggingFace https://huggingface.co/**

Logging, visualization, hyperparameter sweeps

- **Weights & Biases (W&B) https://wandb.ai/site/**
- **Mlflow http://mlflow.org/**
- **Ray https://github.com/ray-project/ray**

Application Examples

- Layer 1: Physical Layer (PHY)
  - AI-Powered Beamforming [“Going beyond RF: A survey on how AI-enabled multimodal beamforming will shape the NextG standard”, Roy et al., 2023]
  - Channel Estimation and Prediction [“Deep Learning-Based Channel Estimation”, Soltani et al., 2019]
  - CSI feedback [“Deep Learning for Massive MIMO CSI Feedback”, Wen et al., 2018]
  - Modulation and Coding [“Machine Learning–Based Adaptive Modulation and Coding Design”, Luo et al., 2020]

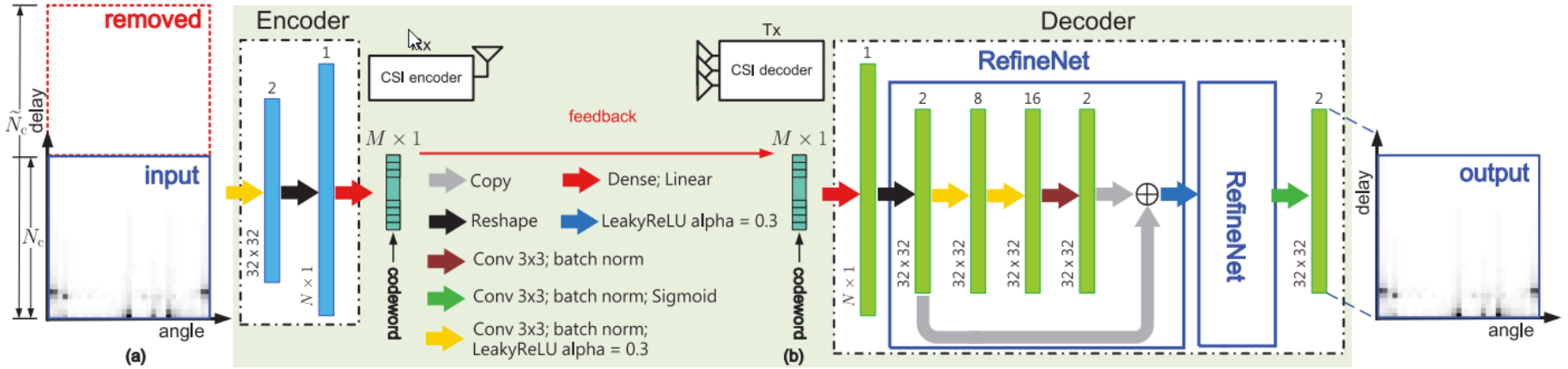
- Level 2: Data Link Layer (MAC)
  - Radio Resource Scheduling [“AI-Enabled Radio Resource Allocation in 5G for URLLC and eMBB Users”, Elsayed et al., 2019]
  - Dynamic Spectrum Access [“A Review on AI-Driven Dynamic Spectrum Access for Next-Generation Wireless Networks”, Lakouismi et al., 2025]
  - Multiple Transmission and Reception Point Management [“AI/ML for Beam Management in 5G-Advanced”, Xue et al., 2023]

- Layer 3: Network Layer and Above
  - Traffic Engineering & Routing [“Towards AI/ML-Driven Network Traffic Engineering”, Alam et al., 2024]
  - Network Slice Management [“Large Language Models meet Network Slicing Management and Orchestration”, Dandoush et al., 2024]
  - Mobility and radio link monitoring [“AI-Driven Mobility Management for High-Speed Railway Communications: Compressed Measurements and Proactive Handover”, Li et al., 2025]
  - Semantic Communications [“Generative AI-driven Semantic Communication Networks: Architecture, Technologies and Applications”, Liang et al., 2024]
  - Adaptive Video Streaming [“Generative AI for HTTP Adaptive Streaming”, Artioli et al, 2024]

Example: CSI Feedback

"Deep Learning for Massive MIMO CSI Feedback", Wen et al., IEEE Wireless Communication Letters 2018

- In frequency division duplexity (FDD) MIMO the downlink CSI is acquired at the user equipment (UE) during the training period and returns to the BS through feedback links.
  - Vector quantization or codebook-based approaches are adopted to reduce feedback overhead
  - feedback quantities need to be scaled linearly with the number of transmit antennas (prohibitive in a massive MIMO)
- Idea: use deep learning to learn a compressed representation of channel matrices
  - UE uses encoder NN compress, the BS uses decoder NN to decompress



Example: CSI Feedback

"Deep Learning for Massive MIMO CSI Feedback", Wen et al., IEEE Wireless Communication Letters 2018

- Data: two types of channel matrices generated through the COST 2100 channel model [Liu et al., 2012]. ULA with 32 antennas at the BS and 1024 subcarriers.
  - indoor picocellular scenario at the 5.3 GHz band
  - outdoor rural scenario at the 300 MHz band
  - training, validation, and test sets contain 100,000, 30,000, and 20,000 samples

- Evaluation compares against 3 popular compressive sensing (CS) based methods
  - Metrics

$$\text{NMSE} = \mathsf{E}\left\{\|\mathbf{H} - \widehat{\mathbf{H}}\|_2^2 / \|\mathbf{H}\|_2^2\right\} \qquad \rho = \mathsf{E}\left\{\frac{1}{\tilde{N}_c}\sum_{n=1}^{\tilde{N}_c}\frac{|\widehat{\tilde{\mathbf{h}}}_n^H \tilde{\mathbf{h}}_n|}{\|\widehat{\tilde{\mathbf{h}}}_n\|_2 \|\tilde{\mathbf{h}}_n\|_2}\right\}$$

TABLE I
NMSE IN dB AND COSINE SIMILARITY ρ

γ	Methods	Indoor		Outdoor	
		NMSE	ρ	NMSE	ρ
1/4	LASSO	-7.59	0.91	-5.08	0.82
	BM3D-AMP	-4.33	0.80	-1.33	0.52
	TVAL3	-14.87	0.97	-6.90	0.88
	CS-CsiNet	-11.82	0.96	-6.69	0.87
	CsiNet	**-17.36**	**0.99**	**-8.75**	**0.91**
1/16	LASSO	-2.72	0.70	-1.01	0.46
	BM3D-AMP	0.26	0.16	0.55	0.11
	TVAL3	-2.61	0.66	-0.43	0.45
	CS-CsiNet	-6.09	0.87	-2.51	0.66
	CsiNet	**-8.65**	**0.93**	**-4.51**	**0.79**
1/32	LASSO	-1.03	0.48	-0.24	0.27
	BM3D-AMP	24.72	0.04	22.66	0.04
	TVAL3	-0.27	0.33	0.46	0.28
	CS-CsiNet	-4.67	0.83	-0.52	0.37
	CsiNet	**-6.24**	**0.89**	**-2.81**	**0.67**
1/64	LASSO	-0.14	0.22	-0.06	0.12
	BM3D-AMP	0.22	0.04	25.45	0.03
	TVAL3	0.63	0.11	0.76	0.19
	CS-CsiNet	-2.46	0.68	-0.22	0.28
	CsiNet	**-5.84**	**0.87**	**-1.93**	**0.59**

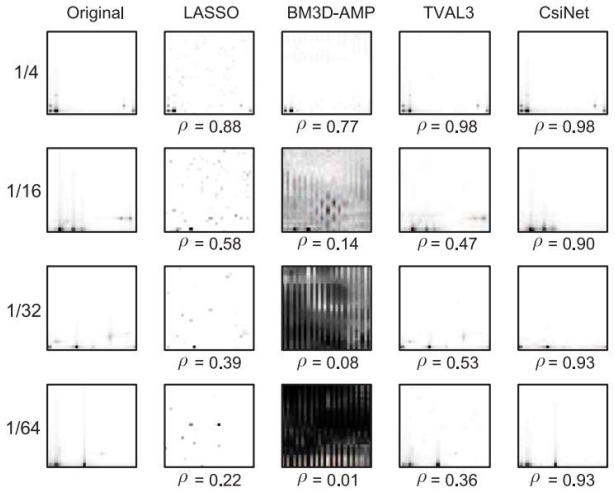


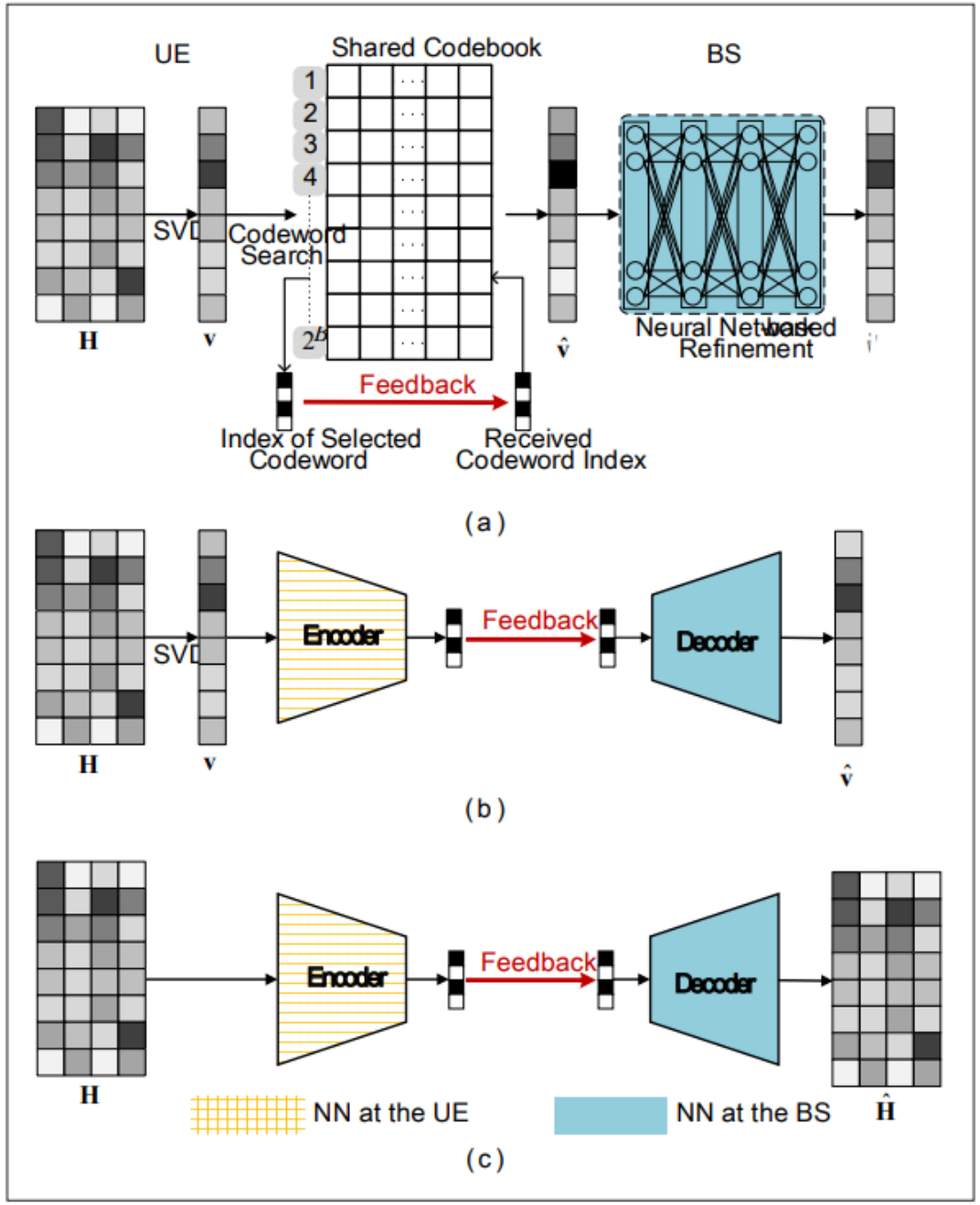
Fig. 2. Reconstruction images for different compression ratios by different algorithms in indoor picocellular scenarios.

Example: CSI Feedback

"AI for CSI Feedback Enhancement in 5G-Advanced", Guo et al., IEEE Wireless Communications 2024

Frameworks for AI for CSI Feedback:

- a) one-sided refinement for implicit CSI feedback
  - Requires no change to existing feedback framework
  - The pretrained NN refines the obtained codeword
- b) two-sided enhancement for implicit CSI feedback
  - Requires partial changes to existing framework
  - NN-based encoder is adopted at the UE to compress and quantize the generated precoding matrix
  - After obtaining the feedback bitstream, the NN-based decoder reconstructs the original precoding matrix
- c) two-sided enhancement for explicit CSI feedback
  - Requires changes to existing framework
  - feedbacks the entire downlink CSI, H, to the BS via an NN-based encoder
  - the decoder at the BS reconstructs the original CSI on the basis of the received bitstream
  - Larger feedback overhead



Example: CSI Feedback

"AI for CSI Feedback Enhancement in 5G-Advanced", Guo et al., IEEE Wireless Communications 2024

Normalized mean-squared error (NMSE) performance of the CS-based algorithms and the AI-enabled methods (CsiNet and TransNet)

Conventional algorithms perform much worse than the AI-enabled methods under all compression ratios

compression ratio	Methods	Indoor	Outdoor
1/4	LASSO	-7.59	-5.08
	BM3D-AMP	-4.33	-1.33
	TVAL3	-14.87	-6.90
	CsiNet	-17.36	-8.75
	TransNet	-32.38	-14.86
1/16	LASSO	-2.72	-1.01
	BM3D-AMP	0.26	0.55
	TVAL3	-2.61	-0.43
	CsiNet	-8.65	-4.51
	TransNet	-15.00	-7.82
1/32	LASSO	-1.03	-0.24
	BM3D-AMP	24.72	22.66
	TVAL3	-0.27	0.46
	CsiNet	-6.24	-2.81
	TransNet	-10.49	-4.13
1/64	LASSO	-0.14	-0.06
	BM3D-AMP	0.22	25.45
	TVAL3	0.63	0.76
	CsiNet	-5.84	-1.93
	TransNet	-6.08	-2.62

TABLE 2. NMSE dB performance comparison of different explicit CSI feedback algorithms [6, 11].

Part 4

Foundation Model Design

- The Transformer model
- Foundation Model paradigm
- Multimodal Foundation Models
- Foundation Model training

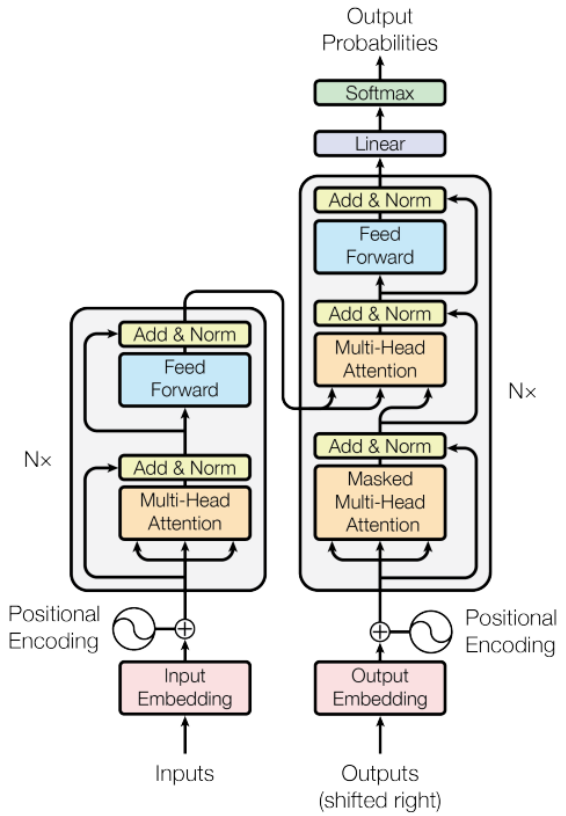
The Transformer Model

- Neural network architecture designed for sequential data
- It is the base of all modern Large Language Models (LLMs) like ChatGPT, Claude, DeepSeek, etc.
- Main architecture for all Foundation Models

- Properties:
  - Allows in input sequences of different lengths
  - Efficiency: processes the entire sequence in parallel during training

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

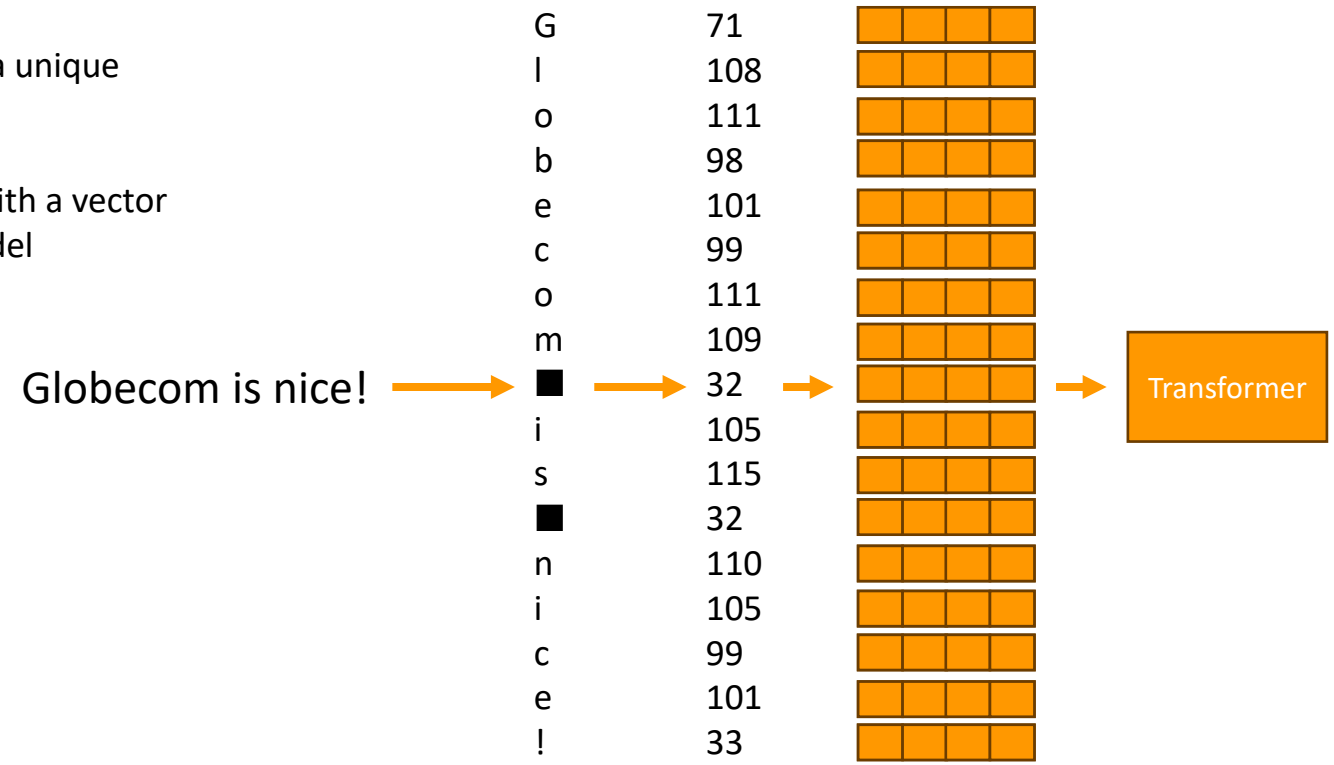


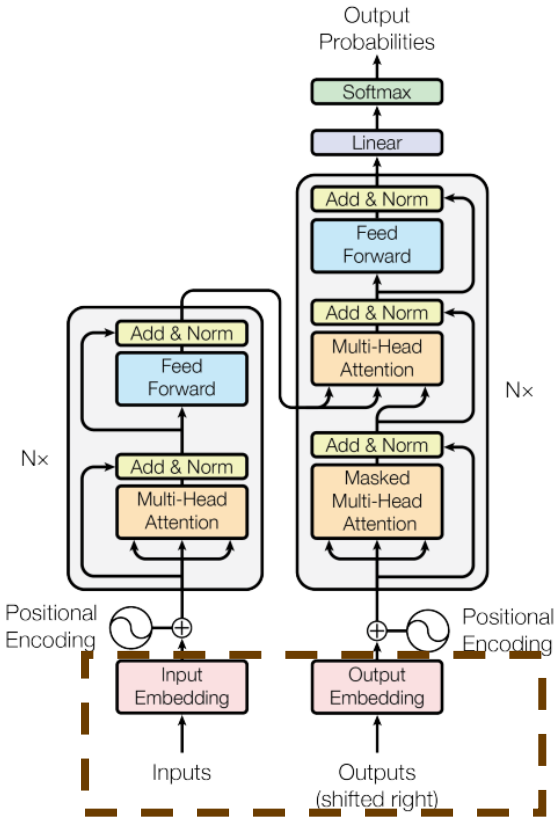
Tokenization

- Transformer models take a sequence, in which each element is a vector, as input
- Tokenization is the process of converting raw input data from any domain (such as text, images, audio, or other modalities) into a structured sequence of discrete units called tokens
  - From raw data to model-readable data

For example, in text we can assign a unique Identifier to each character

Each identifier is then associated with a vector which is passed as input to the model



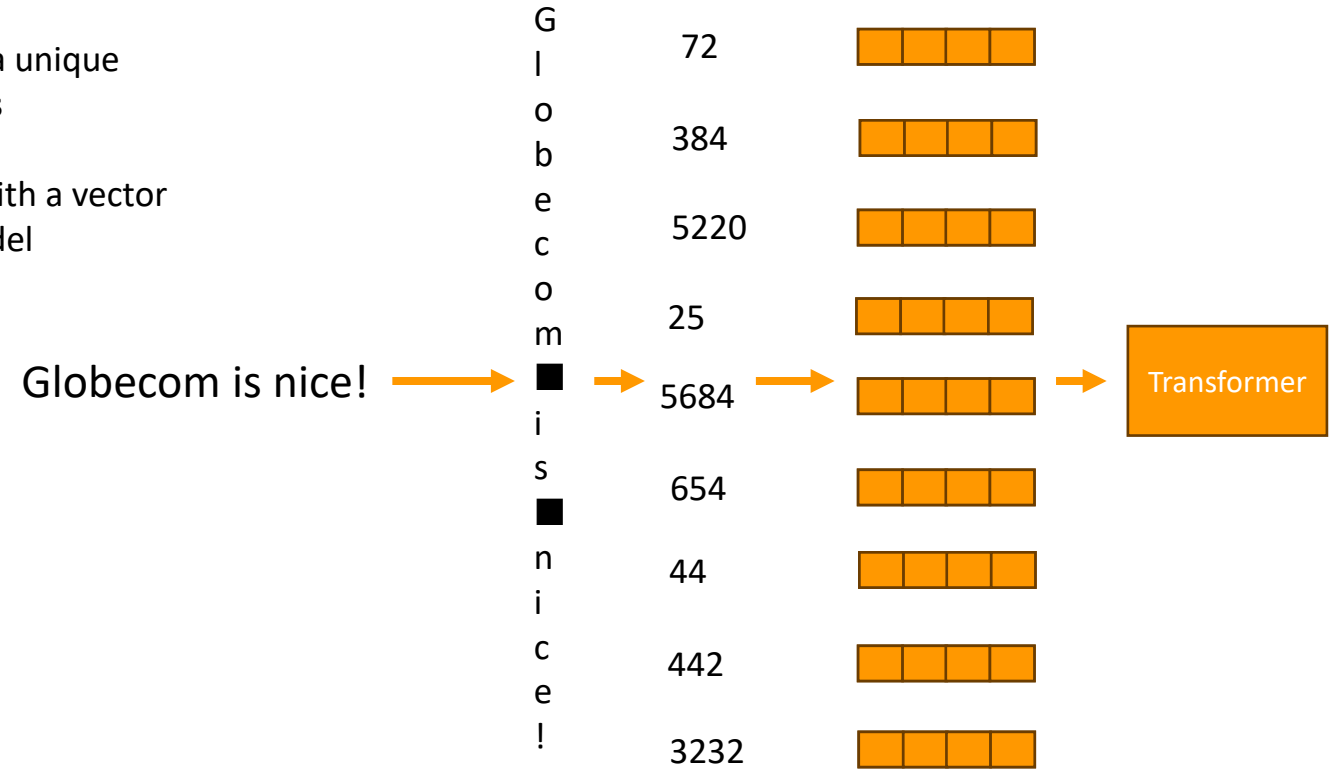


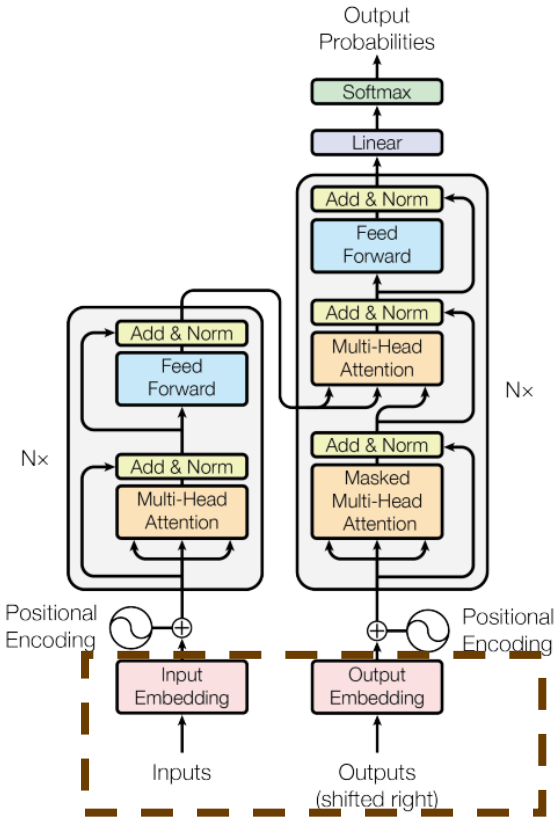
Tokenization

- Transformer models take a sequence, in which each element is a vector, as input
- Tokenization is the process of converting raw input data from any domain (such as text, images, audio, or other modalities) into a structured sequence of discrete units called tokens
  - From raw data to model-readable data

For example, in text we can assign a unique Identifier to each **pair of characters**

Each identifier is then associated with a vector which is passed as input to the model



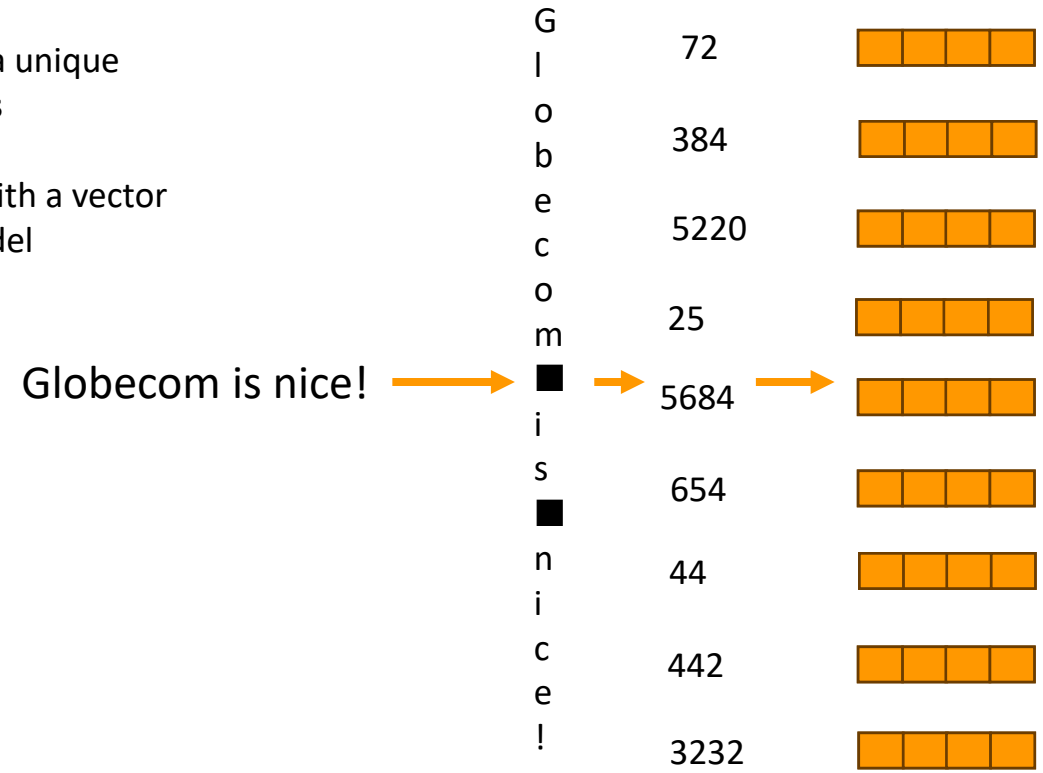


Tokenization

- Transformer models take a sequence, in which each element is a vector, as input
- Tokenization is the process of converting raw input data from any domain (such as text, image sequence of discrete units called tokens
  - From raw data to model-readable data

For example, in text we can assign a unique Identifier to each **pair of characters**

Each identifier is then associated with a vector which is passed as input to the model



The same sentence is now represented with half the number of tokens, but the size of our vocabulary has increased quadratically!

In general there is a tradeoff:

- Smaller vocabularies
  - less memory
  - more granular information
  - typically easier to train (each token appears more often)
  - more compute to process the same amount of information
- Larger vocabularies
  - more information in less tokens (i.e., less compute)
  - Harder to train
  - less granularity
  - require more memory

Positional Embeddings

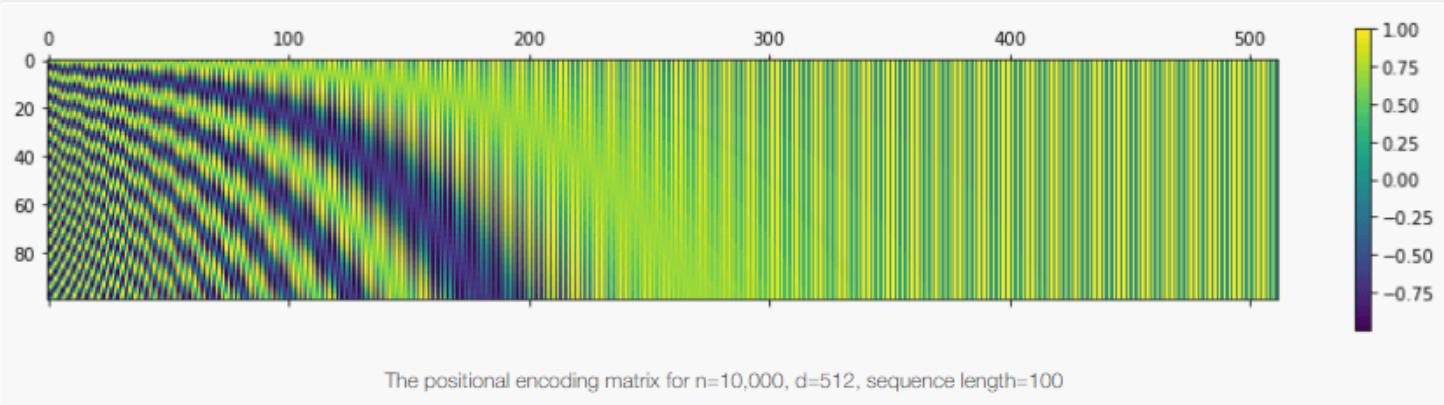
- Formally, the Transformer is a set-to-set architecture
  - Receives a set as input and produces a set of the same size as output

- Sets are unordered by definition -> we need to add extra information to let the transformer know about order between tokens

- A vector PE is added to each token

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



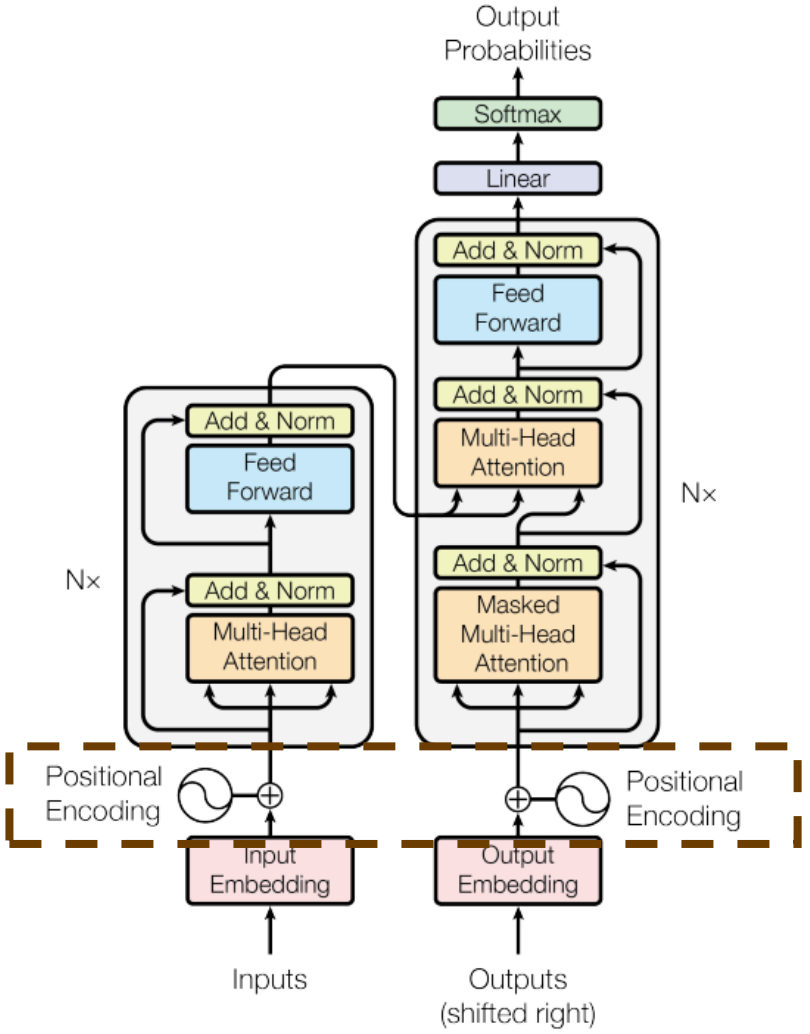


Image from: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Positional Embeddin

- The Transformer is a set-to-set architecture
  - Receives a set as input and produces a set of the same size as
- Sets are unordered by definition -> we need to add ext
  know about order between tokens
- A vector PE is added to each token

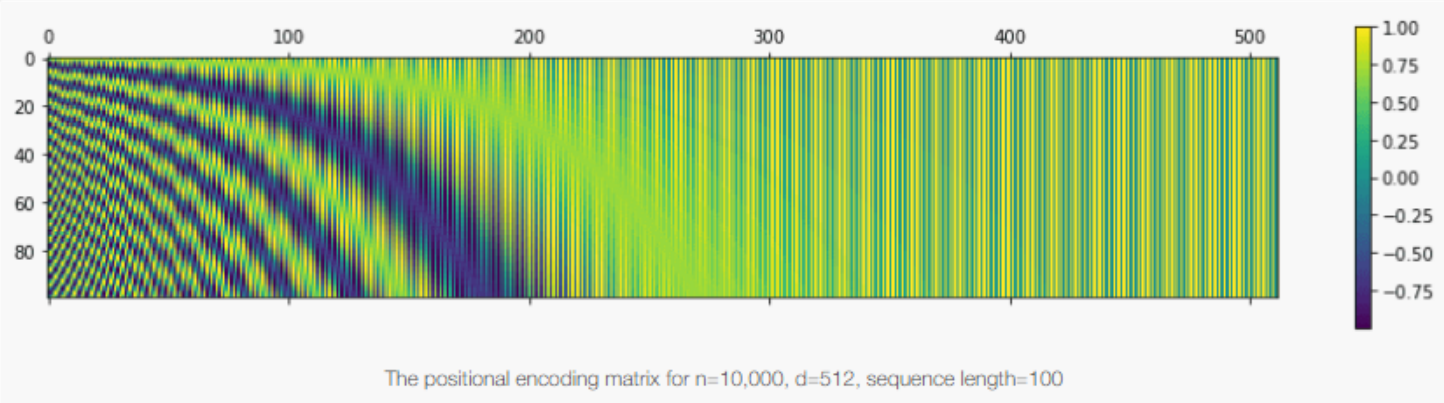
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Why this approach?

- Values are "normalized" (within -1 and 1)
- Each position is encoded uniquely
- It maintains relative distance between positions (the angle between any two pairs of PE with the same distance is the same)
- For any offset k, the PE of positions p+k is a linear function of p (we hope this is easy for models to learn) – this comes from trigonometric identities:

$$\sin(a+b) = \cos(a)\cos(b) - \sin(a)\sin(b)$$



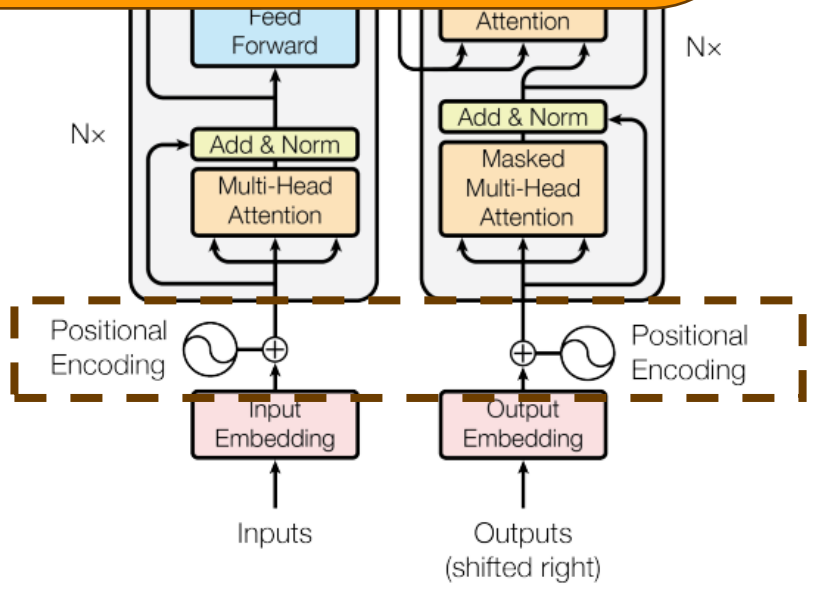


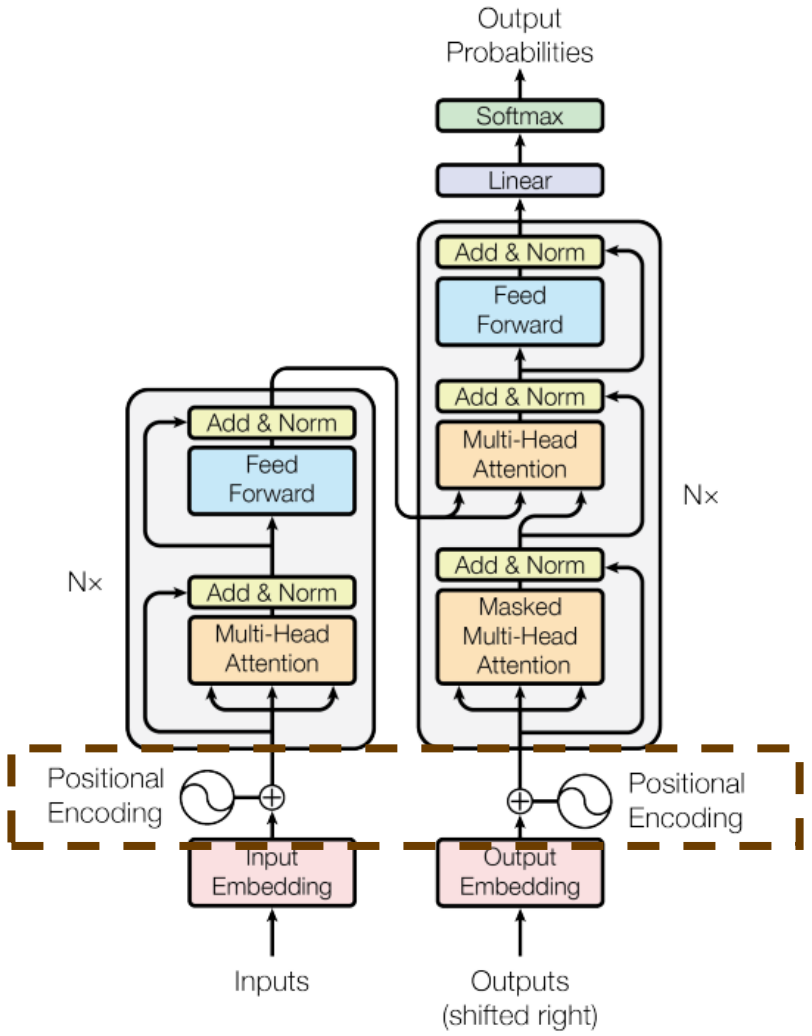
Image from: https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/

Positional Embeddings

- More modern approach: RoPE [“RoFormer: Enhanced Transformer with Rotary Position Embedding”, Su et al., 2023]
- Does not add values to the embeddings, but instead rotates them during the attention computations

$$\boldsymbol{q}_m^\intercal \boldsymbol{k}_n = (\boldsymbol{R}^d_{\Theta,m}\boldsymbol{W}_q\boldsymbol{x}_m)^\intercal(\boldsymbol{R}^d_{\Theta,n}\boldsymbol{W}_k\boldsymbol{x}_n) = \boldsymbol{x}^\intercal\boldsymbol{W}_q R^d_{\Theta,n-m}\boldsymbol{W}_k\boldsymbol{x}_n$$

$$\boldsymbol{R}^d_{\Theta,m} = \begin{pmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{pmatrix}$$



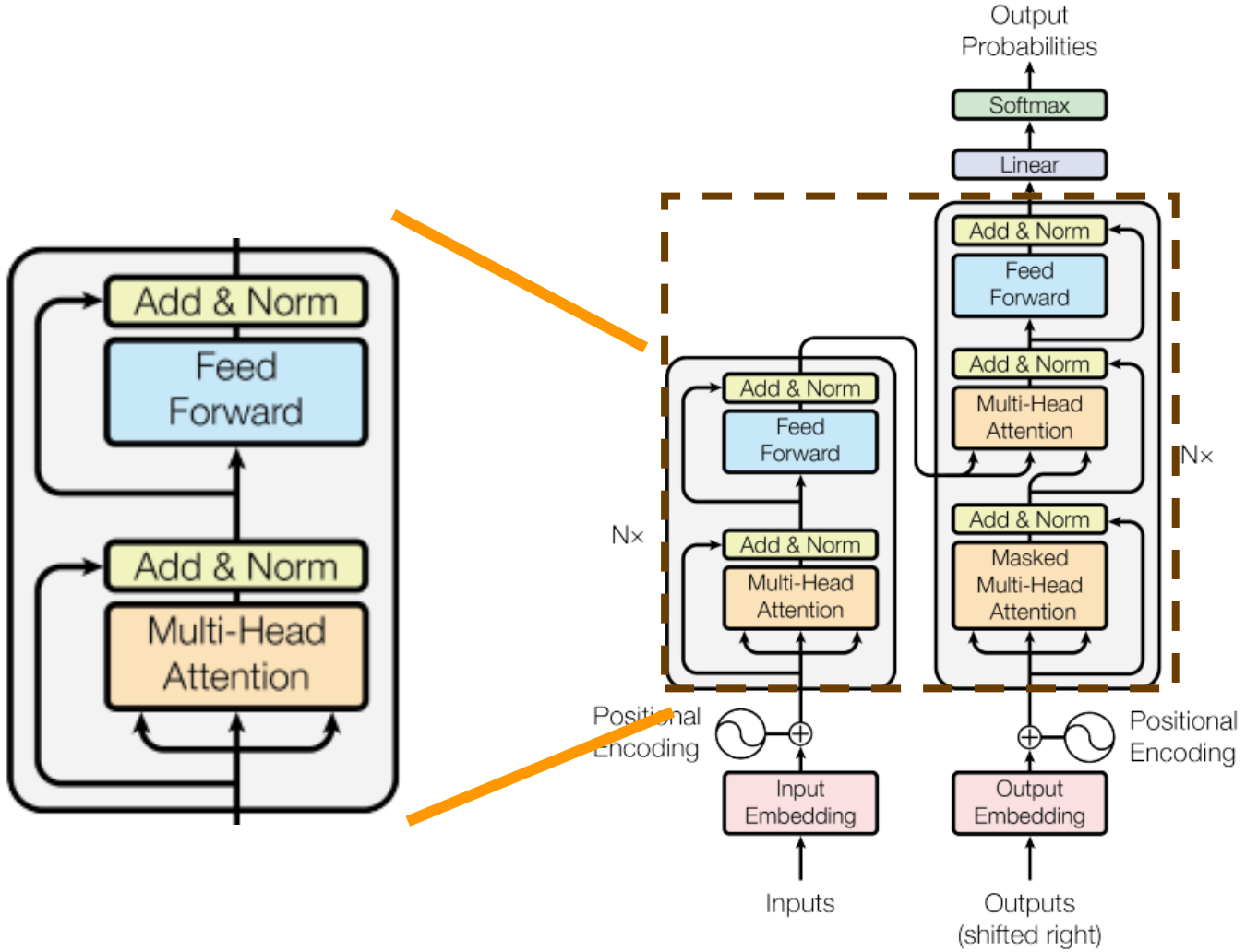
Transformer Block

Transformer blocks are perform the following operations

- $x_{att} = attention(x)$
- $x_{skip} = x + x_{att}$
- $x_{norm1} = normalize(x_{skip})$

- $x_{ff} = MLP(x_{norm1})$
- $x_{skip2} = x_{norm1} + x_{ff}$
- $x_{norm2} = normalize(x_{skip2})$

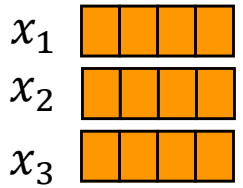
- The main novelty is the introduction of *Attention*



Attention

- The key mechanism in Transformers
- Allows the model to capture dependencies between input elements

- Input $X \in \mathbb{R}^{n \times d}$

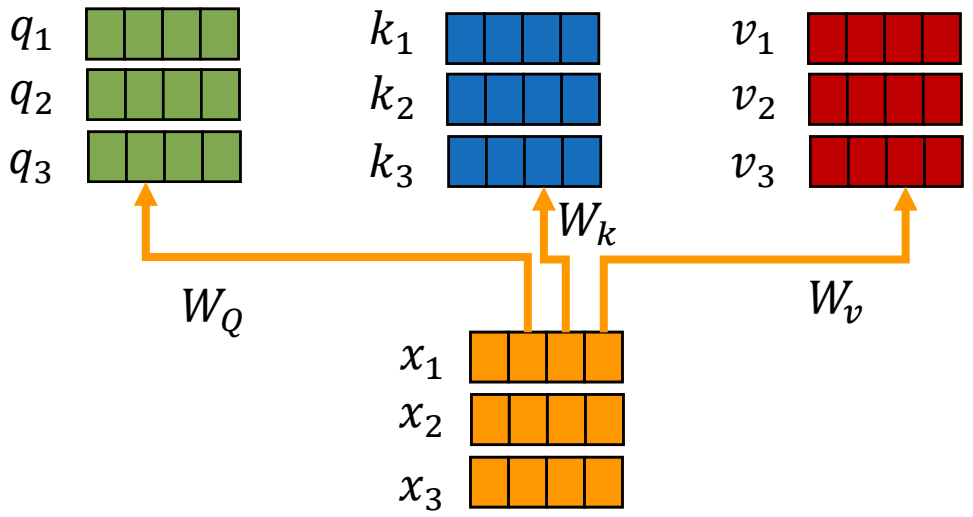


Attention

- The key mechanism in Transformers
- Allows the model to capture dependencies between input elements

- Input $X \in \mathbb{R}^{n \times d}$
- For each input element, a query, a key, and a value is created

$$Q = XW_Q \qquad K = XW_K \qquad V = XW_V$$



Attention

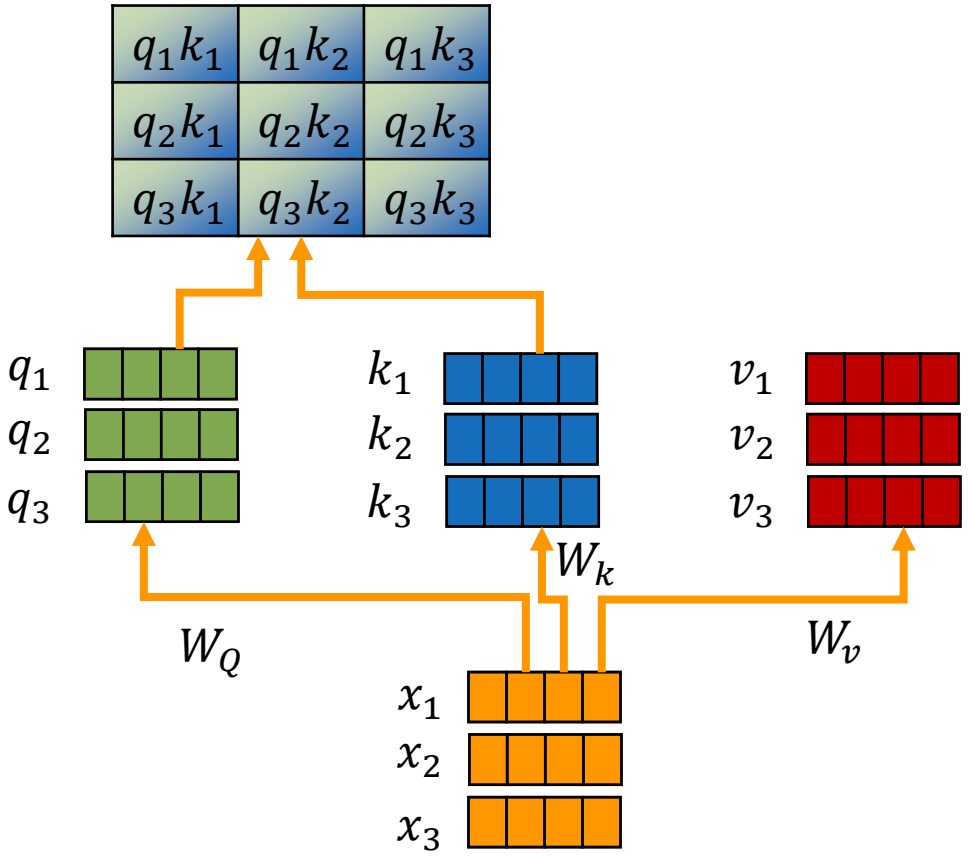
- The key mechanism in Transformers
- Allows the model to capture dependencies between input elements

- Input $X \in \mathbb{R}^{n \times d}$
- For each input element, a query, a key, and a value is created

$$Q = XW_Q \qquad K = XW_K \qquad V = XW_V$$

- Each query is compared to each key (with a dot product)

$$QK^T$$

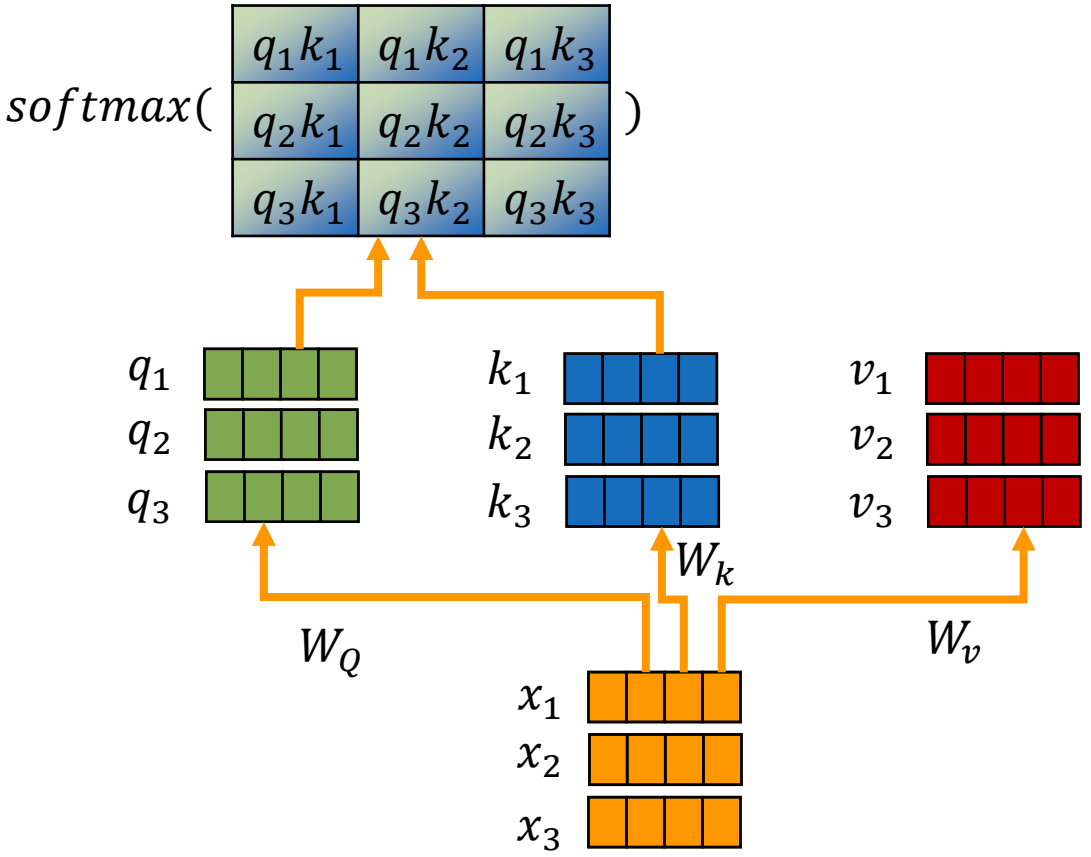


Attention

$$softmax(x)_i = \frac{x_i}{\sum_j x_j}$$

- The key mechanism in Transformers
- Allows the model to capture dependencies between input elements

- Input $X \in \mathbb{R}^{n \times d}$
- For each input element, a query, a key, and a value is created
  $$Q = XW_Q \qquad K = XW_K \qquad V = XW_V$$
- Each query is compared to each key (with a dot product)
  $$QK^T$$
- Normalize each row so that is sums to 1 to obtain the attention weights
  $$softmax(QK^T)$$

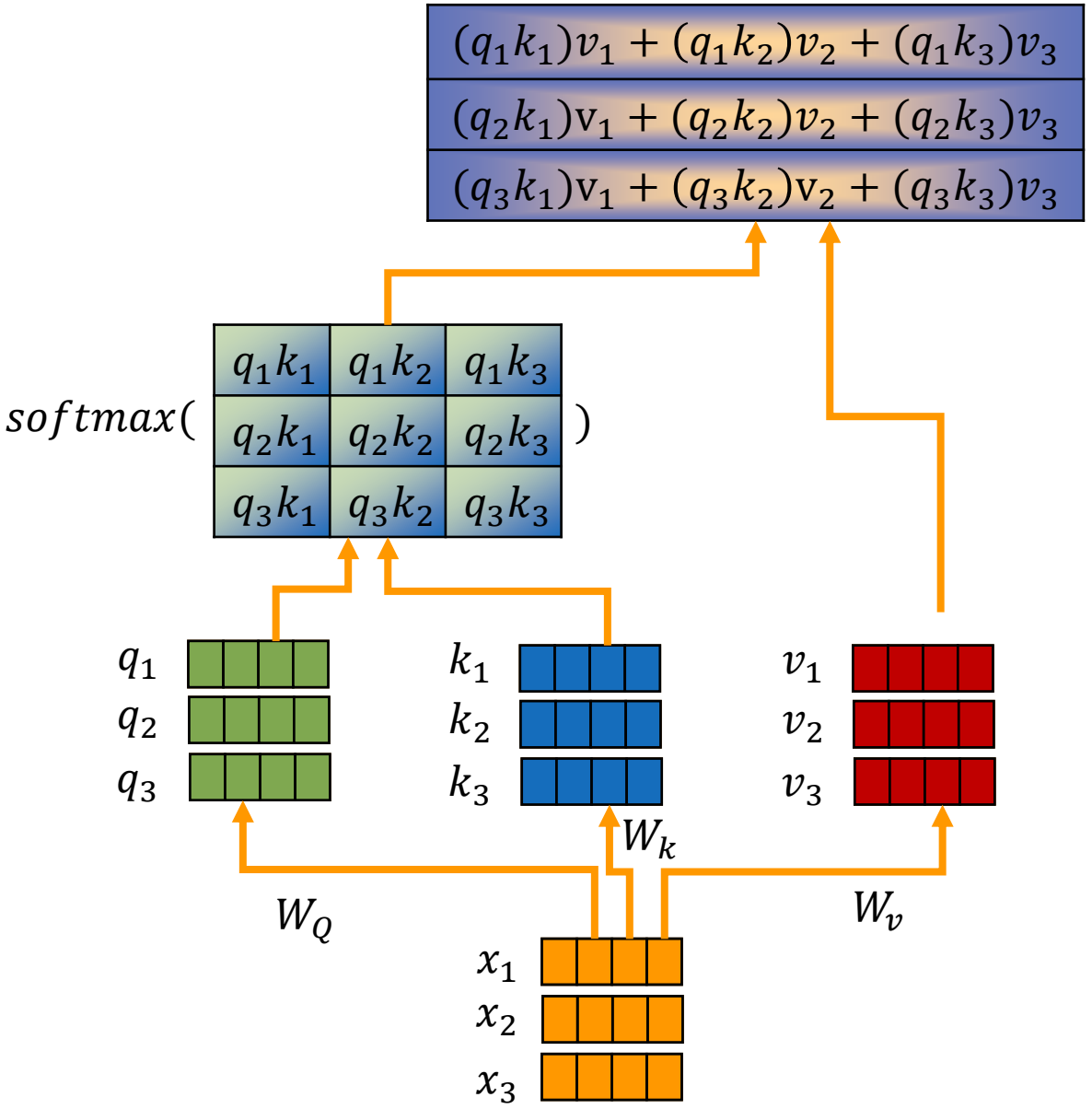


Attention

- The key mechanism in Transformers
- Allows the model to capture dependencies between input elements

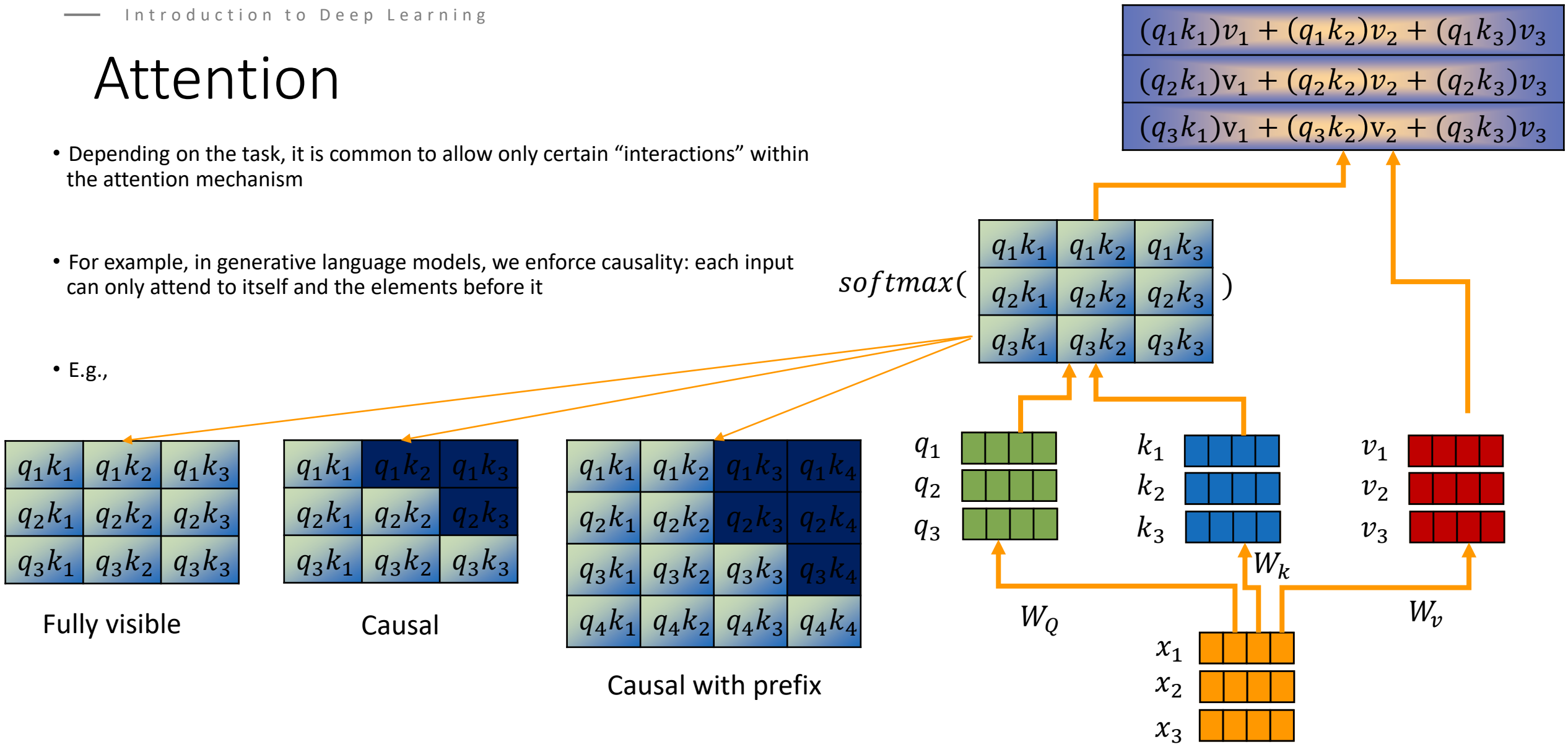
- Input $X \in \mathbb{R}^{n \times d}$
- For each input element, a query, a key, and a value is created
  $Q = XW_Q \quad K = XW_K \quad V = XW_V$
- Each query is compared to each key (with a dot product)
  QK^T
- Normalize each row so that is sums to 1 to obtain the attention weights
  $softmax(QK^T)$
- Combine the values weighted by their attention weight
  $softmax(QK^T)V$

$$Attention\ (X) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$



Attention

- Depending on the task, it is common to allow only certain “interactions” within the attention mechanism
- For example, in generative language models, we enforce causality: each input can only attend to itself and the elements before it
- E.g.,



Implementing a Transformer Block in Pytorch

- nn.Module is a PyTorch base class for neural network modules
  - The parameters and submodules should be defined in __init__
  - It requires a forward method, that specifies how the output is produced given the input
- PyTorch will automatically register the operations that are performed to the input inside the forward method and a simple call to .backward will populate the .grad field of all the parameters of the network (performing backpropagation)
- PyTorch provides ready to use implementations of most of popular neural network modules

```python
import torch
import torch.nn as nn

class SimpleTransformerEncoder(nn.Module):
    def __init__(self, d_model=64, nhead=4,
dim_feedforward=128):
        super().__init__()
        self.self_attn = nn.MultiheadAttention(d_model, nhead)
        self.linear1 = nn.Linear(d_model, dim_feedforward)
        self.linear2 = nn.Linear(dim_feedforward, d_model)
        self.norm1 = nn.LayerNorm(d_model)
        self.norm2 = nn.LayerNorm(d_model)
        self.relu = nn.ReLU()

    def forward(self, src):
        # src shape: (seq_len, batch_size, d_model)
        attn_output, _ = self.self_attn(src, src, src)
        src = self.norm1(src + attn_output)
        ff_output = self.linear2(self.relu(self.linear1(src)))
        src = self.norm2(src + ff_output)
        return src
```

Conv1d	MaxPool1d	LogSigmoid	BatchNorm1d	RNN
Conv2d	MaxPool2d	MultiheadAttention	BatchNorm2d	LSTM
Conv3d	MaxPool3d	PReLU	BatchNorm3d	GRU
ConvTranspose1d	MaxUnpool1d	ReLU	LazyBatchNorm1d	RNNCell
ConvTranspose2d	MaxUnpool2d	ReLU6	LazyBatchNorm2d	LSTMCell
ConvTranspose3d	MaxUnpool3d	RReLU	LazyBatchNorm3d	GRUCell
LazyConv1d	AvgPool1d	SELU	GroupNorm	Transformer
LazyConv2d	AvgPool2d	CELU	SyncBatchNorm	TransformerEncoder
LazyConv3d	AvgPool3d	GELU	InstanceNorm1d	TransformerDecoder
LazyConvTranspose1d	FractionalMaxPool2d	Sigmoid	InstanceNorm2d	TransformerEncoderLayer
LazyConvTranspose2d	FractionalMaxPool3d	SiLU	InstanceNorm3d	TransformerDecoderLayer
LazyConvTranspose3d	LPPool1d	Mish	LazyInstanceNorm1d	
	LPPool2d	Softplus	LazyInstanceNorm2d	
	LPPool3d		LazyInstanceNorm3d	

Implementing a Transformer Block in Pytorch

- nn.Module is a PyTorch base class for neural network modules
  - The parameters and submodules should be defined in __init__
  - It requires a forward method, that specifies how the output is produced given the input

- PyTorch will automatically register the operations that are performed to the input inside the forward method and a simple call to .backward will populate the .grad field of all the parameters of the network (performing backpropagation)

- PyTorch provides ready to use implementations of most of popular neural network modules

Conv1d	MaxPool1d	LogSigmoid	BatchNorm1d	RNN
Conv2d	MaxPool2d	MultiheadAttention	BatchNorm2d	LSTM
Conv3d	MaxPool3d	PReLU	BatchNorm3d	GRU
ConvTranspose1d	MaxUnpool1d	ReLU	LazyBatchNorm1d	RNNCell
ConvTranspose2d	MaxUnpool2d	ReLU6	LazyBatchNorm2d	LSTMCell
ConvTranspose3d	MaxUnpool3d	RReLU	LazyBatchNorm3d	GRUCell
LazyConv1d	AvgPool1d	SELU	GroupNorm	Transformer
LazyConv2d	AvgPool2d	CELU	SyncBatchNorm	TransformerEncoder
LazyConv3d	AvgPool3d	GELU	InstanceNorm1d	TransformerDecoder
LazyConvTranspose1d	FractionalMaxPool2d	Sigmoid	InstanceNorm2d	TransformerEncoderLayer
LazyConvTranspose2d	FractionalMaxPool3d	SiLU	InstanceNorm3d	TransformerDecoderLayer
LazyConvTranspose3d	LPPool1d	Mish	LazyInstanceNorm1d	
	LPPool2d	Softplus	LazyInstanceNorm2d	
	LPPool3d		LazyInstanceNorm3d	

PyTorch also has a built-in Implementation

Transformer

```
class torch.nn.Transformer(d_model=512, nhead=8, num_encoder_layers=6, num_decoder_layers=6,
dim_feedforward=2048, dropout=0.1, activation=<function relu>, custom_encoder=None,
custom_decoder=None, layer_norm_eps=1e-05, batch_first=False, norm_first=False, bias=True,
device=None, dtype=None)
```

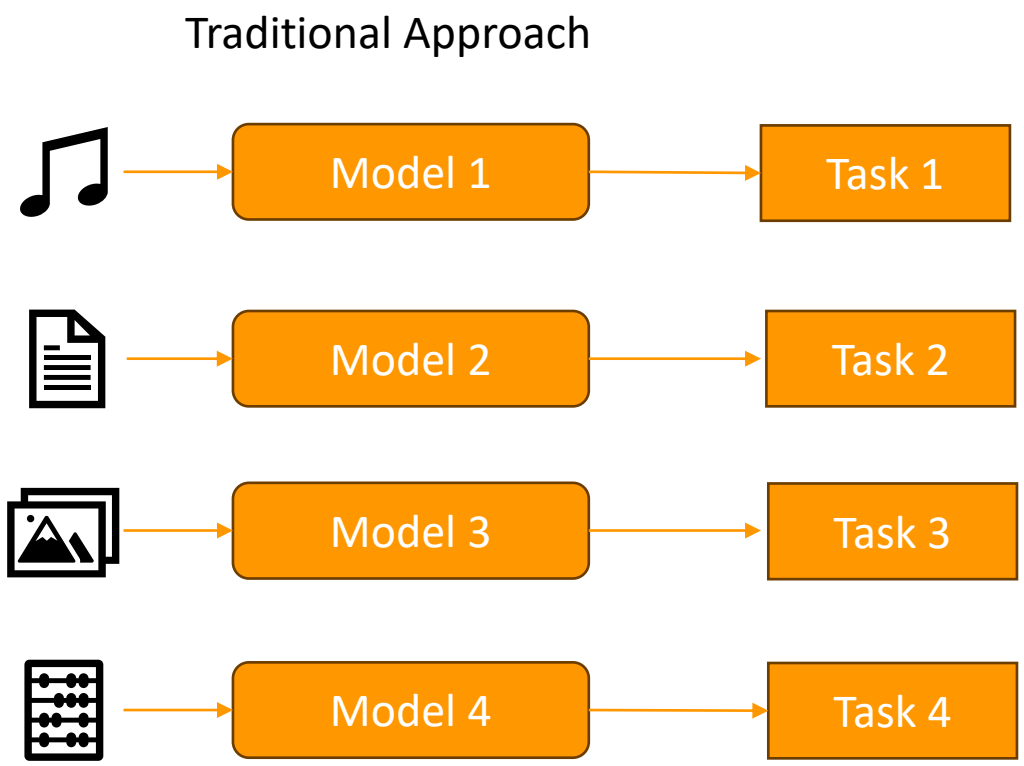
[source]

A basic transformer layer.

This Transformer layer implements the original Transformer architecture described in the Attention Is All You Need paper. The intent of this layer is as a reference implementation for foundational understanding and thus it contains only limited features relative to newer Transformer architectures. Given the fast pace of innovation in transformer-like architectures, we recommend exploring this tutorial to build an efficient transformer layer from building blocks in core or using higher level libraries from the PyTorch Ecosystem.

Before Foundation Models

- The traditional approach of training neural networks involves specific models for specific tasks

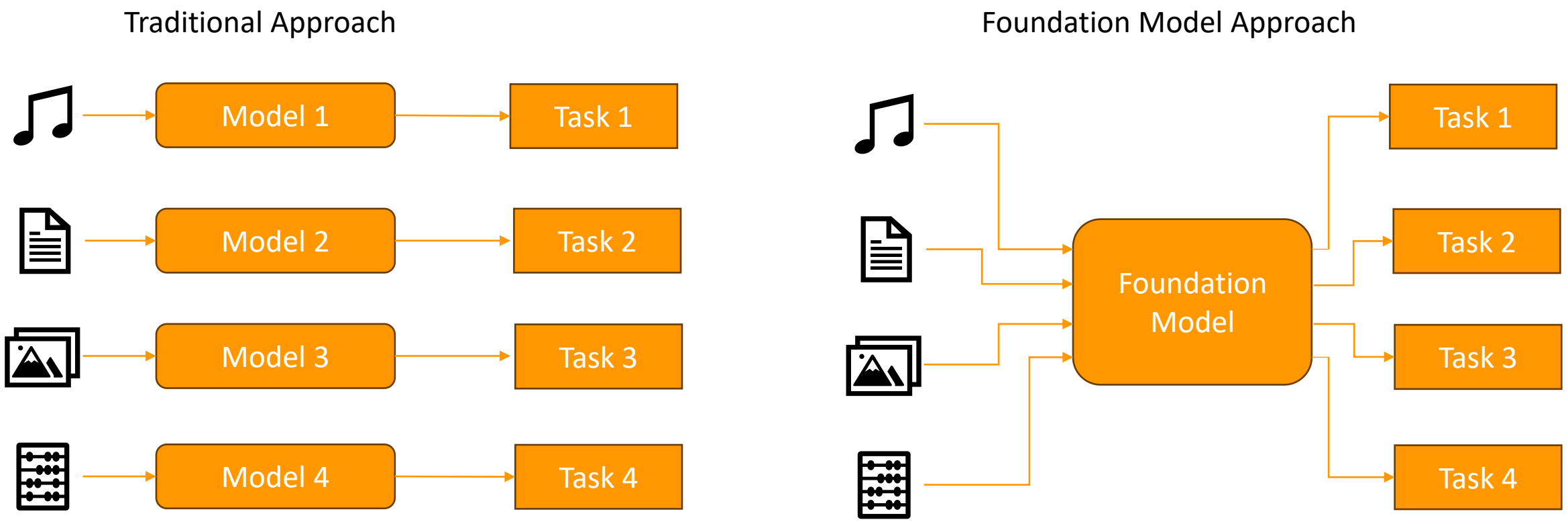


Drawbacks

- Each task requires ad-hoc model design
- Large amounts of data need to be labelled for each task
  - This can be very expensive/time consuming
- Very difficult to use a model for a task different from the one it was trained on

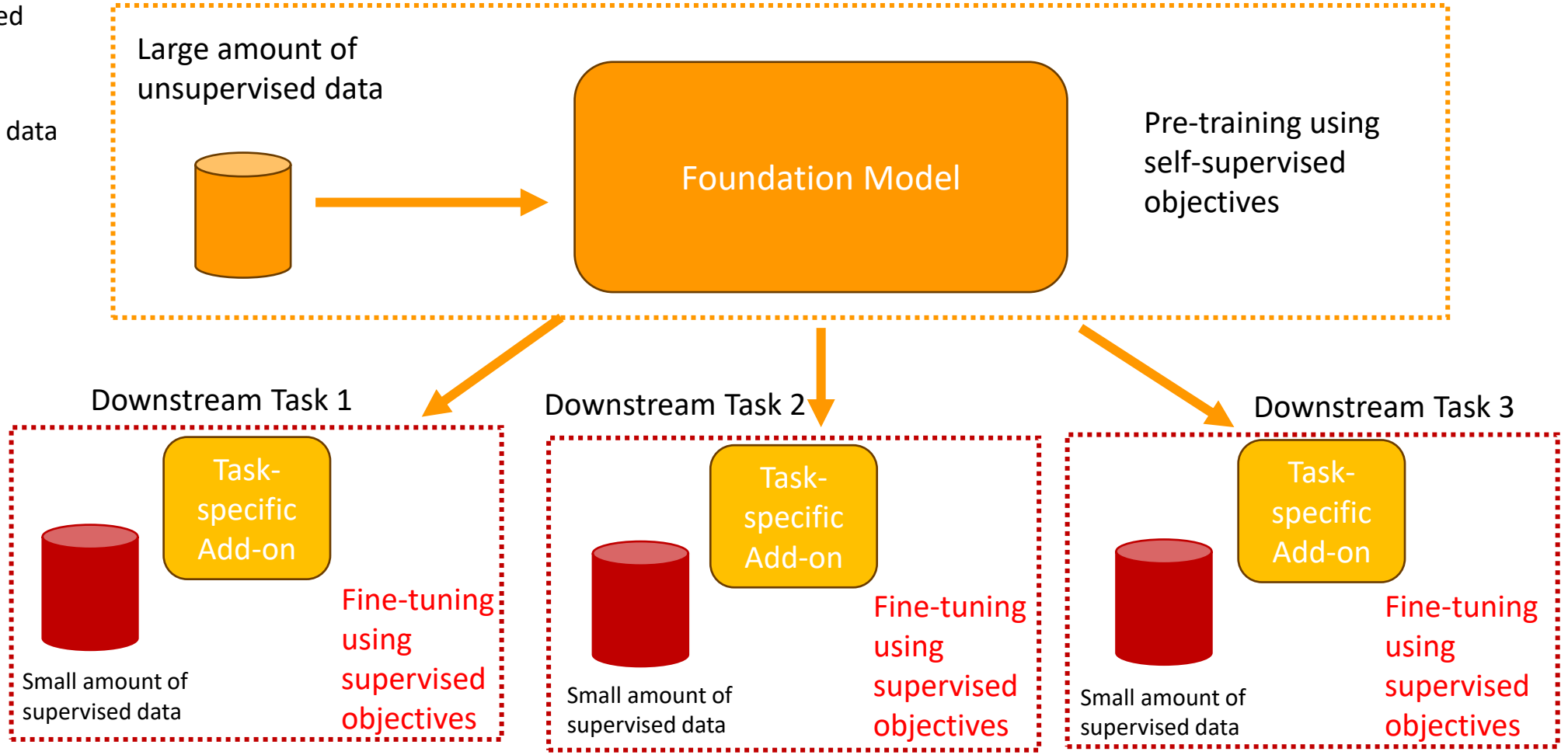
Foundation Models

- Models trained on vast datasets that can be adapted for a wide range of tasks
- They act as a foundational base for building more specialized AI applications



Foundation Models

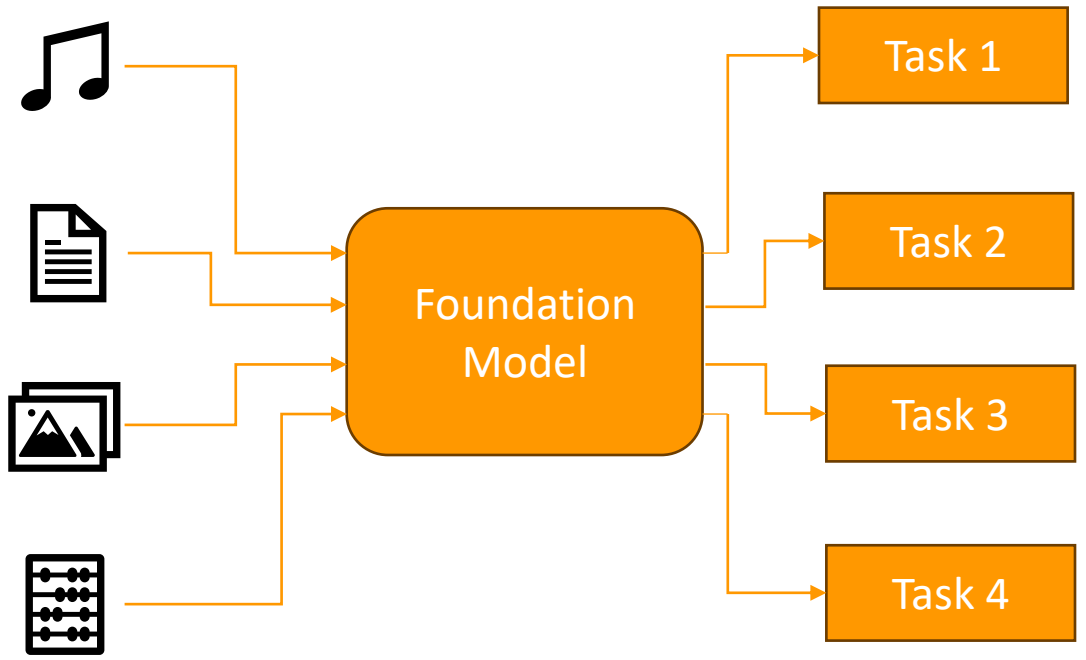
- Foundation models are trained with large amounts of unsupervised data
- Small amounts of supervised data are used for later tasks



Designing Foundation Models

- The transformer architecture is the “processing core”, but what else do we need?

- Input
  - How to provide data to the transformer
- Modality fusion
  - How to combine multiple modalities
- Training procedure
- Downstream task adaptation
  - How to use the model to perform downstream tasks



Input Format

- Transformers take as input a set of embeddings
- Tokenization is the process of turning the raw input into a structured sequence of discrete units called tokens
- How do we tokenize data other than text?

Images

- Split into non-overlapping patches
- Flatten
- Learnable linear projection

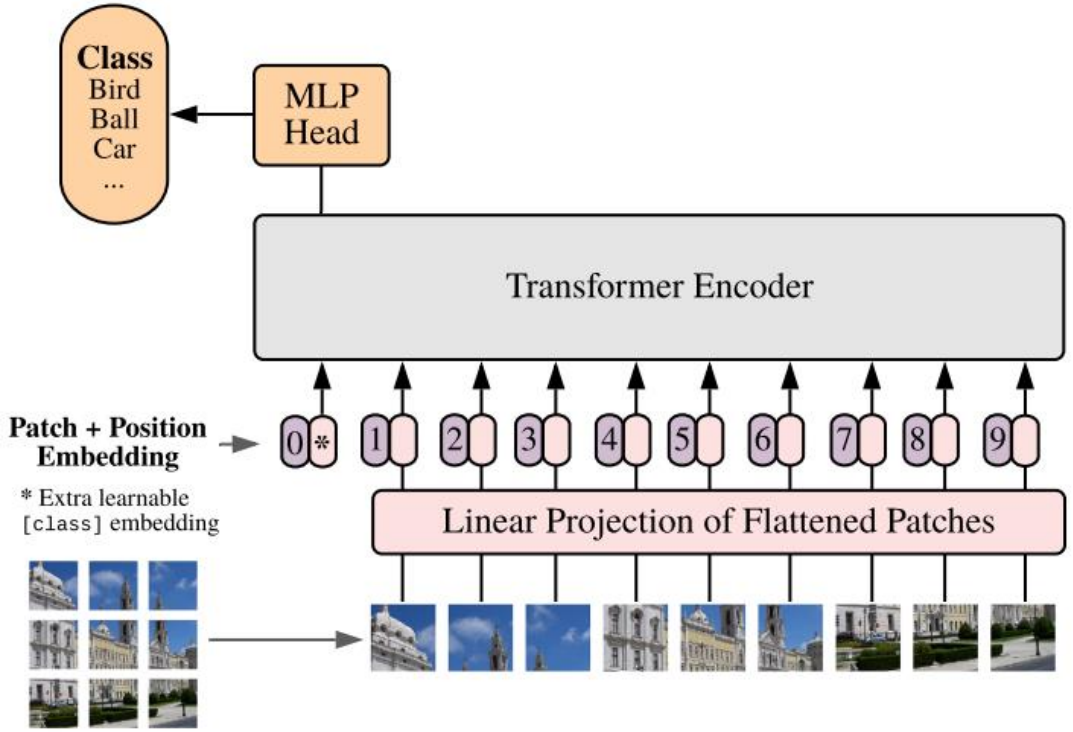


Image from: [“An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, Dosovitskiy et al., ICLR 2021]

Input Format

- Transformers take as input a set of embeddings
- Tokenization is the process of turning the raw input into a structured sequence of discrete units called tokens
- How do we tokenize data other than text?

Audio

- audio is re-sampled to 16,000 Hz
- 80-channel log-magnitude Mel spectrogram representation is computed on 25-millisecond windows with a stride of 10 milliseconds
- globally scale the input to be between -1 and 1 with approximately zero mean across the pre-training dataset
- encoder processes this input representation with a small stem consisting of two convolution layers with a filter width of 3 and the GELU activation function where the second convolution layer has a stride of two

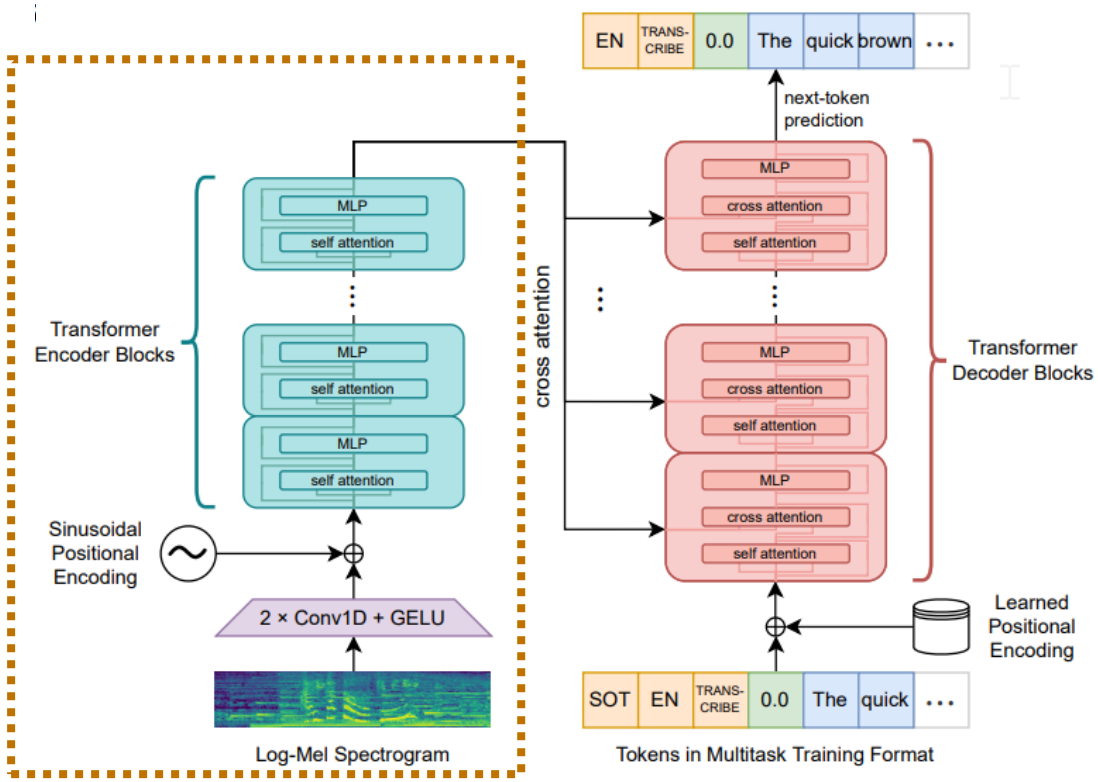


Image from: [“Robust Speech Recognition via Large-Scale Weak Supervision”, Radford et al., ICML 2023]

Input Format

- Transformers take as input a set of embeddings
- Tokenization is the process of turning the raw input into a structured sequence of discrete units called tokens
- How do we tokenize data other than text?

Time Series

- break the input into contiguous non-overlapping patches
- each patch is processed by a Residual Block

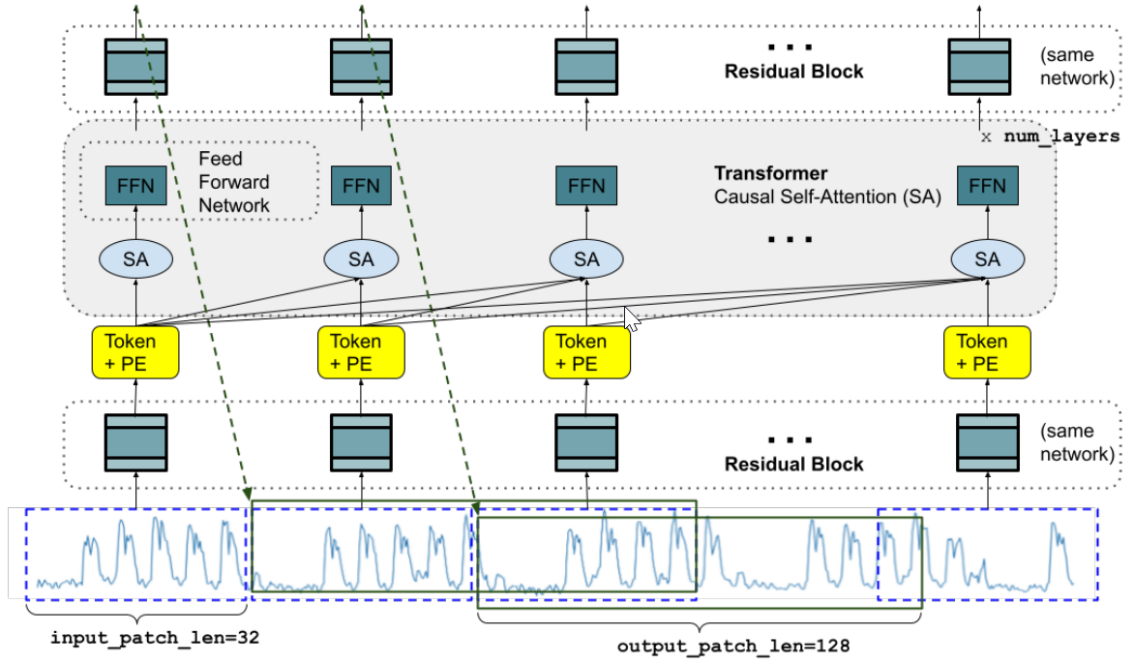


Image from: [“A decoder-only foundation model for time-series forecasting”, Das et al., ICML 2024]

Multimodal Transformers

- Many real-world tasks require understanding across modalities (e.g., image captioning, video QA, speech-to-text)

- We want to extend transformer architecture to handle heterogeneous inputs
- Examples of tasks:
  - Image ↔ Text: Captioning, visual question answering (VQA)
  - Video ↔ Text: Video summarization, action recognition
  - Audio ↔ Text: Speech recognition, audio captioning
  - Multi-modal fusion: Robotics, medical diagnosis

- Challenges
  - How to combine different modalities?
  - How to align the modalities?
    - They may have different frequencies

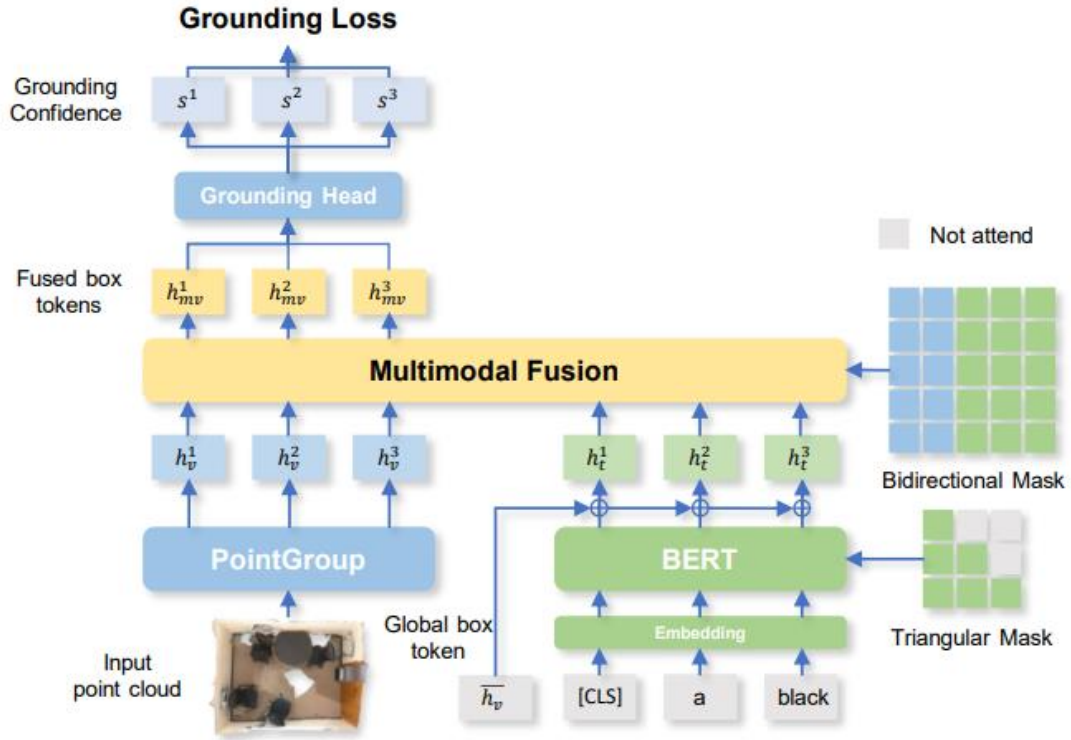


Image from “UniT3D: A Unified Transformer for 3D Dense Captioning and Visual Grounding”, Chen et al.

Multimodal Transformers

Key Challenges

- Data alignment: Matching modalities in time/space (e.g., aligning video frames with text).
- Data scarcity: Paired multi-modal datasets are limited compared to single-modality datasets.
- Modality imbalance: One modality may dominate learning if not balanced.
- Computational cost: Large models + multiple encoders = high memory and training time.
- Noise & missing modalities: Real-world data may have incomplete or noisy inputs.
- Cross-modal generalization: Ensuring the model can handle unseen modality combinations.

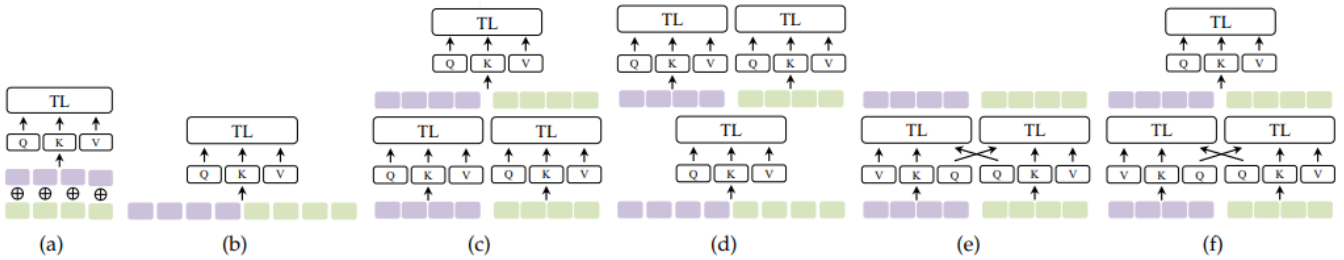


Fig. 2. Transformer-based cross-modal interactions: (a) Early Summation, (b) Early Concatenation, (c) Hierarchical Attention (multi-stream to one-stream), (d) Hierarchical Attention (one-stream to multi-stream), (e) Cross-Attention, and (f) Cross-Attention to Concatenation. "Q": Query embedding; "K": Key embedding; "V": Value embedding. "TL": Transformer Layer. Best viewed in colour.

Image from: ["Multimodal Learning with Transformers: A Survey", Xu et al., IEEE TPAMI 2023]

Image from: ["A survey on multimodal large language models", Yin et al., National Science Review 2024]

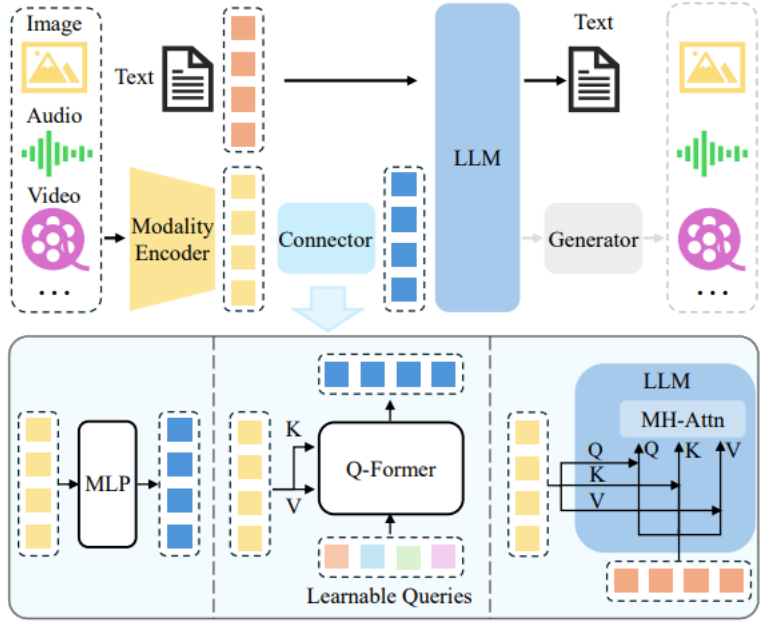


Fig. 2: An illustration of typical MLLM architecture. It includes an encoder, a connector, and a LLM. An optional generator can be attached to the LLM to generate more modalities besides text. The encoder takes in images, audios or videos and outputs features, which are processed by the connector so that the LLM can better understand. There are broadly three types of connectors: projection-based, query-based, and fusion-based connectors. The former two types adopt token-level fusion, processing features into tokens to be sent along with text tokens, while the last type enables a feature-level fusion inside the LLM.

Image from: [“A survey on multimodal large language models”, Yin et al., National Science Review 2024]

Multimodal Transformers

Combination mechanisms typically involve having modality-specific encoders, a fusion mechanism, and an alignment mechanism

- Modality-specific encoders:
  - Vision: CNNs, Vision Transformers (ViT)
  - Text: BERT-like language models
  - Audio: Spectrogram-based CNNs or audio transformers

- Fusion mechanisms:
  - Early fusion: Combine raw features before transformer layers.
  - Late fusion: Process each modality separately, then combine outputs.
  - Cross-attention: One modality attends to another (e.g., text attends to image features).

- Shared embedding space: Project all modalities into a common representation for alignment.

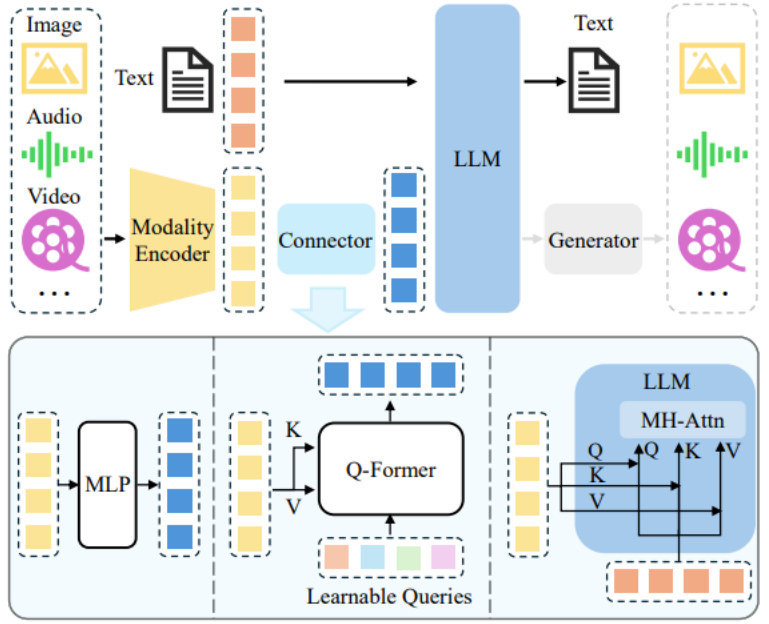


Fig. 2: An illustration of typical MLLM architecture. It includes an encoder, a connector, and a LLM. An optional generator can be attached to the LLM to generate more modalities besides text. The encoder takes in images, audios or videos and outputs features, which are processed by the connector so that the LLM can better understand. There are broadly three types of connectors: projection-based, query-based, and fusion-based connectors. The former two types adopt token-level fusion, processing features into tokens to be sent along with text tokens, while the last type enables a feature-level fusion inside the LLM.

Image from: [“A survey on multimodal large language models”, Yin et al., National Science Review 2024]

Multimodal Transformers

Popular approaches

- CLIP (OpenAI): Contrastive learning to align image and text embeddings.
- ALIGN (Google): Large-scale image-text alignment with noisy web data.
- VisualBERT / LXMERT: Pretrained vision-language transformers for VQA.
- ViLT: Vision-and-language transformer with minimal visual preprocessing.
- Florence / BEiT-3: Unified multi-modal foundation models.
- Audio-Visual Transformers: AV-Hubert, VATT for audio-video-text tasks.

Key trends:

- Pretraining on massive multi-modal datasets.
- Contrastive objectives for cross-modal alignment.
- Unified architectures for multiple modality pairs.

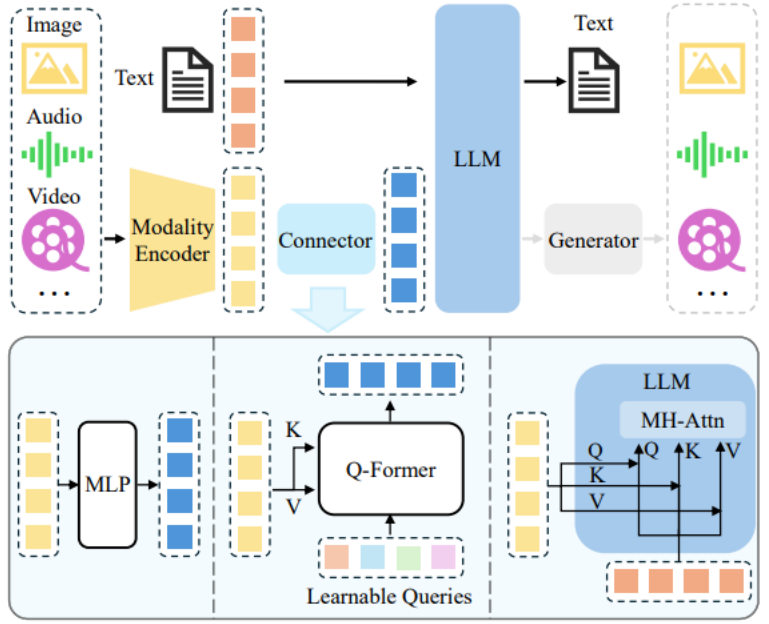


Fig. 2: An illustration of typical MLLM architecture. It includes an encoder, a connector, and a LLM. An optional generator can be attached to the LLM to generate more modalities besides text. The encoder takes in images, audios or videos and outputs features, which are processed by the connector so that the LLM can better understand. There are broadly three types of connectors: projection-based, query-based, and fusion-based connectors. The former two types adopt token-level fusion, processing features into tokens to be sent along with text tokens, while the last type enables a feature-level fusion inside the LLM.

Multimodal Transformers

CLIP from OpenAI is one of the most popular approaches for training multi-modal models.

It is based on the idea of learning a shared embedding space with a “contrastive” approach

The focus in the paper is on text + images, with a dataset of 400 million (image, text) pairs, but it has been extended to more domains

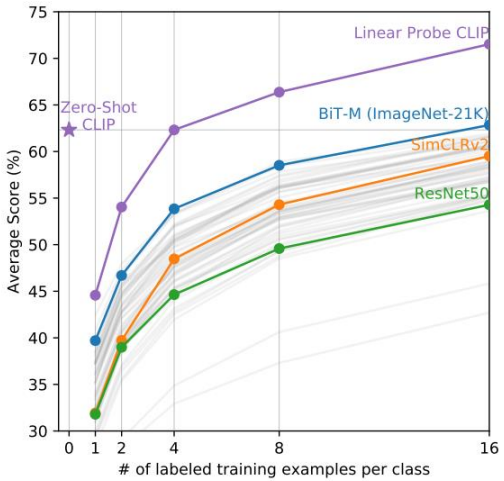


Figure 6. **Zero-shot CLIP outperforms few-shot linear probes.** Zero-shot CLIP matches the average performance of a 4-shot linear classifier trained on the same feature space and nearly matches the best results of a 16-shot linear classifier across publicly available models. For both BiT-M and SimCLRv2, the best performing model is highlighted. Light gray lines are other models in the eval suite. The 20 datasets with at least 16 examples per class were used in this analysis.

Learning Transferable Visual Models From Natural Language Supervision

Alec Radford [*1] Jong Wook Kim [*1] Chris Hallacy [1] Aditya Ramesh [1] Gabriel Goh [1] Sandhini Agarwal [1]
Girish Sastry [1] Amanda Askell [1] Pamela Mishkin [1] Jack Clark [1] Gretchen Krueger [1] Ilya Sutskever [1]

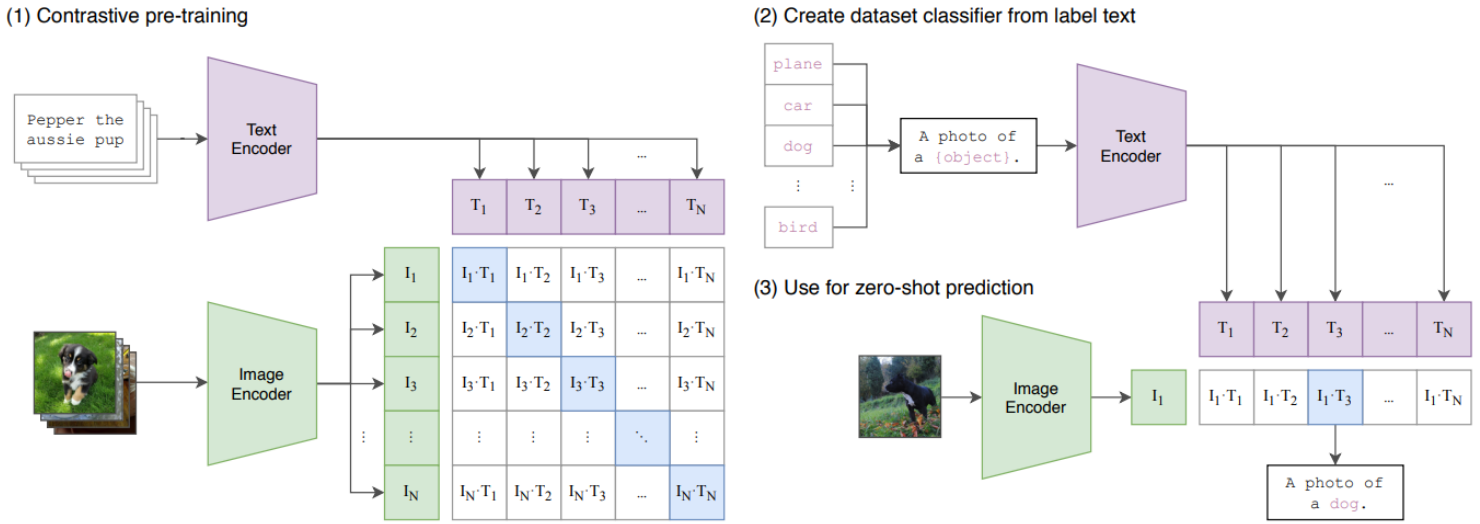


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset’s classes.

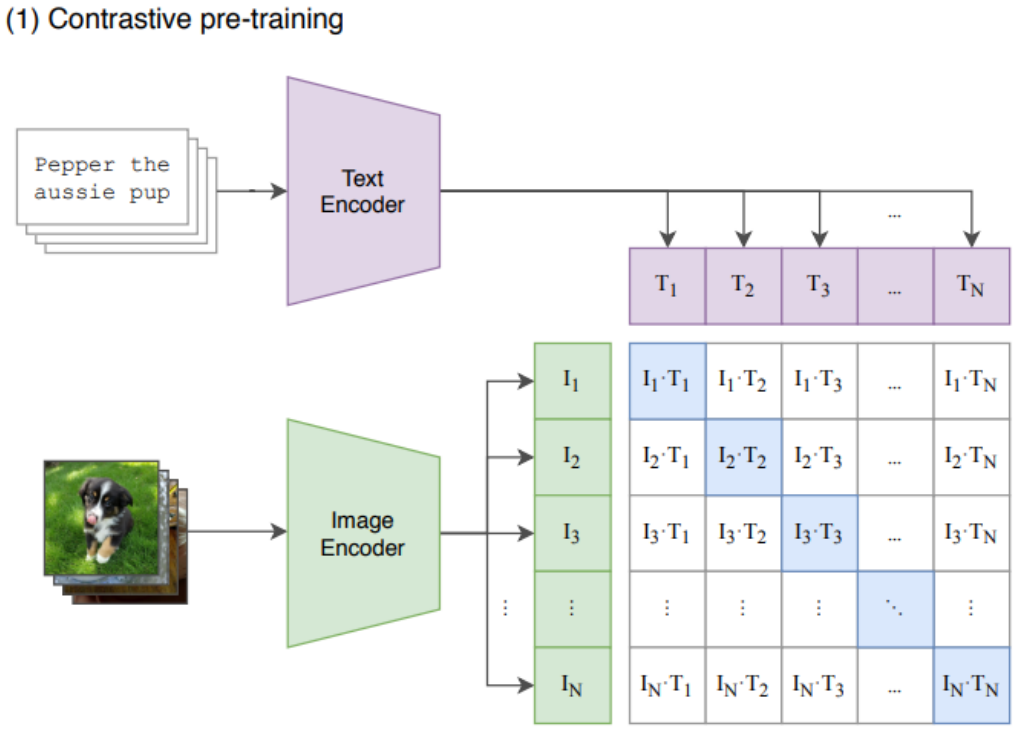
Multimodal Transformers

Given a batch of N (image, text) pairs, CLIP is trained to predict which of the N × N possible (image, text) pairings across a batch actually occurred.

CLIP learns a multi-modal embedding space by jointly training an image encoder and text encoder to maximize the cosine similarity of the image and text embeddings of the N real pairs in the batch while minimizing the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairings.

N batch size, $v_i, t_i \in R^d$ image and text embeddings for i-th element in batch

- Compute similarity between (image, text) pairs of elements in batch $S_{i,j} = \frac{v_i \cdot t_j}{|v_i||t_j|}$
- Image-to-text Loss: $L_{img} = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{\exp(S_{i,i})}{\sum_{j=1}^{N}\exp(S_{i,j})}$
- Text-to-image Loss: $L_{text} = \frac{1}{N}\sum_{j=1}^{N} -\log \frac{\exp(S_{j,j})}{\sum_{i=1}^{N}\exp(S_{i,j})}$
- Final Loss: $L = (L_{img} + L_{text})/2$



Multimodal Transformers

In general, the idea of contrastive losses is the following:

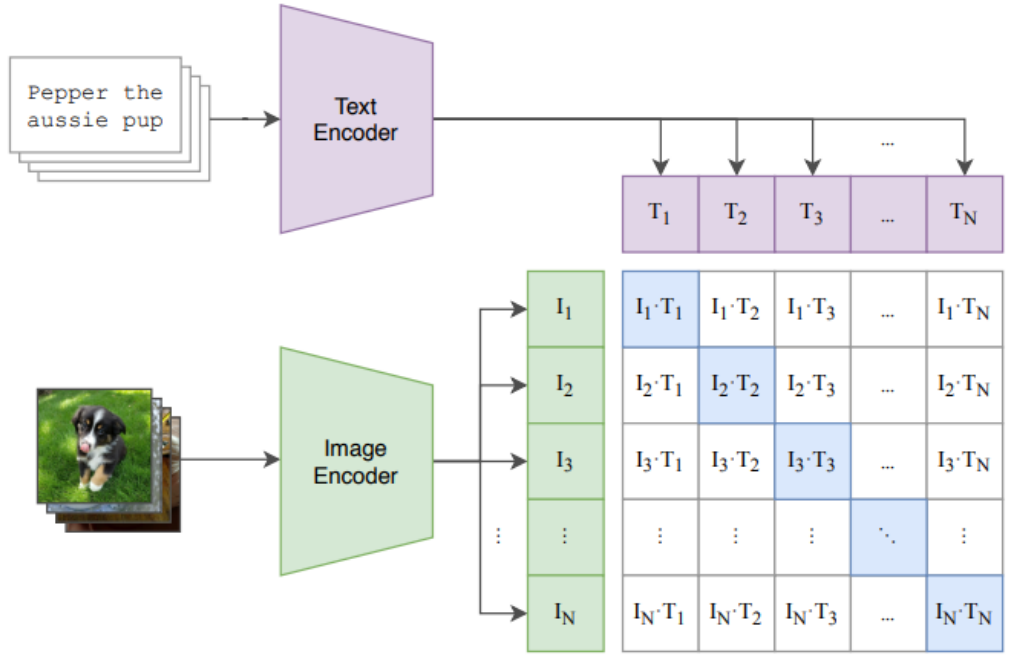
- Given a positive and a negative example
- encourage the model to produce
  - similar representations for positive examples
  - Dissimilar representations for negative examples

Given a batch of N (image, text) pairs, CLIP is trained to predict which of the N × N possible (image, text) pairings across a batch actually occurred.

CLIP learns a multi-modal embedding space by jointly training an image encoder and text encoder to maximize the cosine similarity of the image and text embeddings of the N real pairs in the batch while minimizing the cosine similarity of the embeddings of the $N^2 - N$ incorrect pairings.

N batch size, $v_i, t_i \in R^d$ image and text embeddings for i-th element in batch

- Compute similarity between (image, text) pairs of elements in batch $S_{i,j} = \frac{v_i \cdot t_j}{|v_i||t_j|}$
- Image-to-text Loss: $L_{img} = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{\exp(S_{i,i})}{\sum_{j=1}^{N}\exp(S_{i,j})}$
- Text-to-image Loss: $L_{text} = \frac{1}{N}\sum_{j=1}^{N} -\log \frac{\exp(S_{j,j})}{\sum_{i=1}^{N}\exp(S_{i,j})}$
- Final Loss: $L = (L_{img} + L_{text})/2$



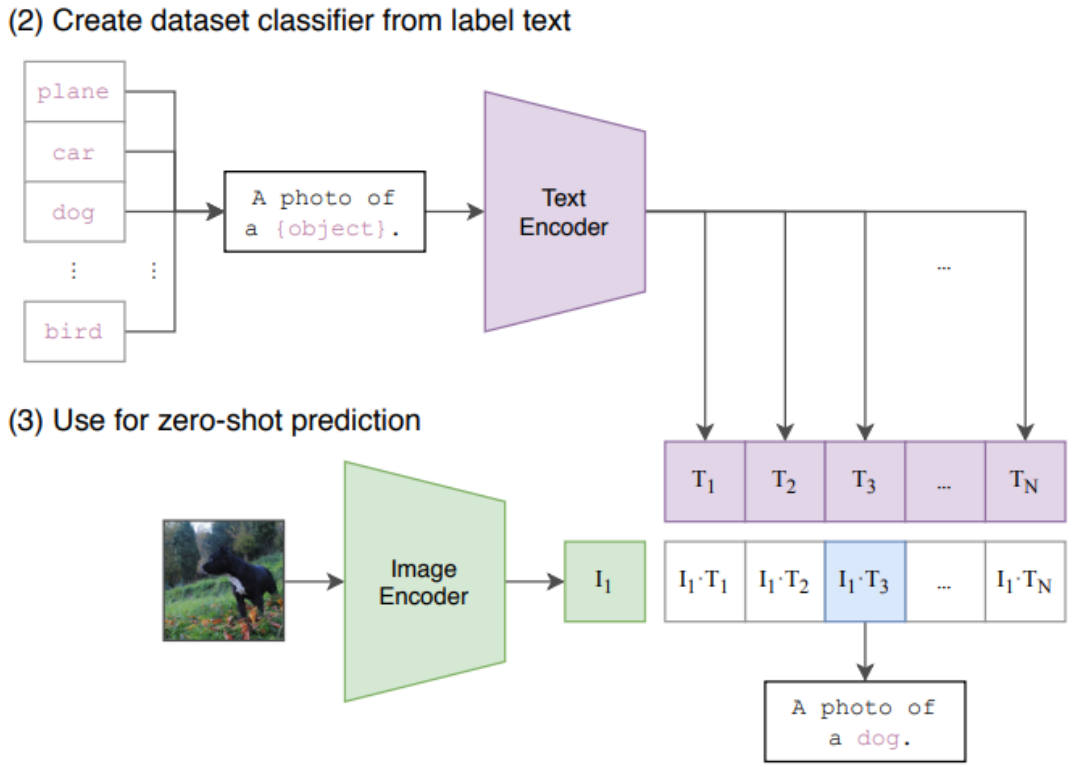
Multimodal Transformers

The model can then be used as a classifier by using the names of all the classes in the dataset as the set of potential text pairings and predict the most probable

In other words, CLIP enables the use of natural language as a supervision signal for image classification!

Strengths

- Zero-shot learning: No fine-tuning needed for new tasks — just provide text prompts.
- Flexible input: Works with any natural language description.
- Scalable: Benefits from massive, noisy datasets.
- Generalization: Learns broad concepts across domains.
- Foundation for generative models: Used in guidance for text-to-image generation (e.g., Stable Diffusion).



Multimodal Transformers

Clip shows strong performance and better scaling behaviour

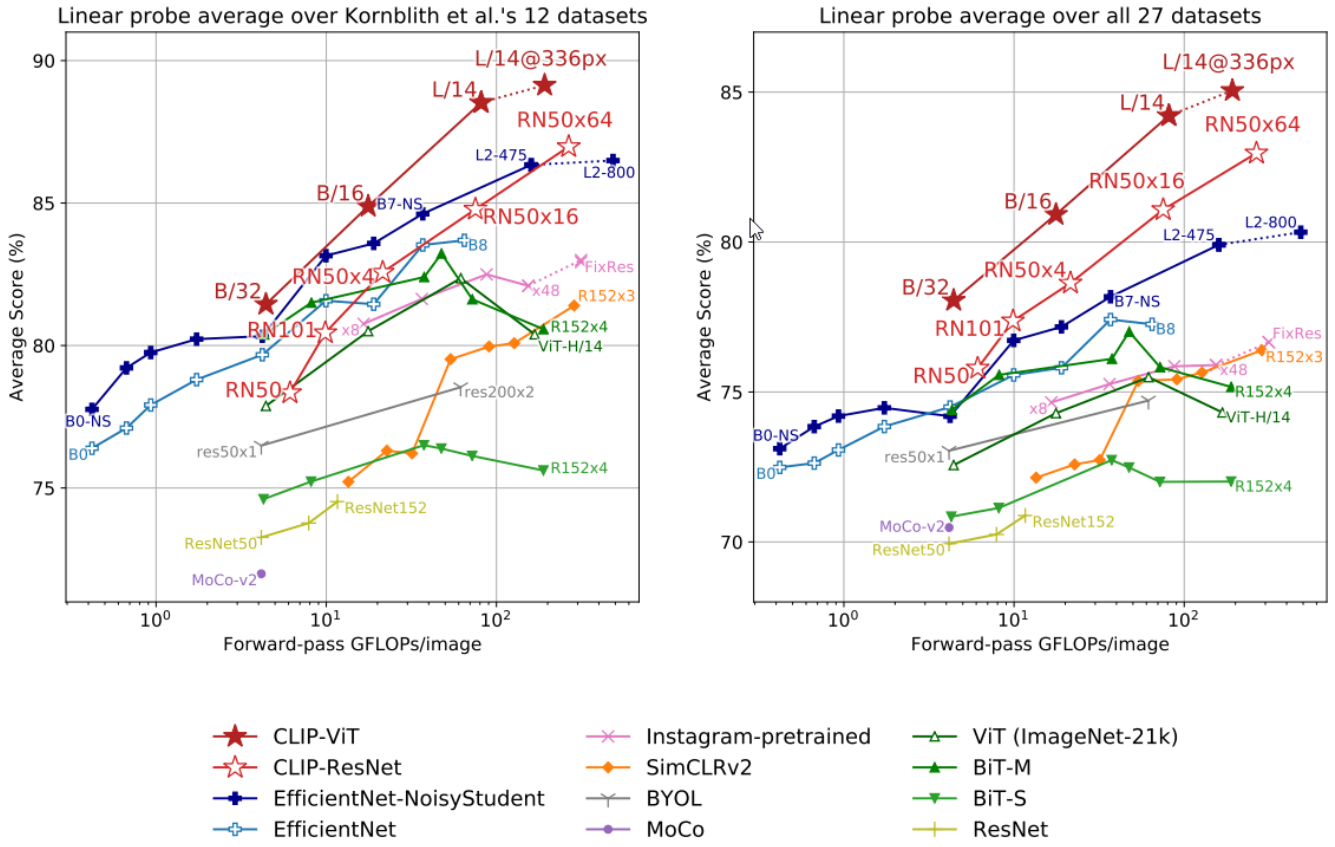


Image from: [“A survey on multimodal large language models”, Yin et al., National Science Review 2024]

Multimodal Transformers

Open Research directions

- Unified foundation models
  - Single model for text, image, audio, video.
- Better efficiency
  - Lightweight fusion, parameter sharing.
- Few-shot & zero-shot multi-modal learning
  - Leveraging pretrained embeddings.
- Handling missing modalities
  - Robust inference with partial inputs.
- Hallucinations

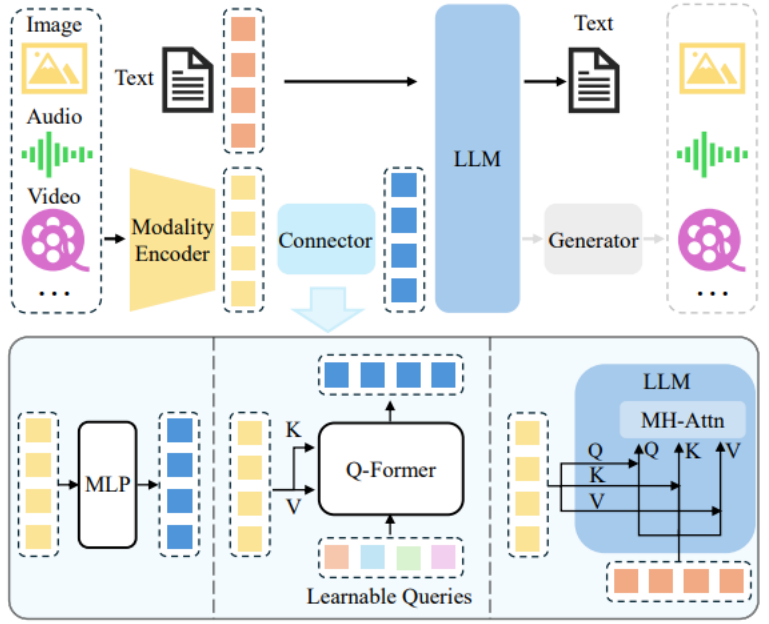


Fig. 2: An illustration of typical MLLM architecture. It includes an encoder, a connector, and a LLM. An optional generator can be attached to the LLM to generate more modalities besides text. The encoder takes in images, audios or videos and outputs features, which are processed by the connector so that the LLM can better understand. There are broadly three types of connectors: projection-based, query-based, and fusion-based connectors. The former two types adopt token-level fusion, processing features into tokens to be sent along with text tokens, while the last type enables a feature-level fusion inside the LLM.

Training a Foundation Model

Two main paradigms

- Random masked token prediction
  - All tokens can attend to all others
  - Use case: embedding model

- Next token prediction
  - Enforces causality in predictions
  - Use case: generative model

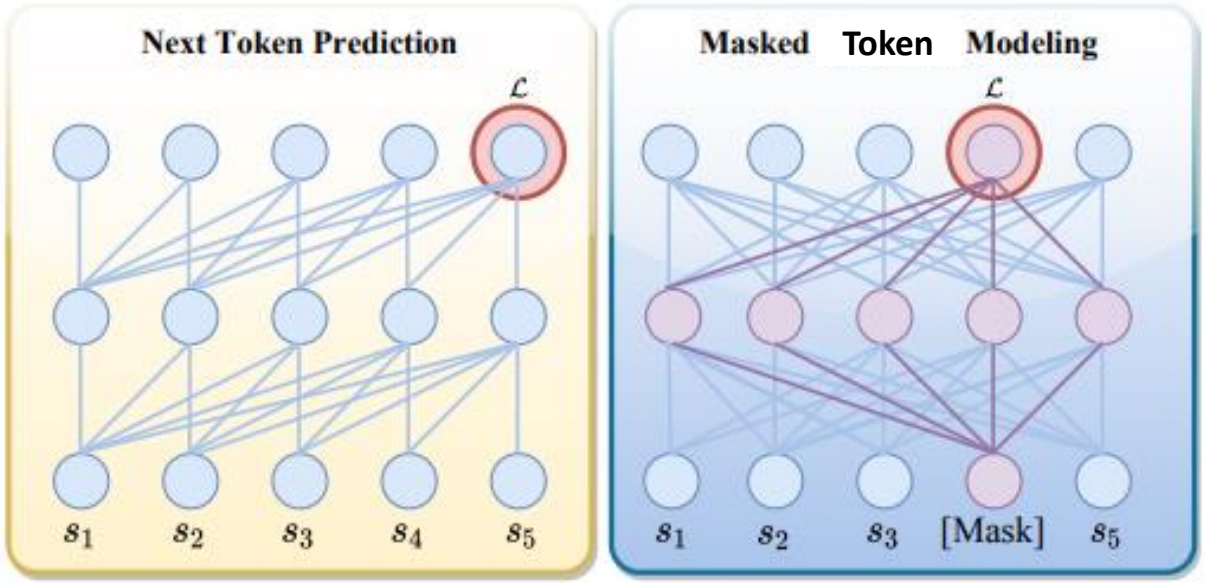


Image from “Mask-Enhanced Autoregressive Prediction: Pay Less Attention to Learn More”, Zhuang et al., ICML 2025

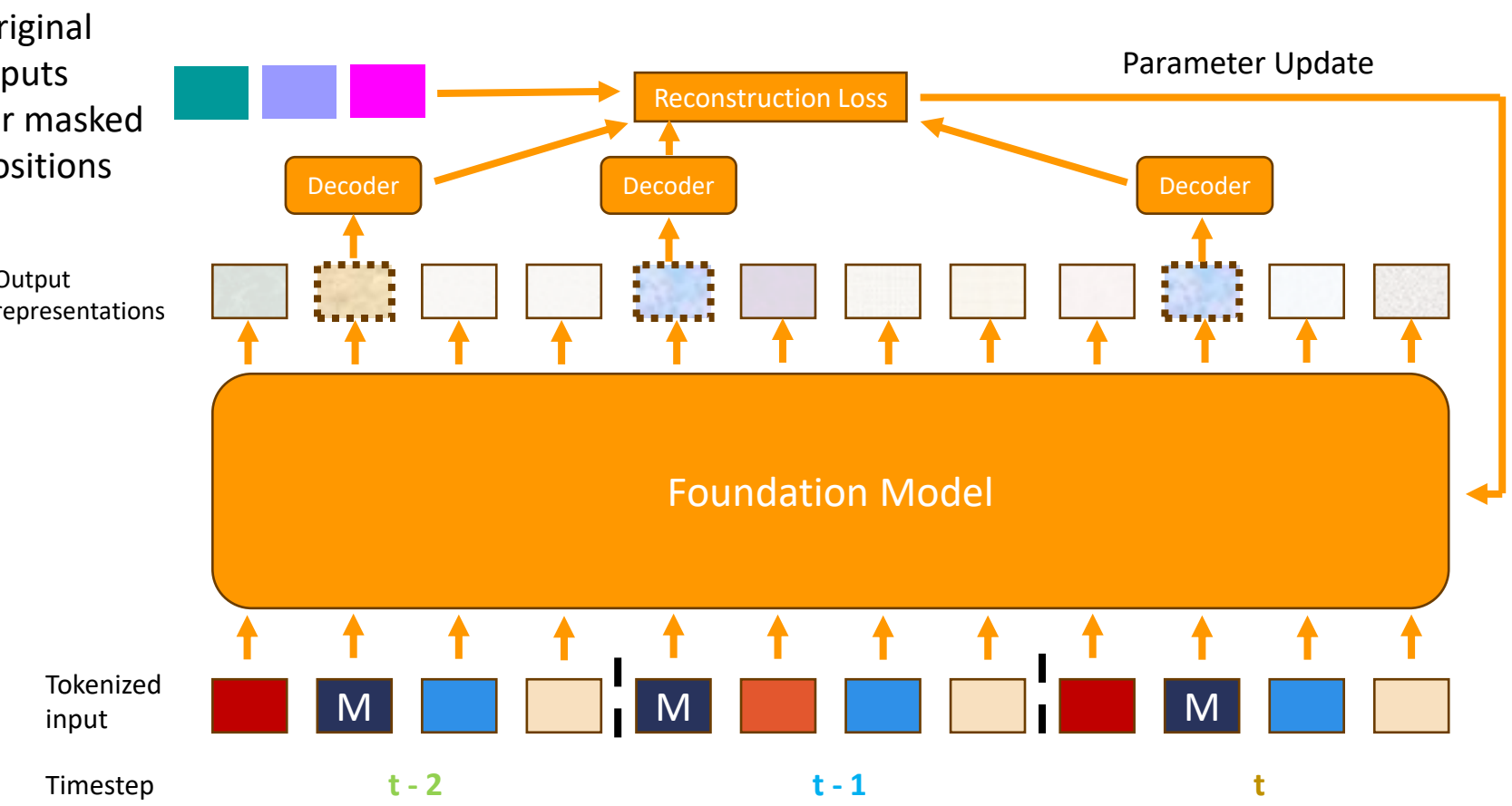
Training a Foundation Model

Random masked token prediction

- A random amount (typically between 15 and 50%) of the input tokens are replaced with a “mask” token
- The transformer model produces representations for all inputs. The ones related to masked inputs are decoded and a reconstruction loss is applied e.g., MSE-Loss

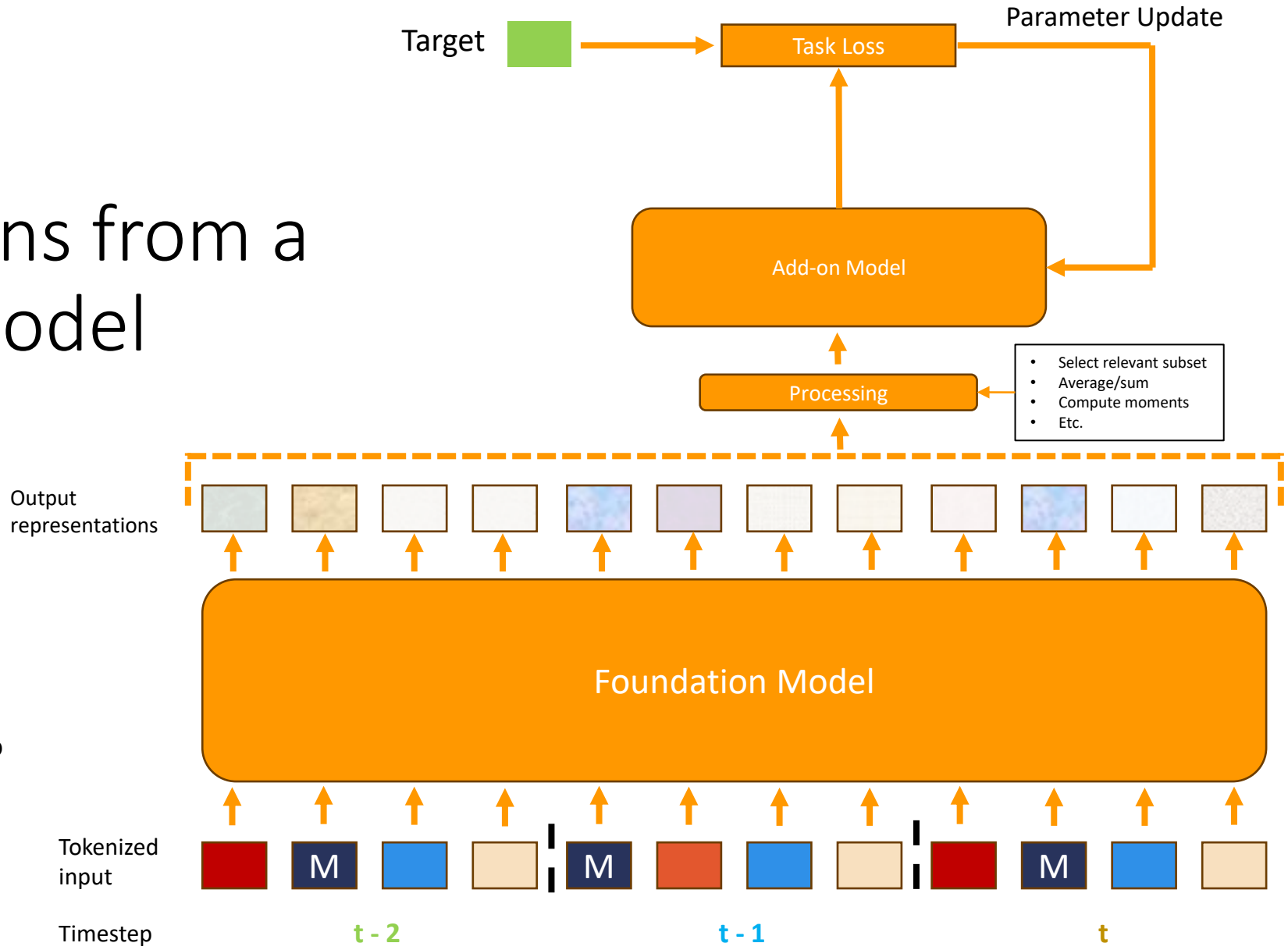
$$L(y, \tilde{y}) = \left|\left|y - \tilde{y}\right|\right|^2$$

- This kind of training encourages the model to understand the structure of the data and to produce representations that encode all the necessary information to reconstruct the input
- Typically used to train “embedding models”, i.e., models that produce representations that can then be used to perform downstream tasks



Using the Representations from a Foundation Model

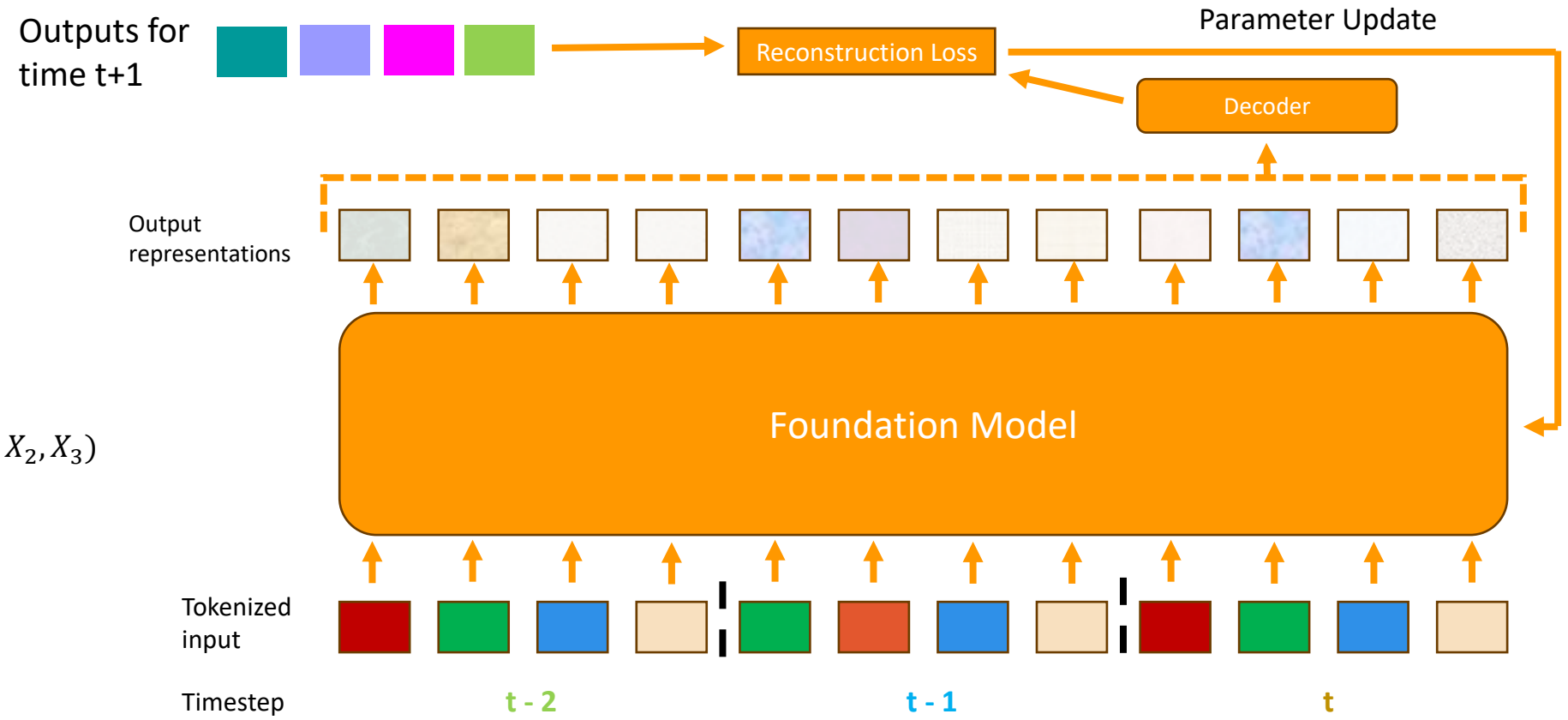
- A typical procedure involves training an Add-On model on s mall amounts of supervised data
- Add-on models are much smaller and simpler than Foundation Models
  - Often a simple MLP, CNN, or Attention-based network
- Some operations/reductions may be applied to the representations before passing them to the Add-on Model



Training a Foundation Model

Next token prediction

- The model is trained to reconstruct the token(s) that follows the input
- Causal masking is applied in the attention mechanism so that there is no leaking of future information
- Typically used to train autoregressive generative models:
$P(X_1, X_2, X_3, X_4) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2)P(X_4|X_1, X_2, X_3)$



Part 5

Case Studies and Experimental Evaluation

- Challenges in communication systems
- Overview of existing approaches
- Overview of empirical results

Challenges with Communication Data

Data in communication systems is extremely heterogeneous

Main Challenges

- Tokenization
  - How to tokenize data with different types/dimensionalities?
- Heterogeneous data
  - How to process and incorporate information from many different sources?
- Handling dimensionalities varying in time
  - How to allow input dimensionalities to change over time?
- Unequal sampling frequencies
  - How to handle features changing at different rates?

Depends on number of receivers

Depends on number of subcarrier groups

Depends on number of transmitters

Example features from communication systems

Feature	Description
Channel type (categorical)	The type of SIONNA channel (UMi, UMa, RMa)
$K \in \mathbb{Z}^+$	Number of subcarriers per OFDM symbol
$\hat{\mathbf{C}}_{\text{n}} \in \mathbb{C}^{N_R \times N_R}$	Estimated noise covariance matrix
$\hat{\mathbf{R}}_{\text{f, robust}} \in \mathbb{C}^{B \times B}$	Estimated frequency correlation matrix
$\hat{\mathbf{C}}_{\text{time}} \in \mathbb{C}^{\lvert\mathcal{S}\rvert \times \lvert\mathcal{S}\rvert}$	Estimated time covariance matrix
$\hat{\mathbf{R}}_{\text{time}} \in \mathbb{C}^{\lvert\mathcal{S}\rvert \times \lvert\mathcal{S}\rvert}$	Estimated time correlation matrix
$\hat{\mu}, \hat{\ell} \in \mathbb{R}$	Estimated center and length of delay profile
$\hat{w} \in \mathbb{R}^+$	Estimated width of Doppler spectrum
$\hat{\mathbf{W}}_{(R)} \in \mathbb{C}^{N_T \times R}$	Selected precoder of rank R
$\hat{R} \in \mathbb{Z}^+$	Selected transmission rank
$\hat{G} \in \mathbb{R}^+$	Estimated Spectral Efficiency

Depends on number of symbols with pilots

Existing Approaches

- “Towards a Wireless Physical-Layer Foundation Model: Challenges and Strategies”, Jaron Fontaine, Adnan Shahid, Eli De Poorter, IEEE International Conference on Communications Workshops (2024)
- “Distributed Foundation Models for Multi-Modal Learning in 6G Wireless Networks”, Jun Du, Tianyi Lin, Chunxiao Jiang, Qianqian Yang, C. Faouzi Bader, Zhu Han, IEEE Wireless Communications (2024)
- “Large Multi-Modal Models (LMMs) as Universal Foundation Models for AI-Native Wireless Systems”, Shengzhe Xu, Christo Kurisummoottil Thomas, Omar Hashash, Nikhil Muralidhar, Walid Saad, Naren Ramakrishnan, IEEE Network (2024)
- “Towards a Foundation Model for Communication Systems”, Davide Buffelli, Sowmen Das, Yu-Wei Lin, Sattar Vakili, Chien-Yi Wang, Masoud Attarifar, Pritthijit Nath, Da-shan Shiu, ML4Wireless Workshop at the 42nd International Conference on Machine Learning (2025)
- “Large Wireless Model (LWM): A Foundation Model for Wireless Channels”, Sadjad Alikhani,Gouranga Charan and Ahmed Alkhateeb, arXiv (2025)
- “A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning”, Berkay Guler, Giovanni Geraci, Hamid Jafarkhani, NeurIPS 2025, AI4NextG Workshop (2025)
- “WiFo: wireless foundation model for channel prediction”, Boxun Liu, Shijian Gao, Xuanyu Liu, Xiang Cheng & Liuqing Yang, Sci. China Inf. Sci. (2025)
- “Addressing the Curse of Scenario and Task Generalization in AI-6G: A Multi-Modal Paradigm”, Tianyu Jiao, Zhuoran Xiao, Yin Xu, Chenhui Ye, Yihang Huang, Zhiyong Chen, Liyu Cai, Jiang Chang, Dazhi He, Yunfeng Guan, Guangyi Liu, Wenjun Zhang, IEEE Transactions on Wireless Communications (2025)
- “ChannelGPT: A Large Model toward Real-World Channel Foundation Model for 6G Environment Intelligence Communication”, Li Yu, Lianzheng Shi, Jianhua Zhang, Zhen Zhang, Yuxiang Zhang, Guangyi Liu, IEEE Communications Magazine (2025)
- “LLM-Enabled Multi-Modal Data Synthesis via Cross-Domain Collaboration”, Xiaomao Zhou, Yujiao Hu, Qingmin Jia, Renchao Xie, IEEE Communications Magazine (2025)

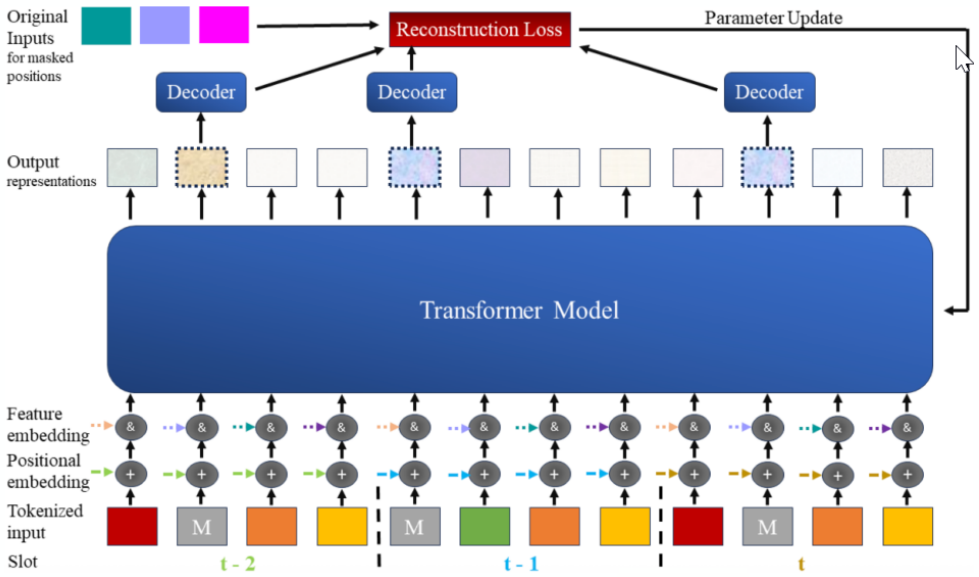
"Towards a Foundation Model for Communication Systems", Buffelli et al.

- Input: Multiple variables of interest
  - Each variable type has a specialized tokenization and normalization strategy
    - Scalars: Fourier Encodings $\mathrm{x} \to Tok(x)_i = [\cos\frac{2\pi x}{\lambda_i}, \sin\frac{2\pi x}{\lambda_i}]$
    - Vectors: Linear transformation $x \to Wx$
    - Matrices: Patching $X \in \mathbb{R}^{d_1 \times d_2} \to \{X_j \in \mathbb{R}^{p_1 \times p_2}, j = 1, \dots\}$
    - Categorical: learnable vector for each category

- Architecture has addition of "feature embeddings"
  - Enables the model to identify features within a timestep

- Training objective: Masked Channel Modeling (MCM)
  Randomly mask a fraction (15%) of the patches but only for "target" variables (transmission rank, selected precoder, Doppler spectrum, center and length of the delay profile. These are features that can be estimated from the remaining ones)
  - Loss is mean squared error between original (unmasked) patch and prediction from Foundation Model

Feature	Description
Channel type (categorical)	The type of SIONNA channel (UMi, UMa, RMa)
$K \in \mathbb{Z}^+$	Number of subcarriers per OFDM symbol
$\hat{\mathbf{C}}_{\mathrm{n}} \in \mathbb{C}^{N_R \times N_R}$	Estimated noise covariance matrix
$\hat{\mathbf{R}}_{\mathrm{f,\,robust}} \in \mathbb{C}^{B \times B}$	Estimated frequency correlation matrix
$\hat{\mathbf{C}}_{\mathrm{time}} \in \mathbb{C}^{\lvert\mathcal{S}\rvert \times \lvert\mathcal{S}\rvert}$	Estimated time covariance matrix
$\hat{\mathbf{R}}_{\mathrm{time}} \in \mathbb{C}^{\lvert\mathcal{S}\rvert \times \lvert\mathcal{S}\rvert}$	Estimated time correlation matrix
$\hat{\mu}, \hat{\ell} \in \mathbb{R}$	Estimated center and length of delay profile
$\hat{w} \in \mathbb{R}^+$	Estimated width of Doppler spectrum
$\hat{\mathbf{W}}_{(R)} \in \mathbb{C}^{N_T \times R}$	Selected precoder of rank R
$\hat{R} \in \mathbb{Z}^+$	Selected transmission rank
$\hat{G} \in \mathbb{R}^+$	Estimated Spectral Efficiency



"Towards a Foundation Model for Communication Systems", Buffelli et al.

- Dataset
  Using open-source SIONNA package [Hoydis et al., 2022] we simulate a 5G OFDM communication. Each simulation runs for 100 slots, and is divided in 5-slot datapoints

- Select multitude of realistic configurations
  - 2900 possible combinations of parameters
  - Around 3M slots of data

Feature	Description
Channel type (categorical)	The type of SIONNA channel (UMi, UMa, RMa)
$K \in \mathbb{Z}^+$	Number of subcarriers per OFDM symbol
$\hat{\mathbf{C}}_{\text{n}} \in \mathbb{C}^{N_R \times N_R}$	Estimated noise covariance matrix
$\hat{\mathbf{R}}_{\text{f, robust}} \in \mathbb{C}^{B \times B}$	Estimated frequency correlation matrix
$\hat{\mathbf{C}}_{\text{time}} \in \mathbb{C}^{\lvert\mathcal{S}\rvert \times \lvert\mathcal{S}\rvert}$	Estimated time covariance matrix
$\hat{\mathbf{R}}_{\text{time}} \in \mathbb{C}^{\lvert\mathcal{S}\rvert \times \lvert\mathcal{S}\rvert}$	Estimated time correlation matrix
$\hat{\mu}, \hat{\ell} \in \mathbb{R}$	Estimated center and length of delay profile
$\hat{w} \in \mathbb{R}^+$	Estimated width of Doppler spectrum
$\hat{\mathbf{W}}_{(R)} \in \mathbb{C}^{N_T \times R}$	Selected precoder of rank R
$\hat{R} \in \mathbb{Z}^+$	Selected transmission rank
$\hat{G} \in \mathbb{R}^+$	Estimated Spectral Efficiency

Parameter	Values
Channel type	UMi, UMa, RMa
(Center frequency, subcarrier spacing) (Hz)	(2.6G, 15k), (3.5G, 30k)
SNR (dB)	[0, 30]
Receiver speed (km/h)	3, 10, 30, 60, 90
(N_T, N_R)	(4, 1), (4, 2), (4, 4), (8, 1), (8, 2), (8, 4)
Number of subcarrier group (B)	25, 50, 75, 100
Size of subcarrier group (M)	12, 24, 48
Set of pilot symbols ($\mathcal{S}$)	{2, 8}, {2, 6, 10}, {4, 8, 12}, {2, 5, 8, 11}

"Towards a Foundation Model for Communication Systems", Buffelli et al.

Evaluation

- 1 Million Datapoints
- 3 Model sizes

- Error on Estimation and Forecasting of different features

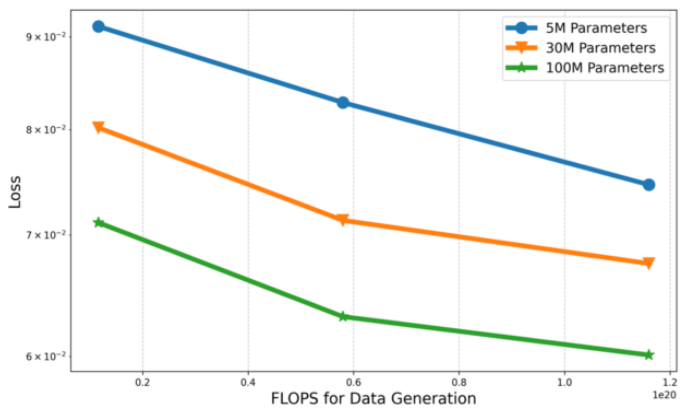
Scaling Laws

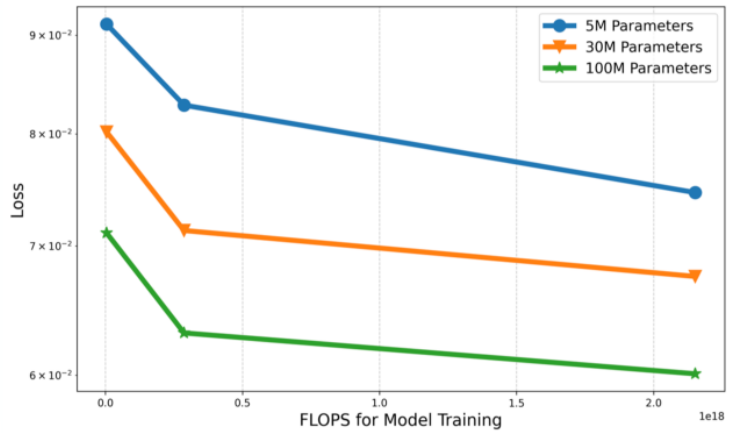
- Performance improves with both model and dataset size
- Larger models benefit more from extended training
- Consistent with known scaling behaviour observed in other domains

Model	#layers	#head	token dim	hidden dim
5M	6	6	387	1548
30M	32	18	387	1548
100M	75	24	429	1716

Table 5. Estimation Error (MSE) for the **forecasting** and **interpolation** tasks on five key features: center of the delay profile ($\hat{\mu}$), length of the delay profile ($\hat{\ell}$), Doppler spectrum width ($\hat{w}$), transmission rank ($\hat{R}$), and selected precoder ($\hat{\mathbf{W}}$).

	$\hat{\mu}$	$\hat{\ell}$	$\hat{w}$	$\hat{R}$	$\hat{\mathbf{W}}$
Forecasting	0.019	0.021	0.077	0.129	0.101
Interpolation	0.019	0.020	0.054	0.128	0.100

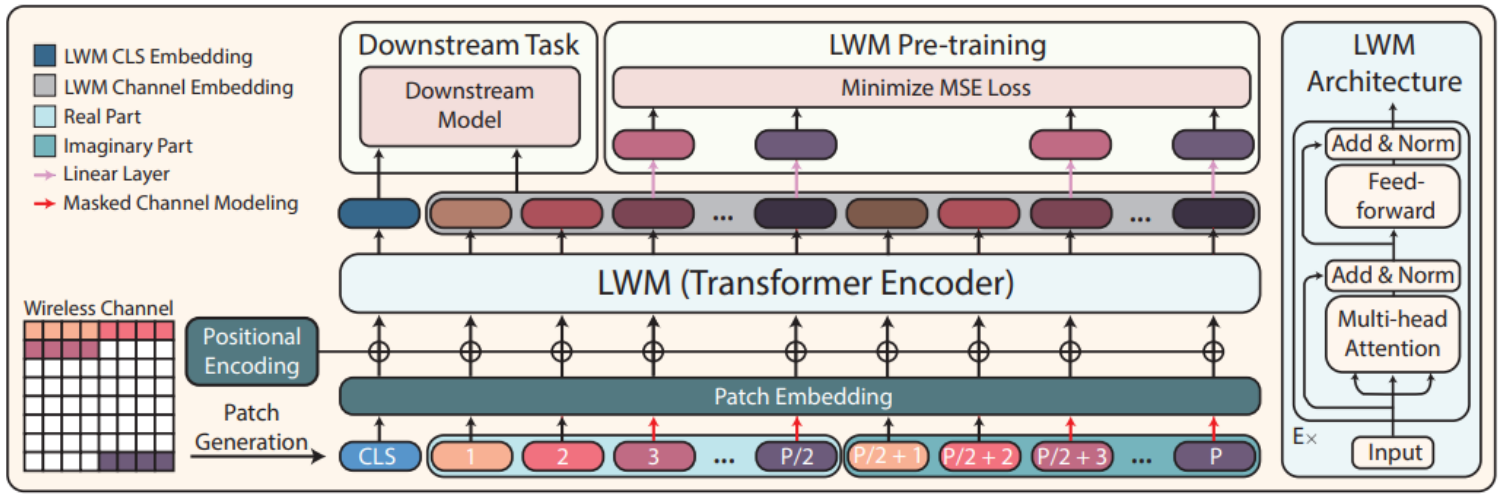




MEDIATEK

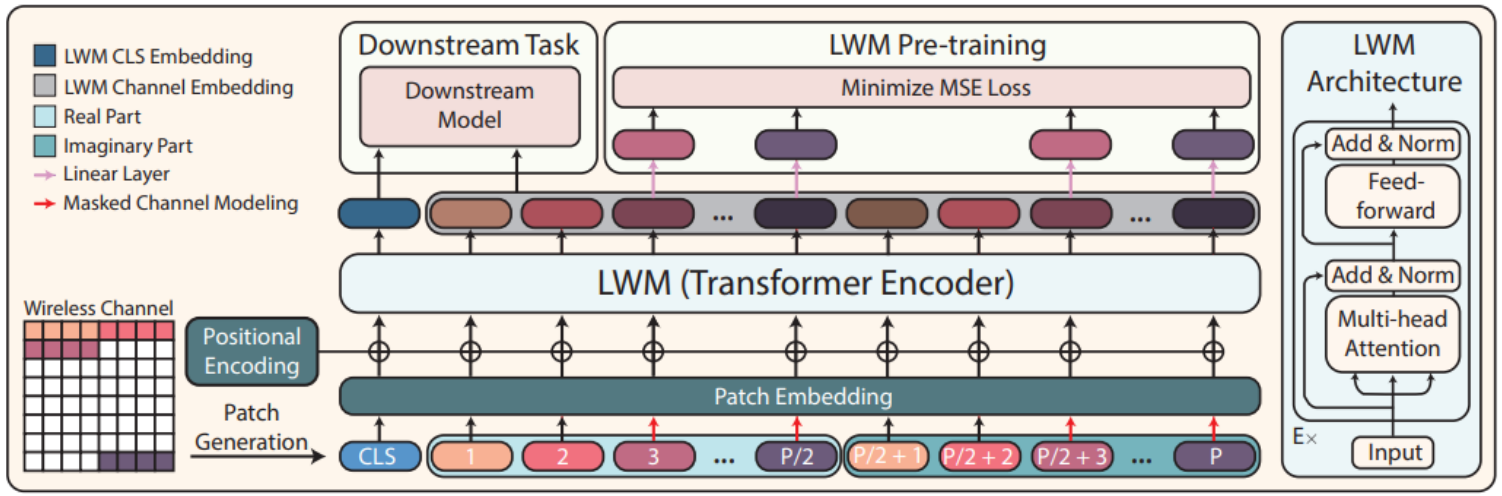
"Large Wireless Model (LWM): A Foundation Model for Wireless Channels", Alikhani et al.

- Input: Patches of Channel Matrix
  Channel matrix $H \in \mathbb{C}^{M \times N}$ is split into P patches as follows:
  1 – split real and imaginary part: $H_{real} = R(H), H_{imag} = I(H)$
  2 – flatten $h_{real} = vec(H_{real}), h_{imag} = vec(H_{imag})$
  3 – divide in patches of length $L = 2MN/P : p_i = h_{real}[(i-1)L + 1 : iL], p_{i+\frac{P}{2}} = h_{imag}[(i-1)L + 1 : iL], i \in \{1, 2, .., \frac{P}{2}\}$
  - Tradeoff: smaller patches capture more fine-grained details but introduce more computational cost; larger patches emphasize long range dependencies at the expense of finer details
  - Each patch is transformed into an embedding with a linear transformation
  - CLS is always position 0, others have positional embedding (uniform vector filled with index of position and then transformed by a learnable matrix)



“Large Wireless Model (LWM): A Foundation Model for Wireless Channels”, Alikhani et al.

- Training objective: Masked Channel Modeling (MCM)
  Randomly mask a fraction (15%) of the patches (masking both the real and corresponding imaginary part to avoid leakage); for the values within each masked patch:
  - 80% are fully masked (a predetermined vector is used to replace them)
  - 10% are replaced with random values sampled from a Gaussian (adds noise to increase robustness)
  - 10% are left unchanged
  - Loss is mean squared error between original (unmasked) patch and prediction from Foundation Model
  - An additional special patch [CLS] (with an associated learnable vector) is appended to each input to obtain a representation for the whole sequence. Furthermore the attention scores with CLS provide importance of each patch(can be used to identify critical segments)



"Large Wireless Model (LWM): A Foundation Model for Wireless Channels", Alikhani et al.

Dataset

- Over 1 million wireless channels from 15 scenarios within the DeepMIMO dataset [Alkhateeb Proc. of Information Theory and Applications Workshop (ITA) 2019]
  - Scenarios: O1, Boston5G, ASU Campus, New York, Los Angeles, Chicago, Houston, Phoenix, Philadelphia, Miami, Dallas, San Francisco, Austin, Columbus, and Seattle
  - Wireless channels are 32x32 matrices split into patches of 16 consecutive subcarriers

Evaluation

- LoS/NLoS (line-of-sight) classification
  - Downstream model: MLP
  - Denver dataset from DeepMIMO (80/10/10 train/val/test)
- Sub-6 to mmWave Beam Prediction
  - predict the strongest mmWave beam at the receiver from a predefined codebook at the base station
  - Downstream model: ResNet CNN on top of channel embeddings
  - The test set comprises six new scenarios from the DeepMIMO dataset, which were not part of LWM's pre-training: Denver, Fort Worth, Oklahoma, Indianapolis, Santa Clara, and San Diego, comprising a total of 14840 samples (70/20/10 train/val/test)

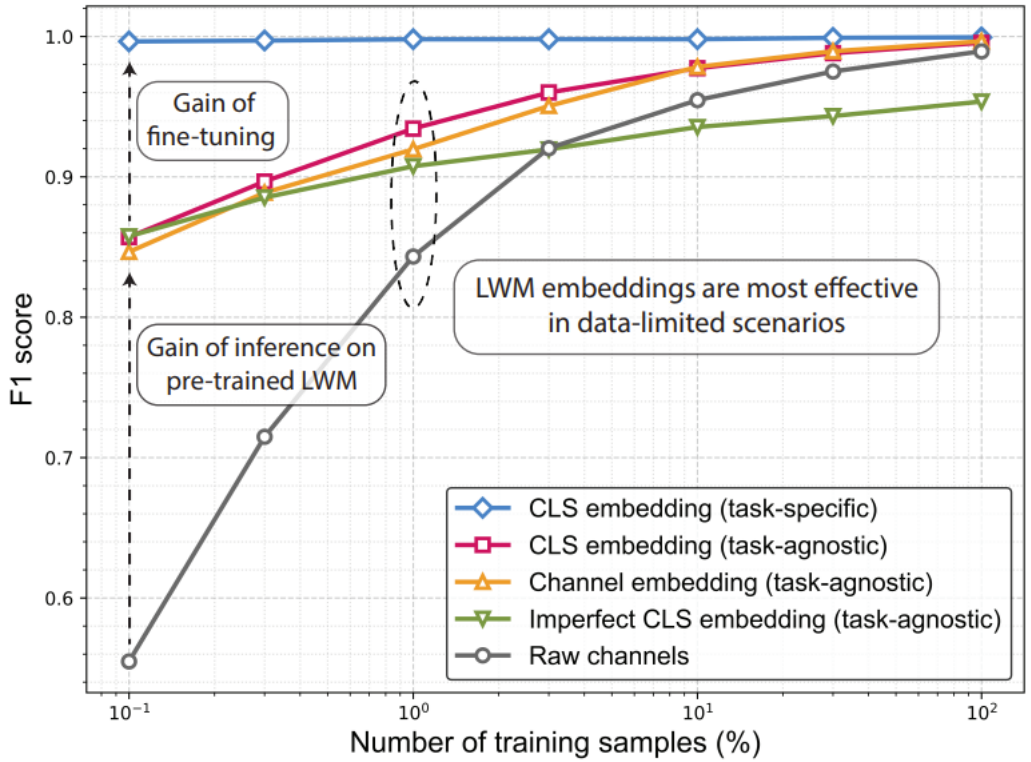
Table I: LWM Pre-training Setup Parameters

Parameter	Value
Antennas at BS (N)	32
Antennas at UEs	1
Subcarriers (M)	32
Patch Size (L)	16
Embedding Size (D)	64
Channel Patches (P)	128
Attention Heads (H)	12
Encoder Layers (E)	12
FFN Hidden Size (D_{FF})	256
Head Dimension (D_{H})	5
Masking Percentage (p)	15 (80/10/10)
Learning Rate	1×10^{-4}
Batch Size	64
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Adam ϵ	1×10^{-8}
Weight Decay	1×10^{-5}
Dropout Rate	0.1
Model Parameters	600K
Training Set Size	820K
Validation Set Size	200K

“Large Wireless Model (LWM): A Foundation Model for Wireless Channels”, Alikhani et al.

LoS/NLoS classification

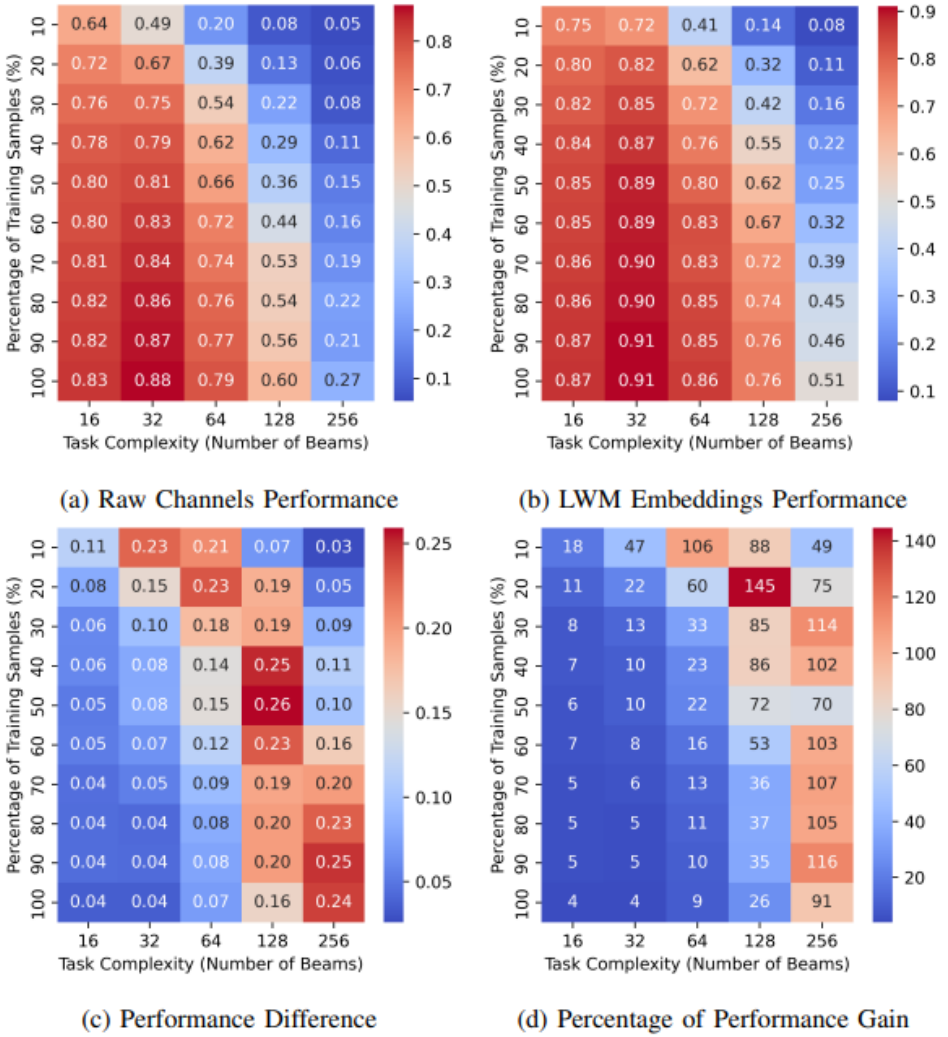
- General-purpose embeddings are extracted from the pre-trained LWM without weight updates, while finetuned embeddings are generated by updating only the last three layers of LWM jointly with the downstream task
- The downstream model is trained using five input types: (i) CLS embeddings (first-patch representation from frozen pre-trained LWM), (ii) channel embeddings (remaining patches of general-purpose embeddings), (iii) raw channels (unprocessed channels), (iv) CLS embeddings inferred from imperfect raw channels corrupted with complex Gaussian noise (SNR = 5 dB), and (v) fine-tuned CLS embeddings
- with only 6 training samples, models trained on raw channel data perform slightly better than random guessing (average F1-score = 0.55), whereas general-purpose embeddings improve performance by +0.31 F1
- fine-tuned embeddings enable perfect class differentiation (F1 ≈ 1.0) even with minimal data
- CLS embeddings outperform channel embeddings in low-data regimes, as channel embeddings capture complex patterns requiring larger datasets for effective utilization; however, channel embeddings slightly surpass CLS performance as training samples grow, leveraging their richer information density



“Large Wireless Model (LWM): A Foundation Model for Wireless Channels”, Alikhani et al.

Sub-6 to mmWave Beam Prediction

- comparison of the performance between raw channels and LWM channel embeddings in training a model for beam prediction
- downstream models trained with raw channels require significantly more data to reach high performance levels, while LWM channel embeddings usually achieve performance saturation with just 50%- 70% of the available data and consistently outperform raw channels
- the F1-score gain percentage heatmap in Fig. d emphasizes the efficiency of embeddings, showing how they scale with fewer resources.
- Fig. c demonstrates that embeddings are most effective when data is limited relative to task complexity



MEDIATEK

“Large Wireless Model (LWM): A Foundation Model for Wireless Channels”, Alikhani, et al.

2D projection of channel embeddings – LWM produces representations that are easier to separate

Divided into LoS/NLoS

Divided into strongest index among 8 beams

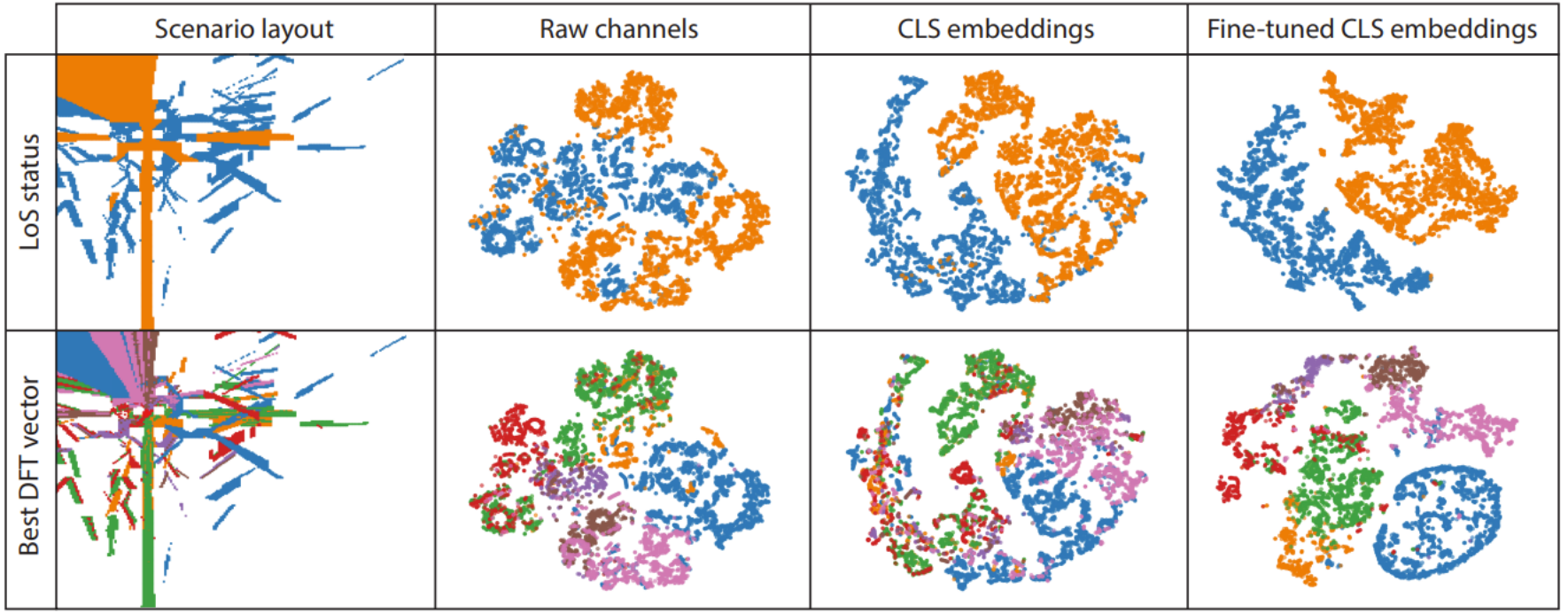
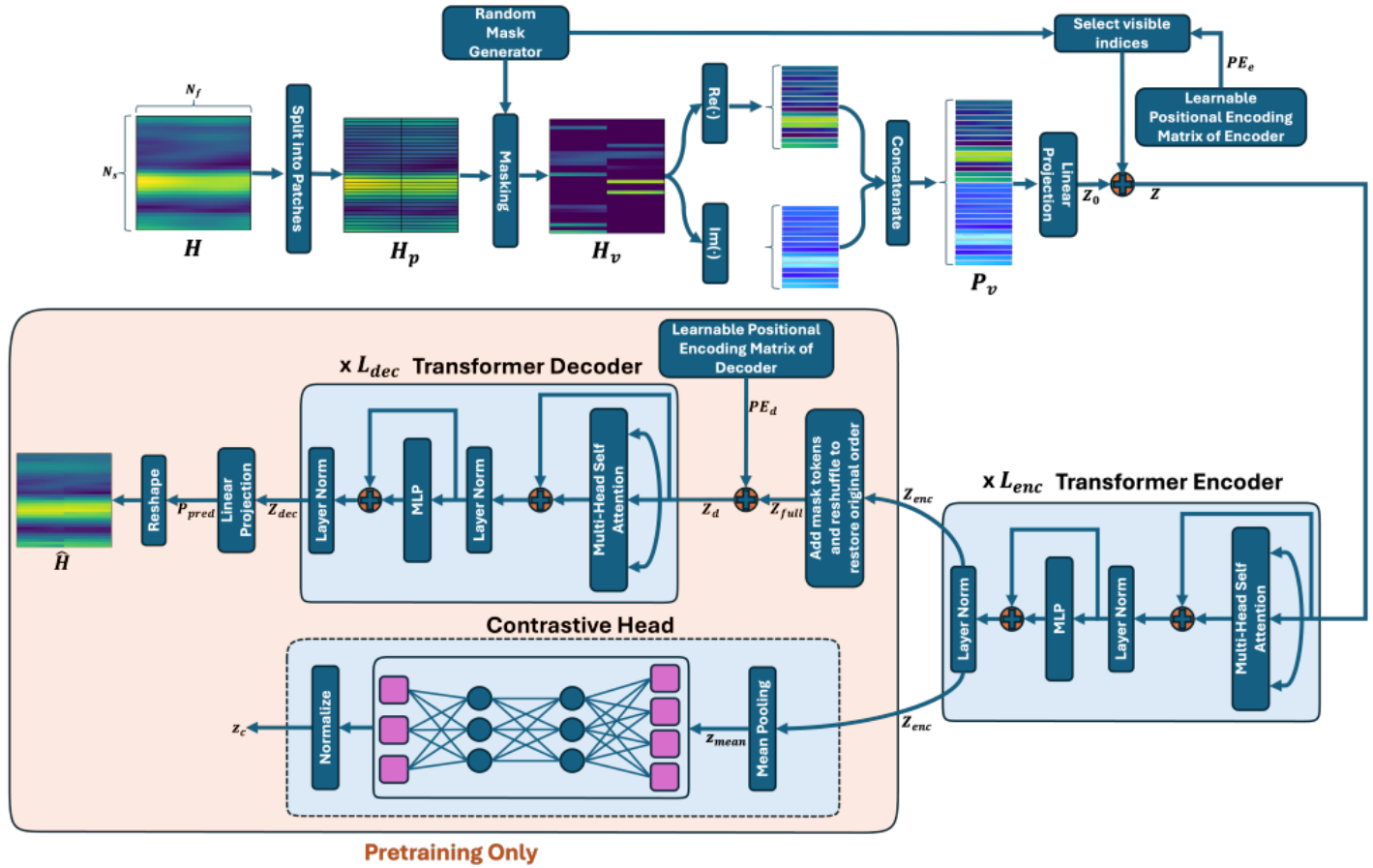


Figure 5: This figure visualizes the distribution of users in the DeepMIMO Denver scenario based on their LoS/NLoS status (top row) and strongest DFT beam index among 8 beams (bottom row). User channels are projected into 2D using t-SNE, comparing raw channels, task-agnostic (general-purpose) LWM embeddings, and fine-tuned LWM embeddings for each task. As seen in Fig. 4, CLS embeddings clearly separate LoS and NLoS channels, enabling high zero-shot classification and strong initialization for downstream training with minimal data. Fine-tuning further enhances downstream task performance.

"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

- Input
  Frequency domain channel response $H \in \mathbb{C}^{N_S \times N_f}$ (with N_f, N_S, number of subcarriers and source antennas)

- Encoder-Decoder Architecture
  - An initial masking is done and only part of the patches (the ones that are not masked) get sent to the encoder.
  - The decoder receives as input the output of the encoder plus mask tokens (in place of the masked patches)
  - A contrastive head only receives the (non-masked) output of the encoder that is used to make the model more robust to noise, fading, and partial observability



"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

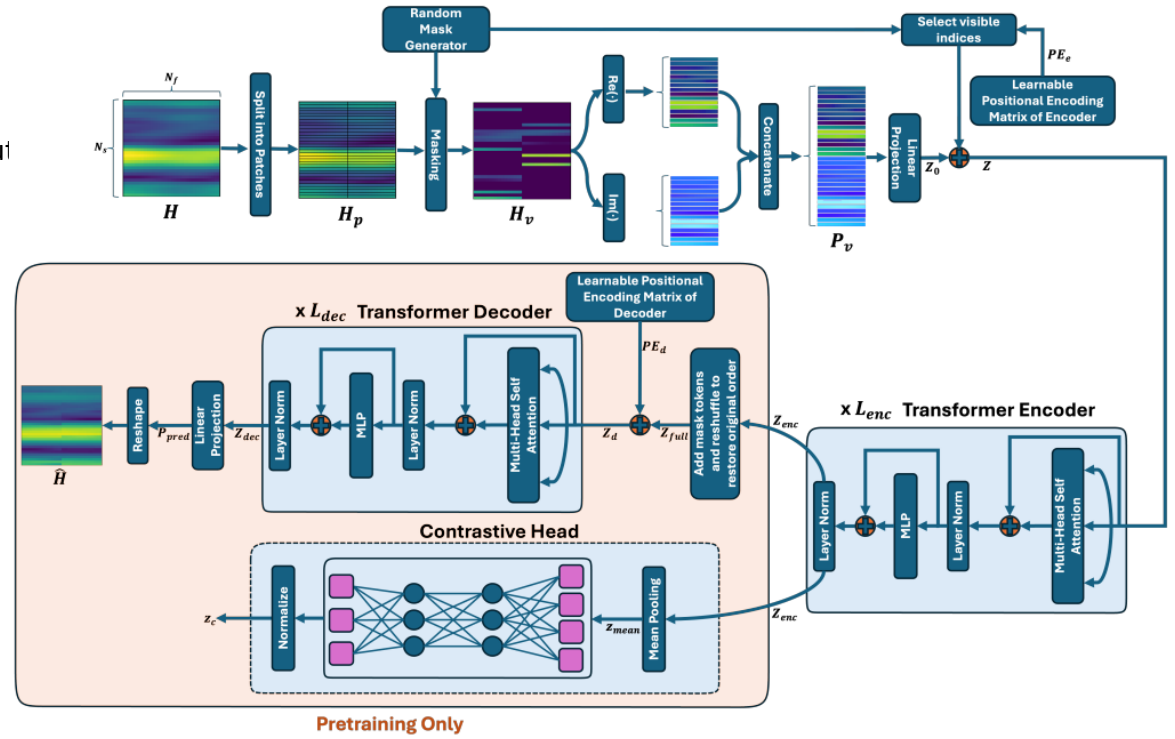
Training objective:

- Masked Channel Modeling (MCM)
  Randomly mask a fraction (between 45% and 90%) of the patches (masking both the real and corresponding imaginary part to avoid leakage); for the values
  - Loss is mean squared error between original (unmasked) patch and prediction from Foundation Model
  - An additional special patch [CLS] (with an associated learnable vector) is appended to each input to obtain a representation for the whole sequence. Furthermore the attention scores with CLS provide importance of each patch(can be used to identify critical segments)

- Additional contrastive learning objective
  Masked version of channel matrix: H_M and *positive channel matrix* obtained by adding AWGN: $H_M^+ \sim \mathcal{CN}(H_M, \sigma^2 I)$

  Let be a learnable non-linear transformation, and $z_i = \mathcal{G}(H_M)$ and $z_i^+ = \mathcal{G}(H_M^+)$
  Then given a batch size $\mathcal{B}$, the loss is

$$\mathcal{L}_{contr} = -\mathbb{E}_{H_i \sim P(H), M \sim P(M)}[\log \frac{\exp(z_i \cdot z_i^+ / \tau)}{\sum_{j=1, j \neq i}^{\mathcal{B}} \exp(z_i \cdot \frac{z_j}{\tau})}]$$



MEDIATEK

"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

Dataset

- DeepMIMO (same as previous paper)

Tasks

- LoS/NLoS and Beam Prediction(same as previous paper)

- Model configurations:

TABLE II: Parameter counts of different configurations.

Model	Encoder	Decoder	C. Head	Total
CWiMAE (12,4)	410,944	143,184	16,576	570,704
WiMAE (6,4)	210,112	143,184	-	353,296
WiMAE (12,4)	410,944	143,184	-	554,128
WiMAE (18,4)	611,776	143,184	-	754,960
WiMAE (12,8)	410,944	277,072	-	688,016
WiMAE (12,12)	410,944	410,960	-	821,904
LWM (12,-)	600,000	-	-	600,000

Note: CWiMAE = ContraWiMAE (contrastive learning variant). Model names show (L_{enc},L_{dec}) where L_{enc} and L_{dec} are encoder and decoder layers. For WiMAE and CWiMAE: M_{enc}=16, M_{dec}=8, d_e=64.

TABLE I: WiMAE and ContraWiMAE pretraining configurations.

Common parameters	**Value**
Training/validation split	80/20
Epochs	3,000
Batch size	3,072
Optimizer	Adam
Learning rate	3×10^{-4} with cosine decay
Attention heads	$M_{\text{enc}} = 16$, $M_{\text{dec}} = 8$
WiMAE	**Value**
Masking ratios m_{r}	$\{0.45,\ \mathbf{0.60},\ 0.75,\ 0.90\}$
Encoder depth L_{enc}	$\{6,\ \mathbf{12},\ 18\}$
Decoder depth L_{dec}	$\{\mathbf{4},\ 8,\ 12\}$
Embedding dim. d_{e}	$\{32,\ \mathbf{64},\ 96,\ 128\}$
Patch size $(N_{\text{p,s}}, N_{\text{p,f}})$	$\{(\mathbf{1}, \mathbf{16}),\ (4, 4)\}$
ContraWiMAE	**Value**
Initialization	Warm-start from WiMAE
Loss coefficient in (4)	$\alpha = 0.9$
Contrastive temperature in (3)	$\tau = 0.1$
Contrastive dimension d_{c}	64
Positive pair generation	AWGN to achieve SNR $\in$ [20, 30] dB

Note: Boldface values are chosen for downstream task evaluation.

"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

Beam Prediction

- With linear probing as downstream model
- Top-1 and Top-3 (a prediction is correct if ranks within top-3) accuracy
- ContraWiMAE achieves the highest accuracy in all training budgets

TABLE III: Top-1/top-3 accuracy for beam selection with linear probing.

Model	Codebook Size (CS)	Training Budget						
		1%	2%	5%	10%	25%	50%	100%
WiMAE	16	**54.8/77.1**	62.7/82.8	67.0/85.2	73.1/86.1	79.7/90.6	81.4/91.4	79.2/87.1
	32	**39.9/58.2**	47.2/66.0	55.3/74.6	61.7/74.9	72.9/82.5	74.8/81.0	72.4/76.6
	64	23.6/40.8	31.1/50.9	30.7/45.4	40.8/57.0	55.3/68.9	57.8/66.8	64.1/72.9
	128	12.7/22.0	16.9/28.0	17.5/30.2	23.6/35.4	31.5/45.3	35.6/47.6	39.8/46.6
	256	6.8/12.5	10.4/**19.0**	11.7/19.4	12.7/20.0	17.6/26.6	22.9/31.5	23.3/31.7
ContraWiMAE	16	54.2/76.4	**65.3/85.9**	**73.8/89.1**	**76.2/87.9**	**81.3/92.5**	**83.0/91.6**	**85.0/92.6**
	32	39.0/55.2	**53.0/74.5**	**64.8/81.2**	**77.0/88.6**	**84.0/94.3**	**85.4/92.3**	**85.6/92.7**
	64	**24.9/41.2**	**32.5/51.8**	**44.7/66.7**	**55.0/74.6**	**67.7/84.4**	**75.8/88.3**	**74.7/82.0**
	128	**15.0/24.9**	**18.3/32.4**	**25.0/41.0**	**28.8/46.6**	**42.5/62.8**	**51.4/70.2**	**55.9/70.6**
	256	**7.9/13.5**	**10.5**/18.7	**14.7/24.1**	**17.3/27.2**	**22.5/35.4**	**27.4/41.5**	**31.3/45.2**
LWM	16	17.5/27.5	23.6/35.3	19.0/32.2	26.0/36.9	36.8/48.5	39.6/47.9	42.7/53.6
	32	18.8/27.3	19.0/26.1	16.3/22.5	26.6/34.3	34.2/41.0	40.1/47.2	43.3/49.1
	64	14.8/20.7	18.9/25.8	11.8/16.4	21.3/27.2	27.8/35.0	33.0/39.7	36.7/42.6
	128	10.0/13.5	12.0/19.1	8.2/13.1	14.1/20.5	19.2/25.5	24.1/29.7	25.1/29.2
	256	5.3/8.0	8.6/12.9	6.0/10.4	8.3/12.2	11.6/17.1	13.7/18.8	13.7/19.2
RAW	16	31.6/41.5	33.9/42.8	34.0/43.7	38.2/45.8	38.8/46.3	39.8/44.4	41.7/48.0
	32	28.5/38.9	27.6/37.5	37.1/45.8	37.6/45.0	37.5/44.7	41.2/47.5	42.6/47.6
	64	19.4/29.2	25.2/36.2	29.3/41.6	32.0/42.8	34.6/43.1	37.1/44.5	36.5/42.2
	128	10.9/19.3	15.0/25.3	20.7/32.1	24.0/35.0	25.3/36.8	29.6/40.6	30.7/38.0
	256	6.2/10.9	9.9/15.8	11.5/19.1	15.0/24.9	18.7/26.1	20.0/31.5	22.7/31.7

Note: Boldface indicates the best result for each (CS, training budget) pair.

"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

Beam Prediction

- Same as previous slide but with with a more powerful downstream model (Resnet)
- Performance decreases for lower training budgets, but increases for higher ones (larger downstream models require more data)

TABLE III: Top-1/top-3 accuracy for beam selection with ResNet-Slim and ResNet-Wide.

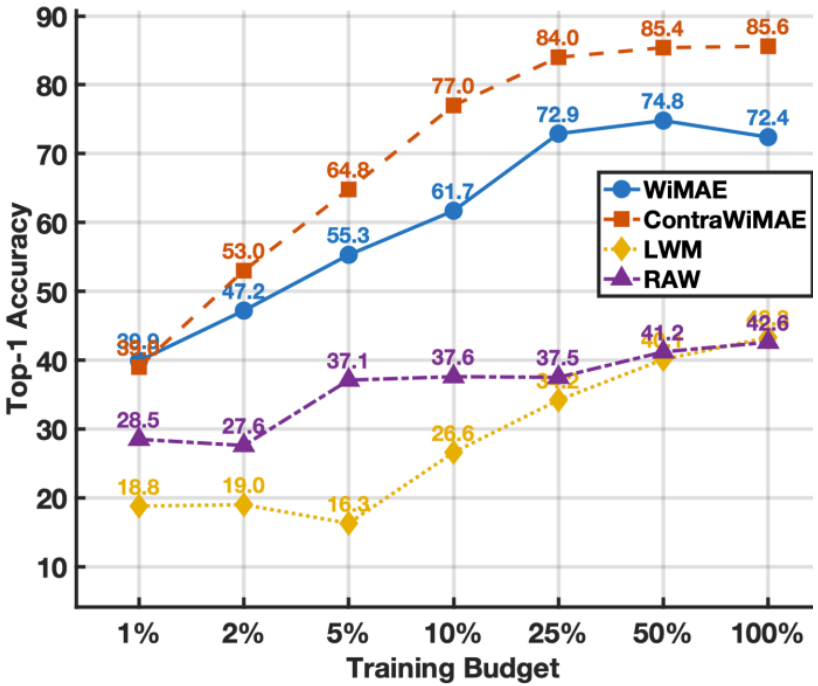
Model	Codebook Size (CS)	Training Budget							
		ResNet-Slim				ResNet-Wide			
		10%	25%	50%	100%	10%	25%	50%	100%
WiMAE	16	8.9/23.8	37.8/71.6	42.9/74.7	53.7/82.2	6.9/24.7	55.6/84.8	61.3/86.1	68.9/89.2
	32	13.4/27.1	25.7/47.5	39.2/70.1	52/79.6	28.4/51.5	47.3/76.5	60.2/86.9	71.1/92.1
	64	2.2/6.1	11.1/27.8	24.1/49	36.8/65.4	14.7/30.3	32.2/59.3	47.6/77.2	60.7/86.7
	128	0.7/2.8	6.4/13.7	12.7/30.7	24.4/50.3	9.2/18.1	17.5/35.8	32.6/61.9	44.3/76.1
	256	0.2/1.6	0.4/1.3	8.1/17.5	13.1/29.4	5.5/10	8.8/17.9	15/33.9	22.2/51.2
ContraWiMAE	16	**31.2/55**	**56.7/80.7**	**68.9/88.8**	**73.9/92.3**	**65.5/87.6**	**74.6/91.9**	**77/93.7**	**79/93.7**
	32	**15.7/33.6**	**46.8/76.8**	**64.7/88.8**	**73.8/93.3**	**49.3/75.3**	**72/93**	**79.1/95**	**81.9/95.8**
	64	**10.4/22.6**	**24.2/49.2**	**40.4/70.8**	**63/88.9**	**26.4/51.4**	**55.5/82.9**	**69.2/93.1**	70.7/**93.5**
	128	**3.2/8**	**11.6/29.9**	**27.3/57.5**	**46.6/78.9**	**15/30.8**	**31.9/57.4**	**50.4/82.1**	**59.6/88.5**
	256	**2.9/5.7**	**6.6/14.9**	**12.7/29.2**	**24.5/56.3**	**7.6/13.8**	**12.4/29.8**	**24.9/53.1**	**39.4/74.7**
LWM	16	23.3/50.9	49.3/79.1	55.1/85	64.4/88.8	49.2/80.5	65.6/90.1	71.5/92.6	77.3/93.3
	32	12/30	35/66.2	48.7/79.9	69.8/92.9	33.1/59.6	58.4/86.2	70.7/93.6	77/95.4
	64	2/5	15.7/35.4	29.3/62.1	51.6/84.1	17.9/38.7	39.1/72.1	55.1/87.2	64.2/92.6
	128	1.1/3.3	9/20.8	11.8/32.6	28.4/65.2	8/17.8	14.3/35.4	30.9/67.6	47.2/81.4
	256	0.4/1.5	4.5/10.6	7.1/16.3	11.9/35.7	4.7/9.4	8/18.9	14.2/32.6	24.3/55.3
RAW	16	9.5/22	29.4/55.7	57.7/80.3	70/88	28/52.4	47.4/74.6	69/88.4	75.9/91.1
	32	3.4/12.1	26.9/49.1	43.3/72.2	63.1/86.8	23.9/41	46/74.3	67.5/88.6	80.3/94.3
	64	2.6/7.4	15.6/30.7	31.4/53.1	57.4/84.9	13.8/27	35.4/58.8	51.1/79.6	**73.4**/92.4
	128	1.7/4.2	2.3/5.7	16.3/36.6	34.6/62.1	1/3.4	22.3/43.4	37.5/62.5	55.9/81.5
	256	1/2.4	0.8/1.8	10.4/23.8	18.7/41.5	0.7/3.4	10.1/21.2	21.3/44.1	38.3/69.3

Note: Best results for slim and wide are bolded and both bolded and underlined, respectively.

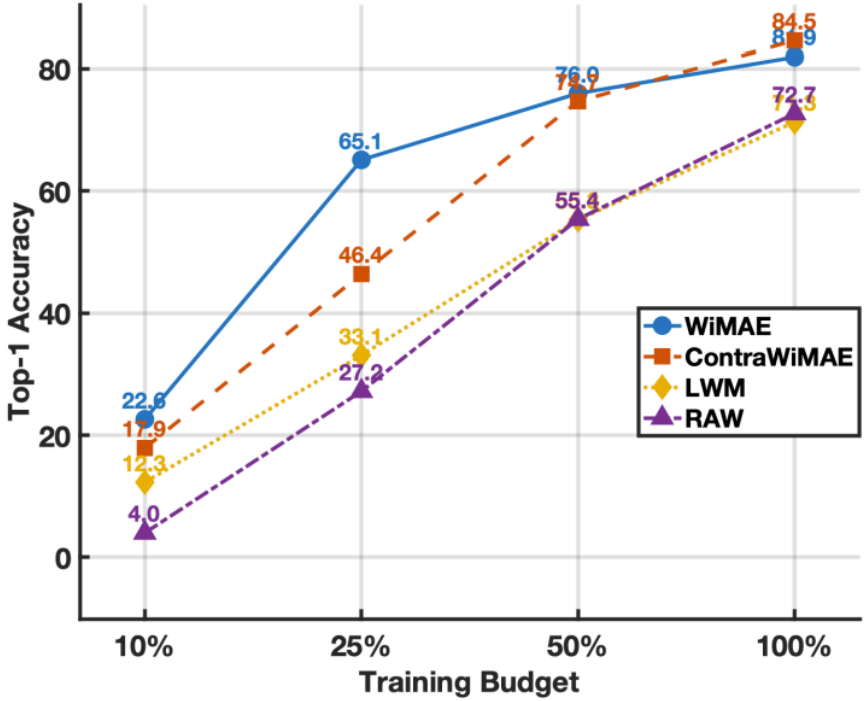
"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

Downstream model analysis - Beam selection

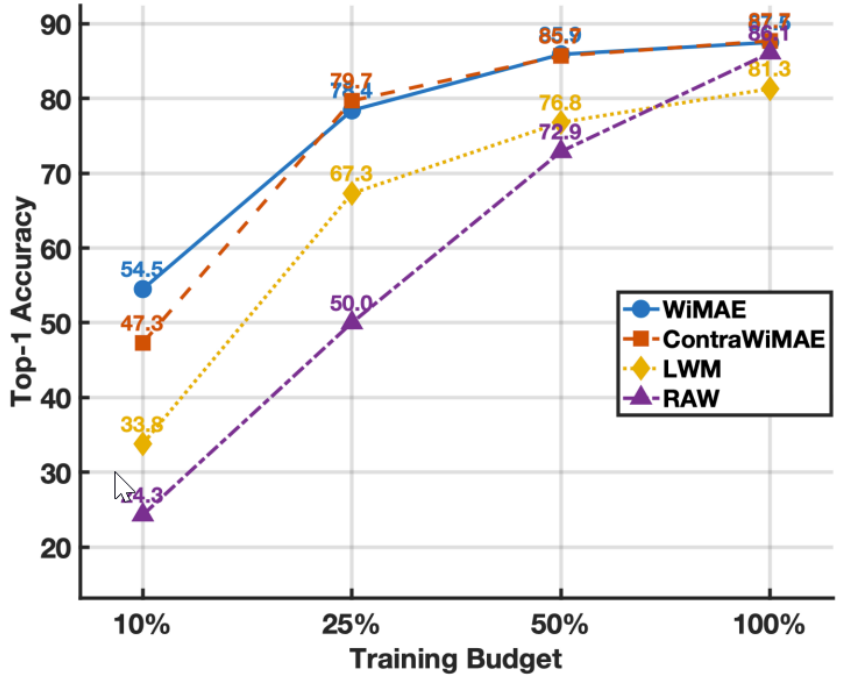
Plots for codebook size of 32. Highest improvement comes for linear-probing scenario (stronger downstream model can handle "worse" representations)



(a) Beam selection with linear probing, codebook size 32.



(b) Beam selection with ResNet-Slim, codebook size 32.



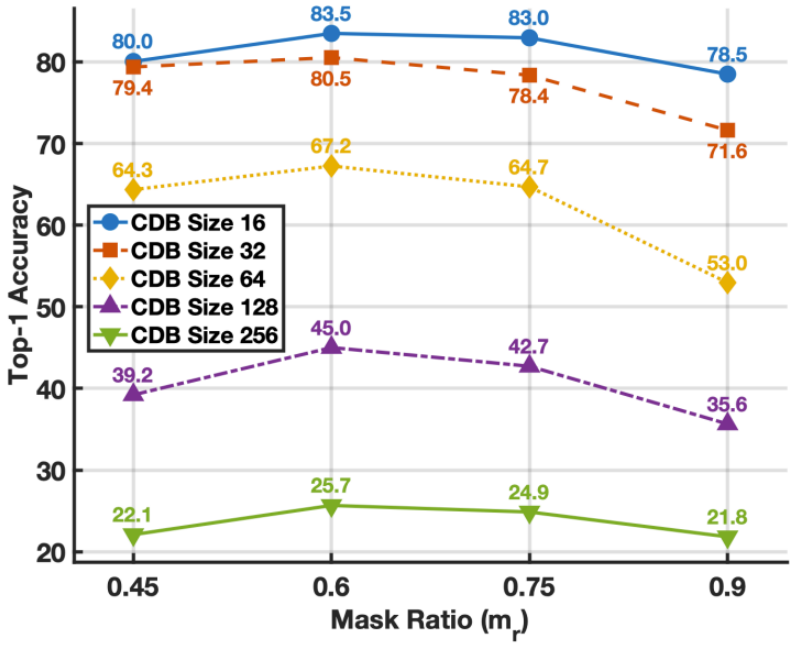
(c) Beam selection with ResNet-Wide, codebook size 32.

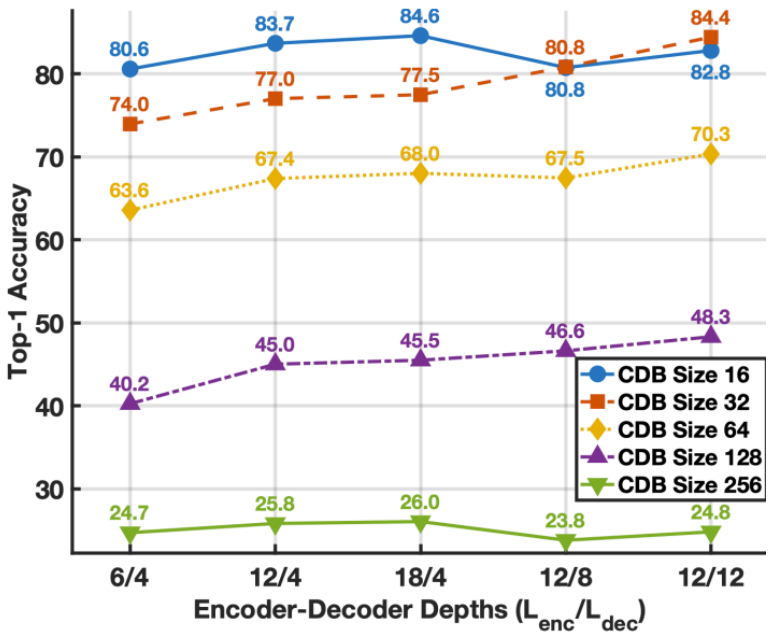
MEDIATEK

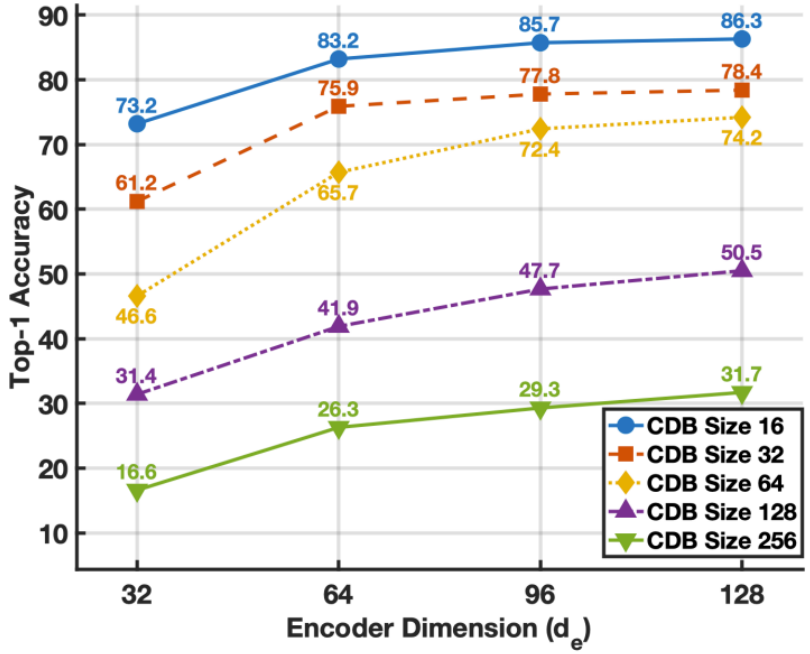
"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

Analysis of hyperparameters - Beam Prediction

Results with different mask ratios, encoder-decoder depths, and dimension







Mask ratio should be tuned carefully

Higher complexity seems to improve performance, but saturates quickly

"A Multi-Task Foundation Model for Wireless Channel Representation Using Contrastive and Masked Autoencoder Learning", Guler et al.

LoS detection

- Embeddings are mean-pooled and passed to linear classifier
- Similar to beam prediction, ContraWiMAE shines in low training budgets

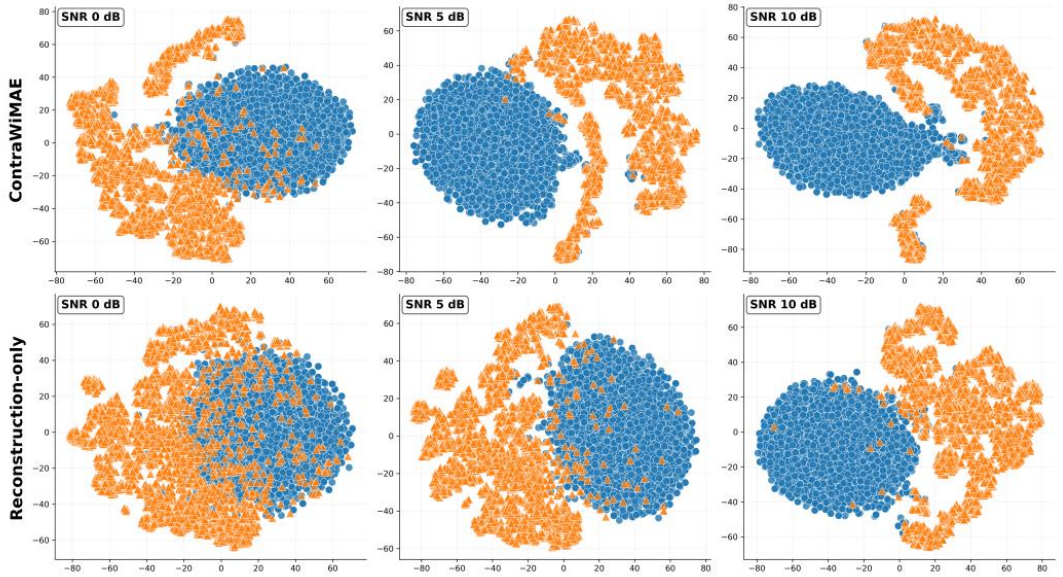


Fig. 6: t-SNE representation with LoS (blue) and non-LoS (orange) labels under varying SNR

TABLE V: Accuracy in the LoS detection task.

Training Budget (%)	LWM			WiMAE			ContraWiMAE		
	Accuracy	F1	AUC	Accuracy	F1	AUC	Accuracy	F1	AUC
1	0.929	0.941	**0.981**	0.931	**0.944**	0.980	**0.931**	**0.944**	**0.981**
2	**0.946**	**0.957**	**0.985**	0.941	0.953	0.983	0.941	0.953	0.983
5	0.926	0.942	0.973	0.944	0.955	0.985	**0.949**	**0.959**	**0.987**
10	0.943	0.955	0.987	0.947	0.958	**0.989**	**0.950**	**0.960**	**0.989**
25	0.946	0.957	0.988	**0.957**	**0.965**	**0.992**	0.951	0.960	0.990
50	0.950	0.960	0.989	**0.958**	**0.966**	**0.992**	0.956	0.965	**0.992**
100	0.948	0.959	0.989	**0.960**	**0.968**	**0.993**	0.959	0.967	0.992

Note: Boldface indicates best result for each training budget.

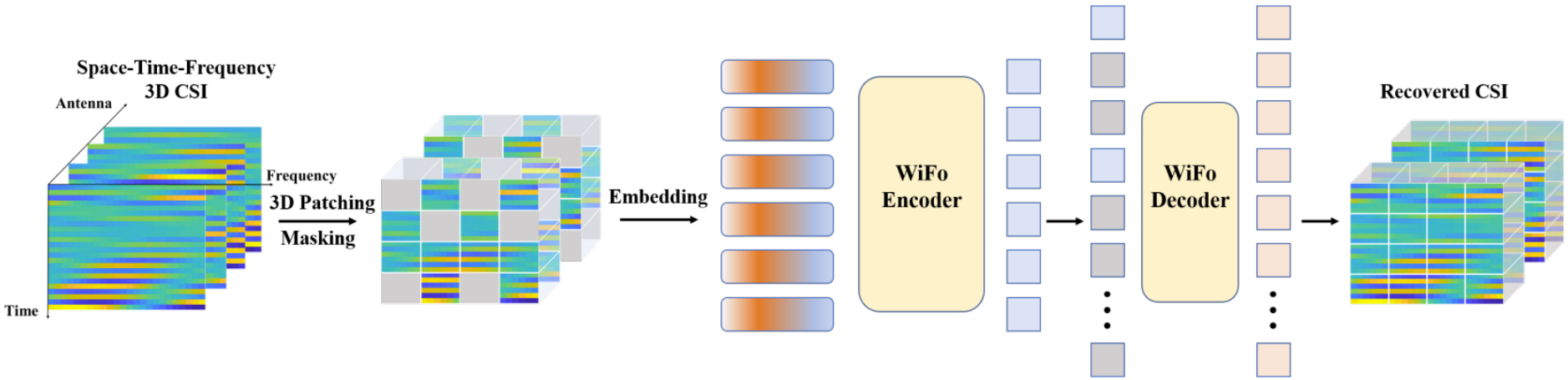
"WiFo: wireless foundation model for channel prediction", Liu et al.

Input

- Space-time-frequency CSI is indicated as $H \in \mathbb{C}^{T \times K \times N}$ with T resource blocks in Time, K resource blocks in frequency, and N elements in uniform planar array (user's grid of antenna elements)
  - Converted into real valued tensor as $H \in \mathbb{C}^{2 \times T \times K \times N}$
  - 3D patching is applied (to the last 3 dimensions), then flattening and embedding

Model

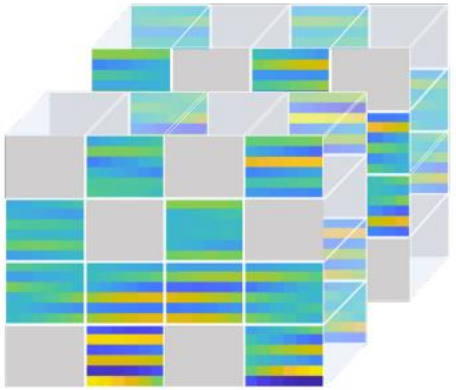
- Masked autoencoder architecture
  - Encoder is a transformer with 3D positional embeddings (for time, frequency, and space). Encoder operates only with the non-masked patches
  - Decoder is a transformer with 3D positional embeddings receiving output of encoder with added mask tokens



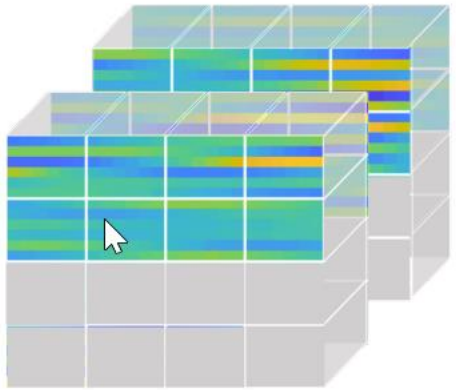
“WiFo: wireless foundation model for channel prediction”, Liu et al.

Training

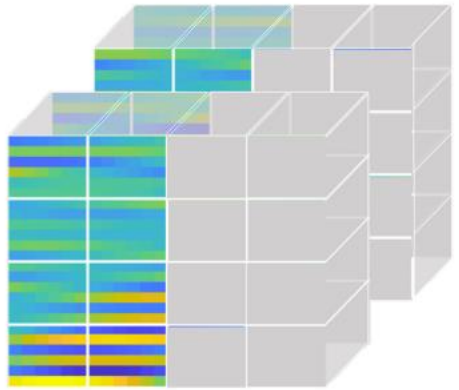
- three pre-training tasks are executed sequentially for each batch
  - Random mask
  - Time-mask
  - Frequency mask



Random-masked reconstruction



Time-masked reconstruction



Frequency-masked reconstruction

"WiFo: wireless foundation model for channel prediction", Liu et al.

Dataset

- Series of diverse 3D CSI datasets, generated through channel generator QuaDRiGa [Jaeckel et al., IEEE Trans Antennas Propagat, 2014] compliant with the 3GPP standards
  - Seven 5G New Radio (NR) frequency bands
  - Eight 3GPP scenarios
  - Seven user speed ranges
  - For each CSI sample, the user has a random initial position and a straight-line motion trajectory
  - Each dataset contains 12000 samples, which are randomly split into 9000, 1000, and 2000 samples for training, validation, and inference, respectively

- total of 19 datasets are simulated, indexed from D1 to D19, covering various space-time-frequency CSI configurations, cell scenarios, and user speed. 16 datasets are used for pre-training, and the remaining for testing

- 5 different model size (0.3M to 86M)

Table 1 An illustration of the system configurations of the constructed 3D CSI datasets.

Dataset	f_C(GHz)	K	Δf(kHz)	T	Δt(ms)	UPA	Scenario	User speed(km/h)
D1	1.5	128	90	24	1	1×4	UMi+NLoS	3-50
D2	1.5	128	180	24	0.5	2×4	RMa+NLoS	120-300
D3	1.5	64	90	16	1	1×8	Indoor+LoS	0-10
D4	1.5	32	180	16	0.5	4×8	UMa+LoS	30-100
D5	2.5	64	180	24	0.5	2×2	RMa+NLoS	120-300
D6	2.5	128	90	24	1	2×4	UMi+LoS	3-50
D7	2.5	32	360	16	0.5	4×8	UMa+LoS	30-100
D8	2.5	64	90	16	1	4×4	Indoor+NLoS	0-10
D9	4.9	128	180	24	1	1×4	UMi+NLoS	3-50
D10	4.9	64	180	24	0.5	2×4	RMa+LoS	120-300
D11	4.9	64	90	16	0.5	4×4	UMa+NLoS	30-100
D12	4.9	32	180	16	1	4×8	Indoor+LoS	0-10
D13	5.9	64	90	24	0.5	2×8	RMa+LoS	120-300
D14	5.9	128	180	24	1	2×4	UMi+NLoS	3-50
D15	5.9	64	90	16	1	4×4	Indoor+LoS	0-10
D16	5.9	32	360	16	0.5	4×8	UMa+NLoS	30-100
D17	3.5	32	180	16	0.5	4×8	UMa+NLoS	30-100
D18	6.7	64	180	24	1	4×4	UMi+LoS	3-50
D19	28	32	360	16	0.25	4×8	UMa+LoS	30-100

Table 2 Network parameters of WiFo with different sizes.

Model	Enc. depth	Enc. width	Enc. heads	Dec. depth	Dec. width	Dec. heads	Parameters
WiFo-Tiny	6	64	8	4	64	8	0.3M
WiFo-Little	6	128	8	4	128	8	1.4M
WiFo-Small	6	256	8	4	256	8	5.5M
WiFo-Base	6	512	8	4	512	8	21.6M
WiFo-Large	8	768	8	4	768	8	86.1M

Table 3 Pre-training parameters of WiFo.

Parameter	Value
Optimizer	AdamW ($\beta_1 = 0.9, \beta_1 = 0.999$, weight decay=0.05)
Batch size	128
Epochs	200
Learning rate schedule	Cosine decay [25] (warmup epochs = 5)
Base learning rate	5×10^{-4}

"WiFo: wireless foundation model for channel prediction", Liu et al.

Evaluation

- Time-domain channel prediction
- Frequency-domain channel prediction

Table 4 The NMSE performance of WiFo-Base and other baselines on the time-domain channel prediction task across the D1-D16 datasets. The best results are highlighted in **bold**, while the second-best results are underlined.

Dataset	WiFo-Base	Transformer	LSTM	3D ResNet	PAD	LLM4CP	LLM4CP*
D1	0.082	0.112	0.356	0.088	0.529	0.117	**0.074**
D2	**0.260**	0.416	0.797	0.351	1.074	0.451	0.305
D3	0.016	0.016	0.027	0.014	0.038	0.015	**0.013**
D4	**0.048**	0.107	0.418	0.055	0.317	0.106	0.060
D5	**0.494**	0.638	0.788	0.751	5.008	0.637	0.510
D6	**0.095**	0.174	0.542	0.157	0.568	0.206	0.133
D7	**0.081**	0.219	0.576	0.103	0.617	0.198	0.112
D8	0.018	0.024	0.092	**0.016**	0.073	0.025	**0.016**
D9	0.347	0.483	0.835	0.349	1.087	0.475	**0.312**
D10	**0.467**	0.649	0.689	0.869	3.863	0.709	0.563
D11	**0.227**	0.440	0.834	0.274	1.017	0.405	0.273
D12	**0.023**	0.035	0.166	0.025	0.132	0.035	0.026
D13	**0.482**	0.718	0.876	0.815	5.213	0.758	0.648
D14	0.369	0.546	0.884	0.388	1.021	0.562	**0.358**
D15	**0.029**	0.039	0.156	0.032	0.151	0.038	0.030
D16	**0.318**	0.591	0.944	0.329	1.034	0.545	0.349
Average	**0.210**	0.325	0.561	0.289	1.359	0.330	0.236

Table 5 The NMSE performance of WiFo-Base and other baselines on the frequency-domain channel prediction task across the D1-D16 datasets. The best results are highlighted in **bold**, while the second-best results are underlined.

Dataset	WiFo-Base	Transformer	LSTM	3D ResNet	LLM4CP	LLM4CP*
D1	**0.318**	0.532	0.705	0.839	0.392	0.375
D2	**0.181**	0.556	0.763	0.647	0.419	0.223
D3	0.027	**0.016**	0.037	0.071	0.023	0.025
D4	**0.073**	0.270	0.475	0.215	0.211	0.151
D5	**0.152**	0.315	0.577	0.386	0.267	0.165
D6	**0.081**	0.310	0.540	0.458	0.193	0.140
D7	**0.092**	0.392	0.578	0.354	0.318	0.189
D8	0.061	**0.024**	0.348	0.139	0.068	0.069
D9	0.436	0.481	0.895	0.918	0.574	**0.418**
D10	**0.087**	0.261	0.451	0.257	0.163	0.096
D11	**0.245**	0.723	0.859	0.823	0.621	0.349
D12	**0.023**	0.048	0.131	0.029	0.032	0.026
D13	0.068	0.238	0.531	0.177	0.165	**0.067**
D14	**0.395**	0.744	0.911	0.924	0.637	0.414
D15	**0.023**	0.053	0.083	0.045	0.024	0.024
D16	**0.270**	0.855	0.929	0.723	0.712	0.456
Average	**0.158**	0.364	0.551	0.438	0.301	0.199

"WiFo: wireless foundation model for channel prediction", Liu et al.

Ablation study

The average performance is measured by the mean NMSE of columns 2 to 7.

- Replacing the proposed STFPE with learnable space-time positional encoding degrades performance.
- Removing the random-masked reconstruction task degrades performance for both the time-domain and frequency-domain channel prediction. "It can be attributed that the additional random masking strategy facilitates WiFo's ability to capture the intrinsic 3D relationships of CSI."
- Removing time-masked or frequency-masked reconstruction enhances frequency-domain or time-domain channel prediction performance but significantly degrades the other.

Table 6 Results of ablation experiments. The best results are highlighted in **bold**, while the second-best results are underlined.

	Time-domain prediction			Frequency-domain prediction			
	D1-D16	D17	D18	D1-D16	D17	D18	**Average**
WiFo-Base	0.210	**0.305**	0.420	0.158	0.229	0.280	**0.267**
w/ learnable PE [19]	0.224	0.354	0.442	0.154	0.220	0.267	0.277
w/o random-masked reconstruction	0.214	0.317	0.435	0.165	0.233	0.294	0.276
w/o time-masked reconstruction	0.497	0.754	0.723	**0.145**	**0.199**	**0.257**	0.429
w/o frequency-masked reconstruction	**0.205**	0.310	**0.412**	0.472	0.819	0.656	0.479

"Addressing the Curse of Scenario and Task Generalization in AI-6G: A Multi-Modal Paradigm", Jiao et al.

Input

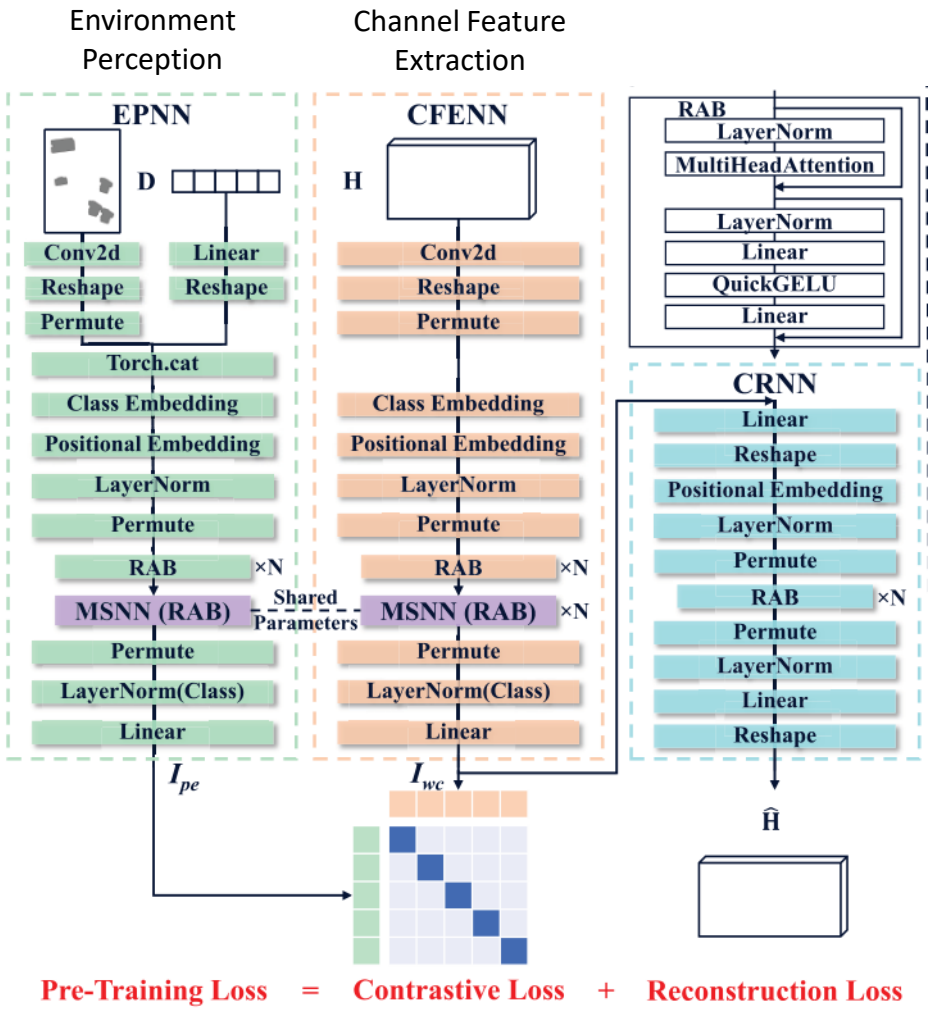
- Physical data (D): matrices like geographic maps and structured vectors such as BS and UE positions
- Wireless channel data (H): CSI matrix

Two-tower architecture

- Idea: data from the physical environment and the wireless channel contain key information that explicitly or implicitly defines propagation paths.
  - green tower, EPNN, and the orange tower, CFENN, to extract features from the physical environment data D and the wireless channel data H
  - CRNN network for CSI matrix reconstruction

Training

- Contrastive loss between the representations from the two towers
  - Aligns representations of EPNN and CFENN
- CSI reconstruction
  - Ensures representation from CFENN encode all relevant information



MEDIATEK

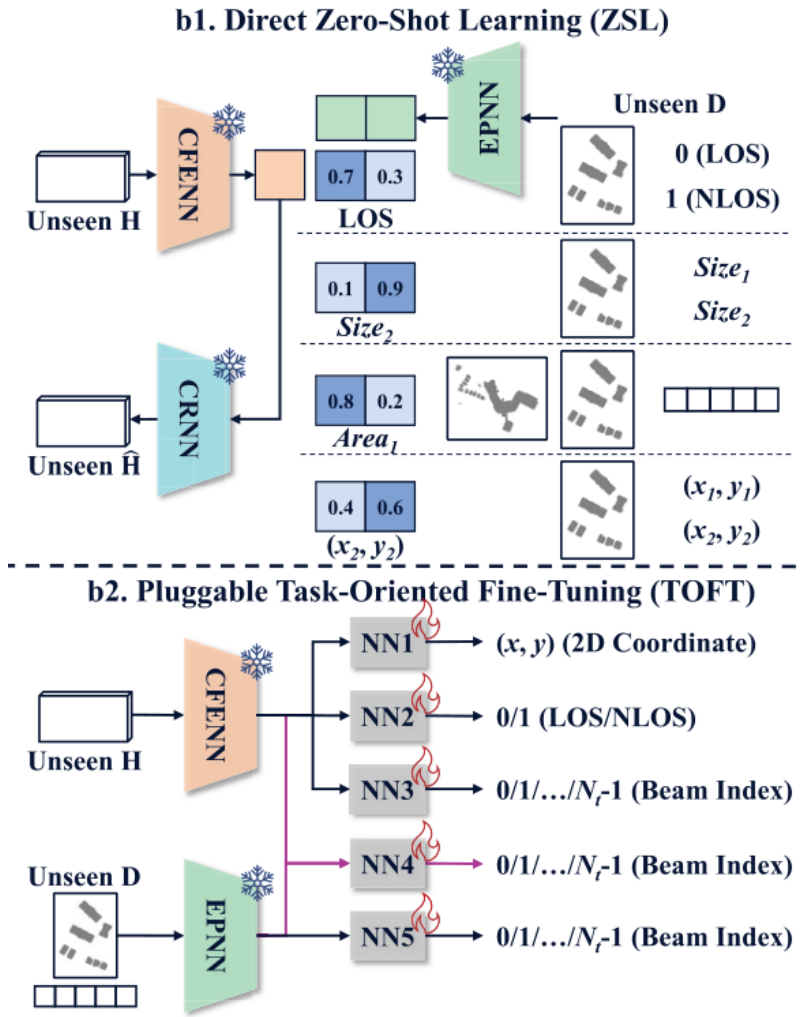
"Addressing the Curse of Scenario and Task Generalization in AI-6G: A Multi-Modal Paradigm", Jiao et al.

Inference

- Zero-shot learning: leverages the alignment of various modalities and directly predicts using the trained modules
  - E.g., given the representations from CFENN and CRNN, we can look at the class with highest "alignment" to select the answer

$$R_{\text{los}} = \text{argmax}(I_{pe1} {I_{wc1}}^T, I_{pe2} {I_{wc1}}^T)$$

- Task-Oriented Fine-tuning approach: plugging a lightweight task-specific NN after the frozen pre-trained model for supervised training
  - Requires gathering some data to train the add-on network

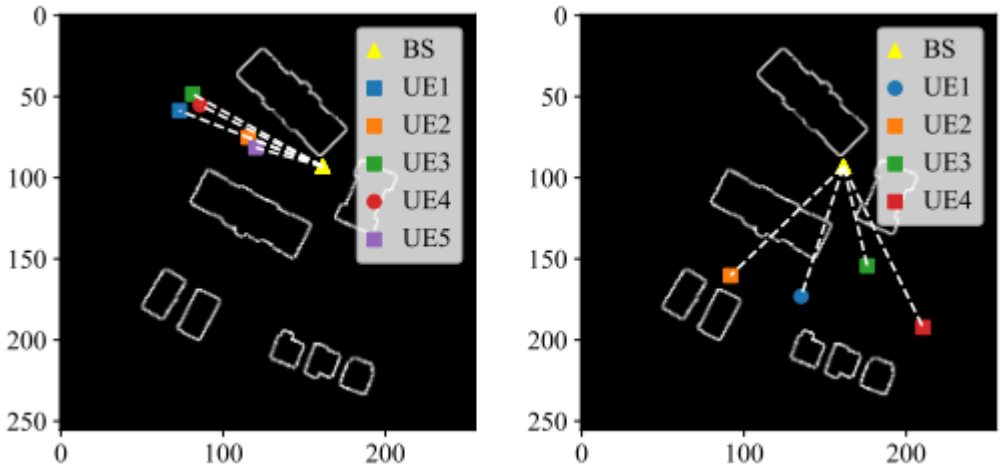


MEDIATEK

"Addressing the Curse of Scenario and Task Generalization in AI-6G: A Multi-Modal Paradigm", Jiao et al.

Dataset

- wireless AI research dataset (WAIR-D) [Huangfu et al., 2022]
  - randomly chooses 10,000 real-world areas of varying sizes from over 40 cities and provides their building layout information
    - 9000 used for training, 1000 for downstream task adaptation.
  - data generated by WAIR-D includes BS and UE positions, UE LOS status, physical-to-pixel scaling factor, area top-view map, and CSI



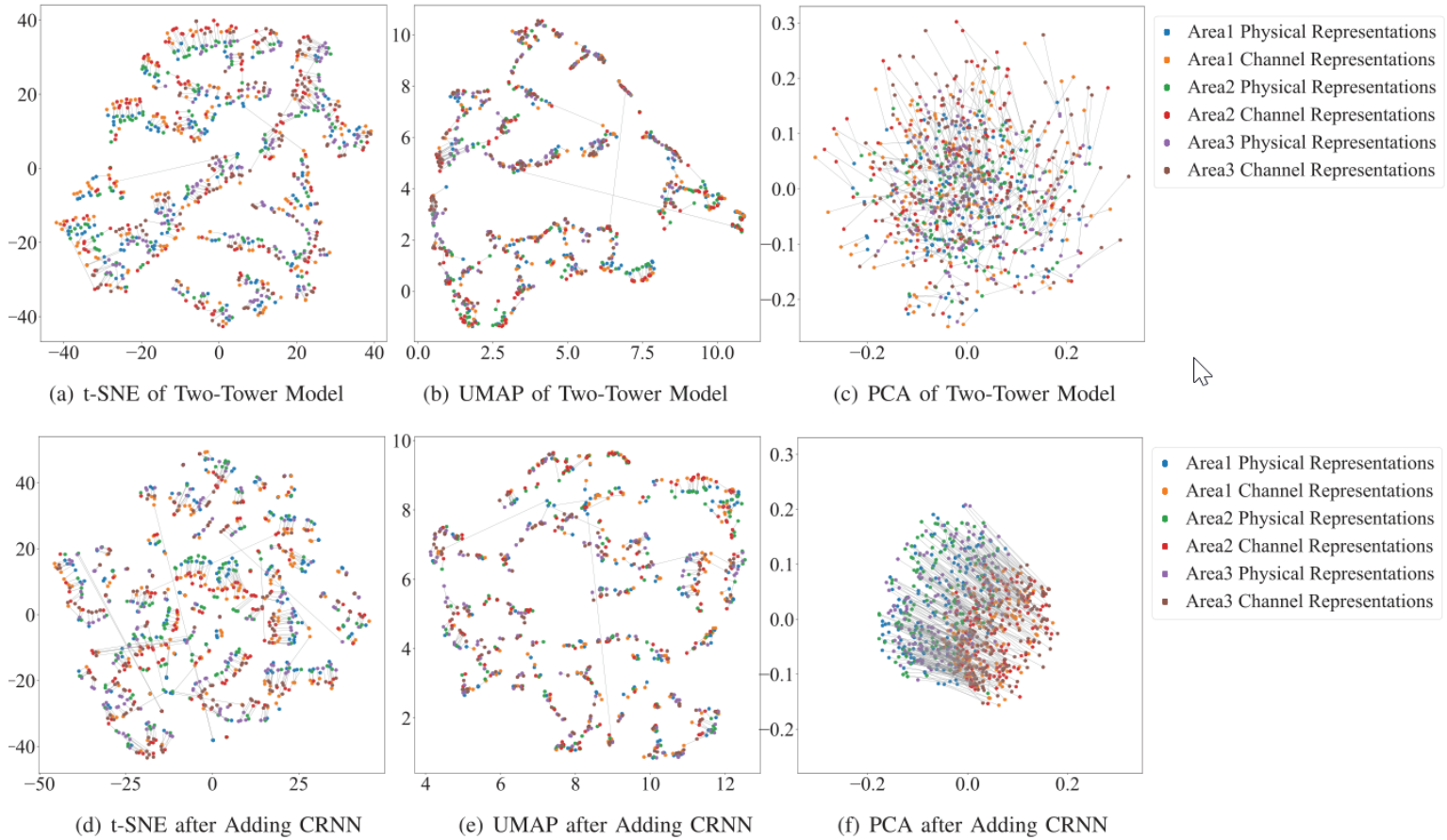
PARAMETERS OF THE WAIR-D DATASET

Parameter	Value
Carrier Frequency	28GHz
Bandwidth	46.08MHz
Sub-Carrier Number	64
Antenna Configuration	32 ULA Tx Ports, 1 Rx Port
BS and UE Height	6m, 1.5m
Pre-Training Dataset	$9000 \times 5 \times 30 + 90 \times 1 \times 10000$ samples
ZSL and TOFT Dataset	$1000 \times 5 \times 30 + 10 \times 1 \times 10000$ samples

"Addressing the Curse of Scenario and Task Generalization in AI-6G: A Multi-Modal Paradigm", Jiao et al.

Effect of adding reconstruction loss

- The t-SNE and UMAP results resemble those of the two-tower model, indicating that the nonlinear relationships are unaffected.
- However, the PCA result shows that the directions of the lines connecting both modalities are consistent, marking the emergence of linear relationships in the embedding space
- Looking at the embedding space can be a useful debugging and development tool



MEDIATEK

"Addressing the Curse of Scenario and Task Generalization in AI-6G: A Multi-Modal Paradigm", Jiao et al.

Evaluation

- 4 tasks
- Compare zero-shot learning (ZSL) with task-oriented fine-tuning (TOFT)
- Compare single-area vs multi-area training

TASK TYPES, INPUTS, OUTPUTS, AND ADAPTATION METHODS OF THE FOUR EXEMPLARY DOWNSTREAM TASKS

Downstream Task	Task Type	Input	Output	Adaptation Method
CSI Feedback	Regression	CSI	CSI	ZSL
Direct Positioning	Regression	CSI	UE Position	TOFT
CSI/Position-based Beam Selection	Classification	CSI/Position	Beam Index	TOFT
LOS/NLOS Identification	Classification	CSI	UE LOS Status	ZSL, TOFT

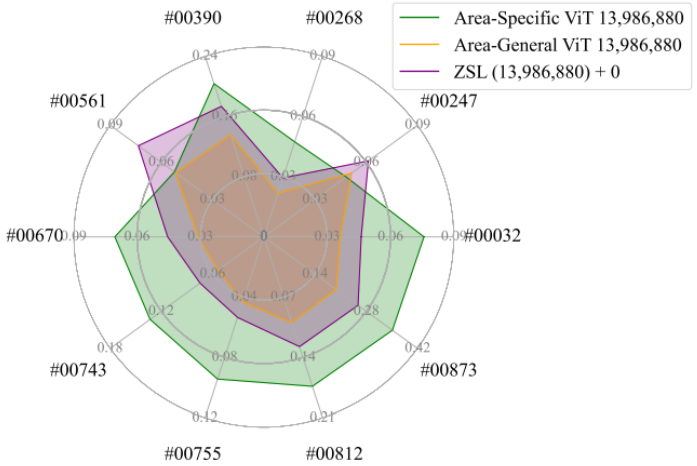


Fig. 9. The NMSE of CSI feedback and parameter size for Area-Specific ViT, Area-General ViT, and our ZSL methods in 10 unseen areas of S2.

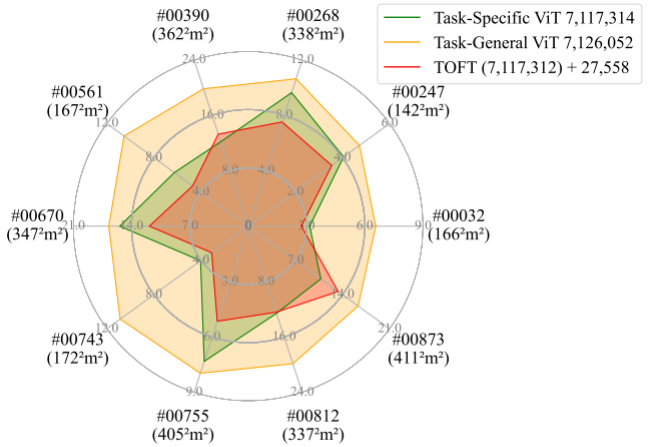


Fig. 10. The single-BS positioning error (at CDF90 in meters) and the number of parameters for Task-Specific ViT, Task-General ViT, and the proposed TOFT methods, as well as the area physical size, in 10 unseen areas of S2.

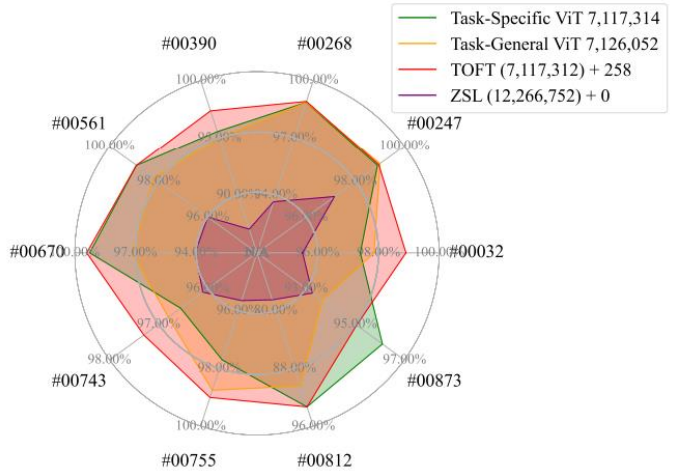


Fig. 13. The LOS/NLOS identification accuracy and parameter size for Task-Specific ViT, Task-General ViT, and our TOFT and ZSL in 10 unseen areas.

"Addressing the Curse of Scenario and Task Generalization in AI-6G: A Multi-Modal Paradigm", Jiao et al.

Evaluation

- 4 tasks
- Compare zero-shot learning (ZSL) with task-oriented fine-tuning (TOFT)
- Compare single-area vs multi-area training

TASK TYPES, INPUTS, OUTPUTS, AND ADAPTATION METHODS OF THE FOUR EXEMPLARY DOWNSTREAM TASKS

Downstream Task	Task Type	Input	Output	Adaptation Method
CSI Feedback	Regression	CSI	CSI	ZSL
Direct Positioning	Regression	CSI	UE Position	TOFT
CSI/Position-based Beam Selection	Classification	CSI/Position	Beam Index	TOFT
LOS/NLOS Identification	Classification	CSI	UE LOS Status	ZSL, TOFT

TOP-1 ACCURACY AND PARAMETER SIZE OF THE BEAM SELECTION TASK FOR DIFFERENT METHODS IN 10 UNSEEN AREAS OF S2

Method	#00032	#00247	#00268	#00390	#00561	#00670	#00743	#00755	#00812	#00873
Task-Specific ViT 7,125,024	95.55%	94.70%	96.55%	94.45%	97.15%	95.85%	95.40%	95.65%	94.35%	93.25%
Task-General ViT 7,126,052	86.75%	87.65%	85.50%	74.30%	87.55%	75.20%	77.25%	68.40%	67.40%	70.70%
CSI-based TOFT (7,117,312)+10,336	95.45%	95.45%	96.10%	93.85%	97.10%	95.90%	94.85%	94.80%	94.85%	92.90%
MLP 10,720	91.20%	85.85%	94.70%	82.35%	91.60%	93.35%	88.15%	90.70%	85.30%	76.85%
Position-based TOFT (7,518,720)+10,336	92.00%	89.05%	95.50%	85.15%	92.65%	94.60%	90.30%	93.10%	88.40%	80.85%

"ChannelGPT: A Large Model toward Real-World Channel Foundation Model for 6G Environment Intelligence Communication", Yu et al.

Input:

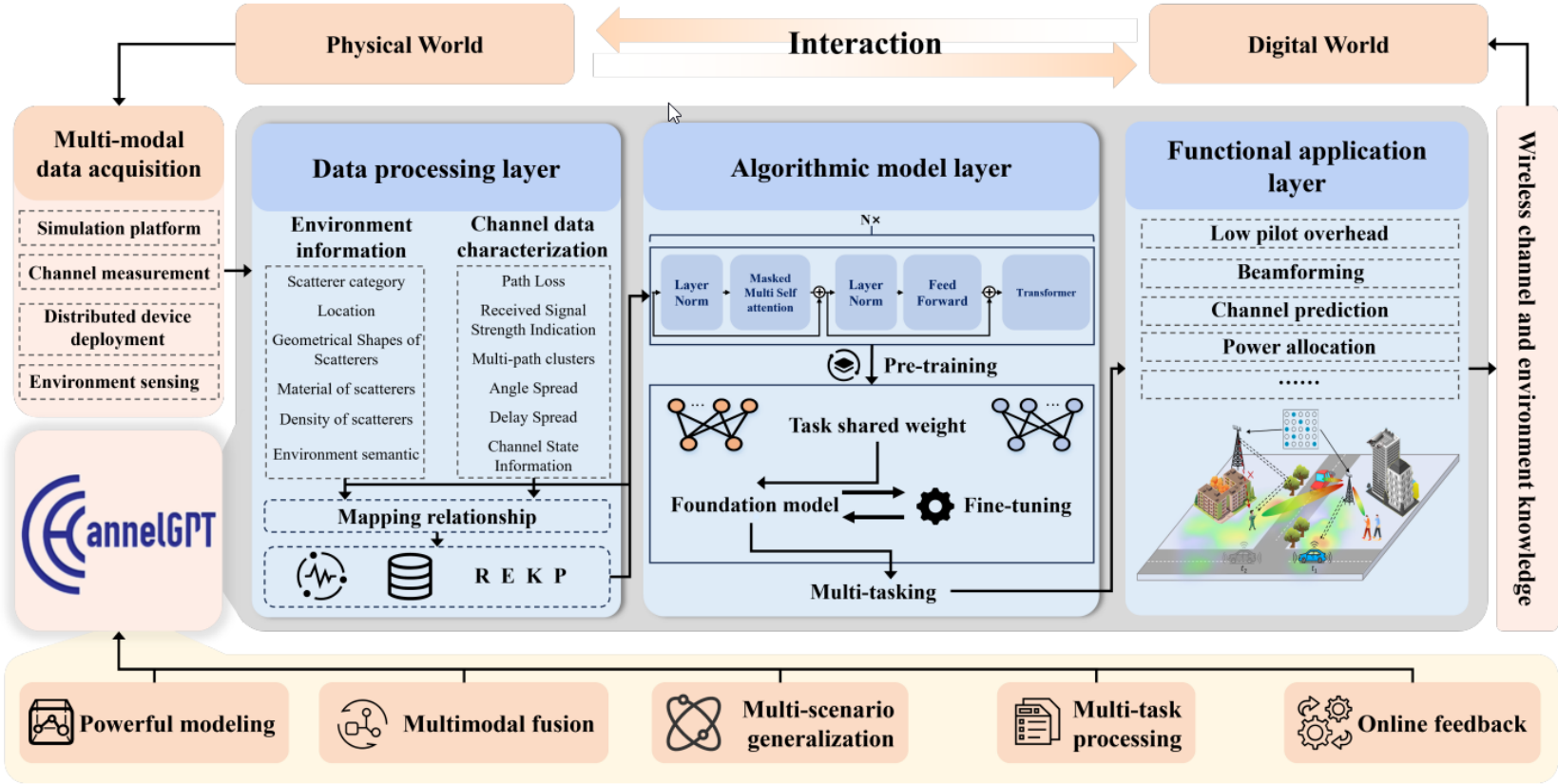
- historical CSI of 25 time slots is utilized

Training task:

- predict the future CSI of N consecutive time slots with a prediction slot size ranging from 1 to 20

Framework:

- Data processing layer: collects data from interactions
- Algorithmic model layer: Transformer layers are taken from pre-trained LLM (e.g., Llama, Qwen, DeepSeek, etc.), use GPT-2
  - Shows that these general models trained on vast amount of language contain knowledge that can be useful also for other tasks
- Functional application layer: acts on the network



"ChannelGPT: A Large Model toward Real-World Channel Foundation Model for 6G Environment Intelligence Communication", Yu et al.

Dataset

- Constructed by making a simulation (a) of a real world scenario (b)
  - outdoor urban scenario, 200 m long and 200 m wide.
  - Includes four building groups and four roads, with various types of vehicles randomly placed on the roads to ensure sufficient diversity.
  - The transmitter (Tx) is placed on the rooftop of a building in the center of the scenario
  - a series of receivers (Rxs) are placed in the middle of the street for RT simulations.
  - The mobile user's trajectory is predefined to generate the CSI time series dataset, including 6,293 samples split into 70% training, 10% validation, and 20% testing sets.

(a)

(b)

MEDIATEK

"ChannelGPT: A Large Model toward Real-World Channel Foundation Model for 6G Environment Intelligence Communication", Yu et al.

Evaluation

- Channel prediction
  - (top) ChannelGPT is more accuracurate on predictions further in the future
  - (C) ChannelGPT has faster convergence when training

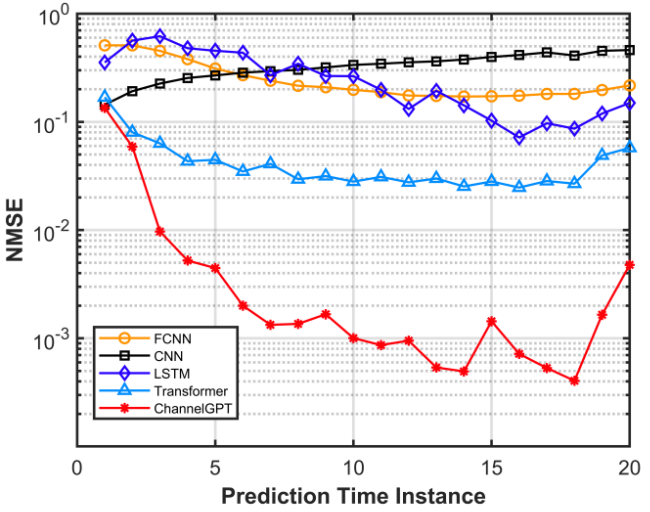
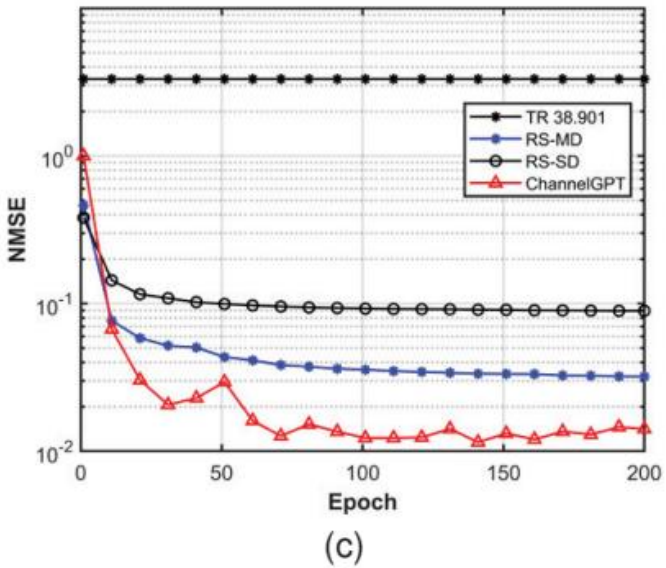
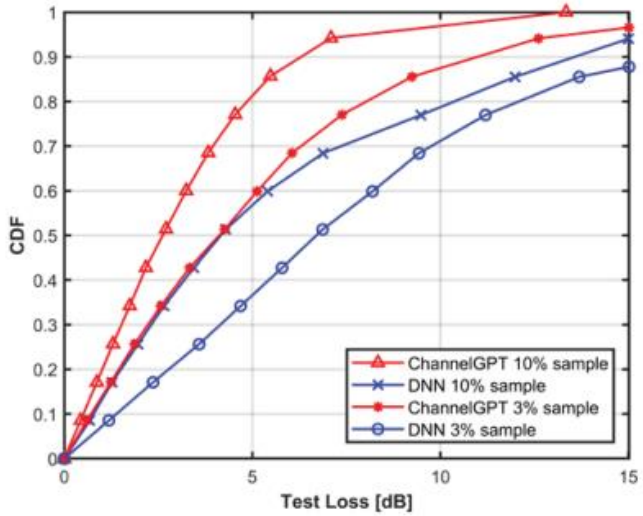


Fig. 3. The NMSE performance of ChannelGPT and other baselines versus different time instance for long-term channel prediction.

- Few shot path loss prediction (d)
  - ChannelGPT performs better than a baseline NN and the difference is more pronounced for smaller training budgets
    - Confirms that Foundation Models enable downstream tasks with less data



(c)

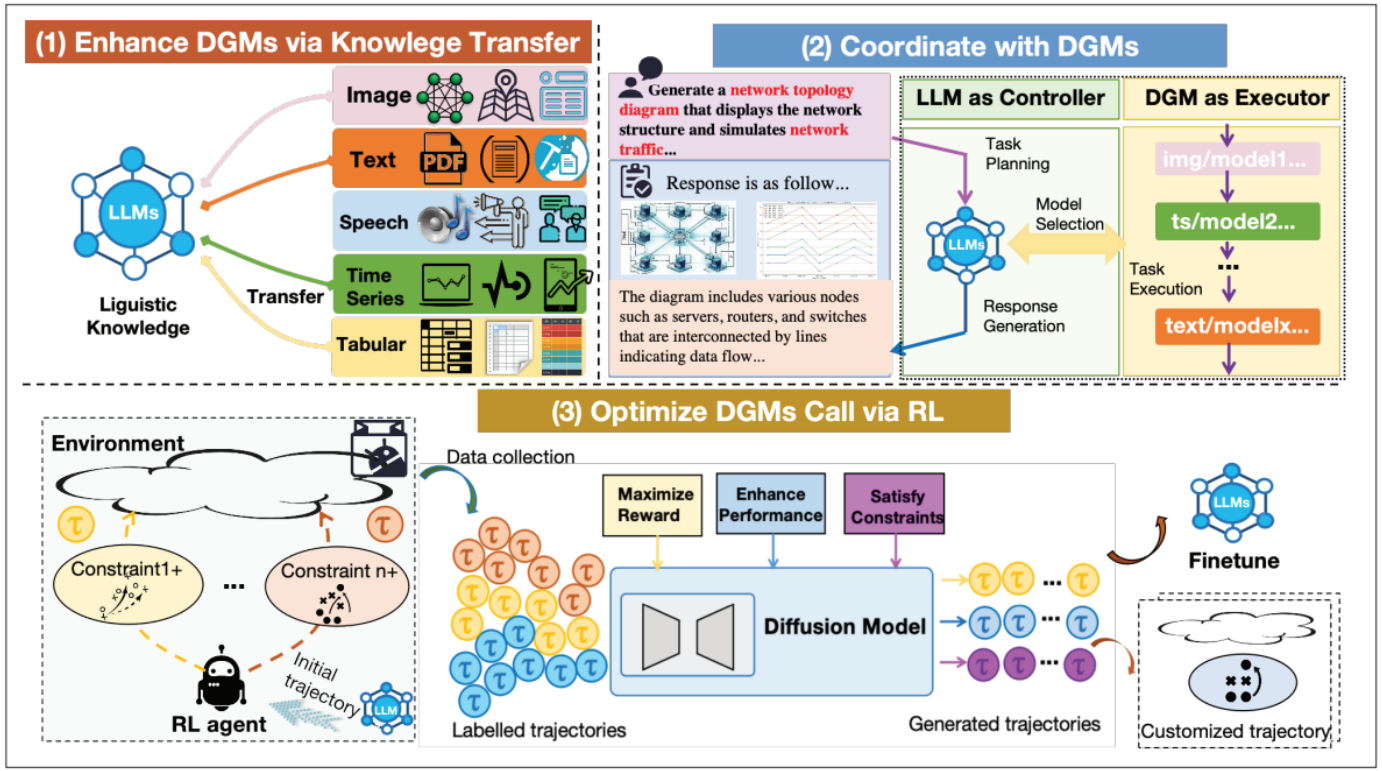


(d)

MEDIATEK

"LLM-Enabled Multi-Modal Data Synthesis via Cross-Domain Collaboration", Zhou et al.

- (Good) Data is hard to find
  - scarcity, privacy issues, high cost of acquisition
- Existing data generation models lack generalizability
- Propose a large language models (LLMs) driven framework
  - Combines LLMs and domain-specific generative models (DGMs)
  - LLMs act as the core to interpret user requests, decompose a complex task into a manageable set of sub-tasks, and delegate each sub-task to the most suitable DGM



"LLM-Enabled Multi-Modal Data Synthesis via Cross-Domain Collaboration", Zhou et al.

Generation Pipeline

- Domain specific tokenizers
- LLM interacts with domain specific generative models
- The refinement process introduces a learning mechanism to enhance the LLMs' proficiency in selecting the most suitable DGMs and maximizing their utility.
  - Structured as an offline reinforcement learning task, with the objective of deriving optimal policies from collected offline data, thus avoiding direct interaction with the live environment.

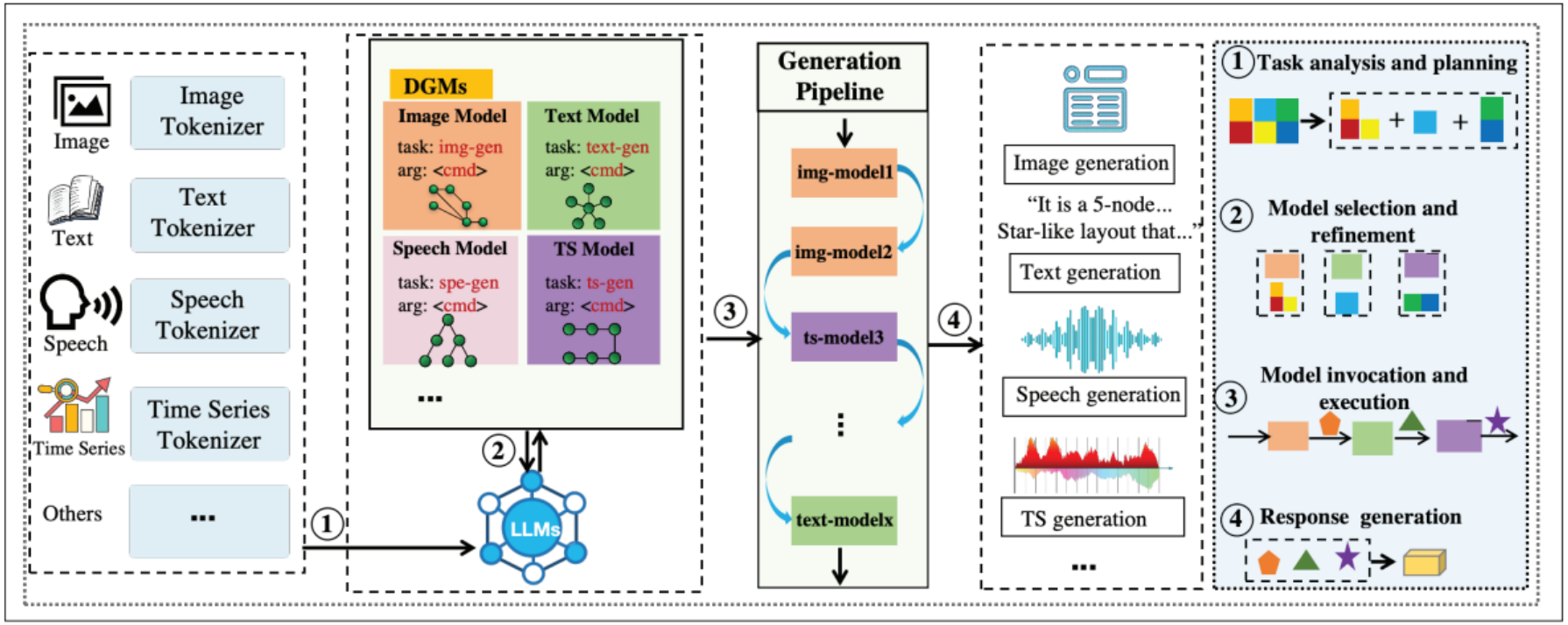


FIGURE 2. he proposed LLM-enabled framework for general multi-modal data synthesis. For the task "Develop a 5G network simulation with time-based performance metrics and troubleshooting logs," the LLM will deduce and deploy a "Time-Series Generation Model" to produce data like signal strength, latency, and packet loss at different time points. Concurrently, it will also deduce and deploy a "Text Generation Model" for creating textual logs.

"LLM-Enabled Multi-Modal Data Synthesis via Cross-Domain Collaboration", Zhou et al.

Evaluation

- Table 1 show that the generated time series sequences achieve excellent performance in conforming to abstract features, for example, trend and periodicity.
- Fig. 5a illustrate that enhanced model descriptions can marginally improve the success rate. The refinement process consistently yields a high success rate
- Figure 6 benchmarks against state-of-the-art data synthesis models, assessing both generation quality and computational efficiency.

Model	Fidelity				Usefulness		
	TP(↑)	FID (↓)	IS (↑)	DS (↓)	RR	RS	SR
BS1	0.71	4.81	3.89	0.58	0.80	0.71	0.60
BS2	0.83	5.23	3.88	0.51	0.81	**0.75**	0.58
Prop	**0.92**	**3.42**	**5.53**	**0.36**	0.79	0.74	**0.76**

TABLE 1. Quantitative comparison with the baselines in terms of Fidelity and Usefulness. BS1 is trained solely on time series data without the integration of the LLM, while BS2 is trained with the LLM's text descriptions as supplementary inputs but without employing the DM-based knowledge transfer.

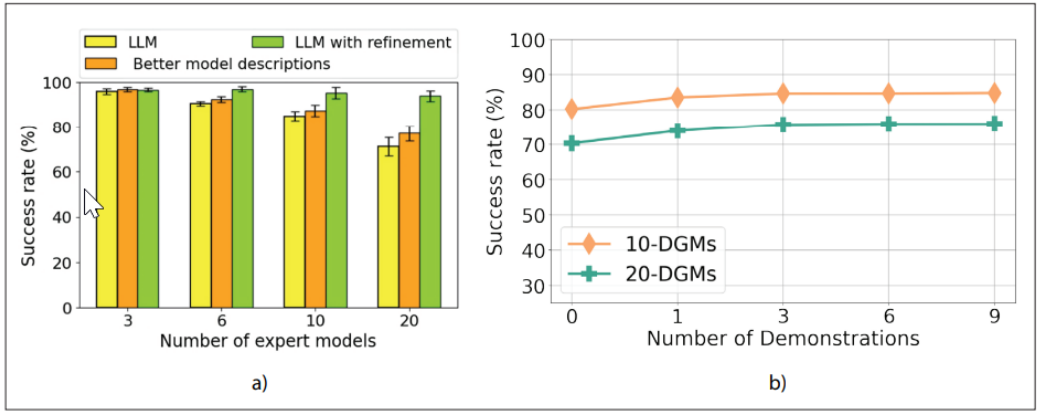


FIGURE 5. Comparison of the success rate for selecting appropriate DGMs in different data generation tasks: a) Performance in relation to the number of DGMs; b) Effect of different number of demonstrations.

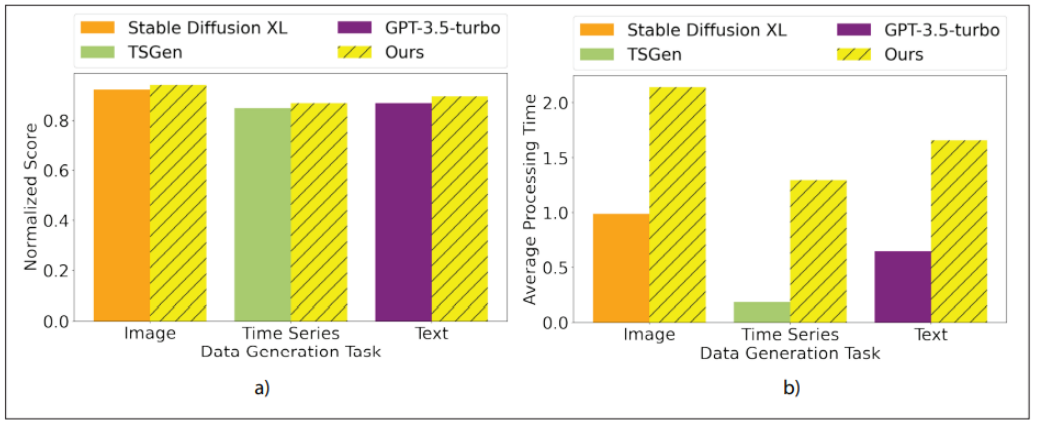


FIGURE 6. Performance comparison in different generation tasks: a) Data quality; b) Processing efficiency.

Conclusions

Observations

- Foundation models in communication systems are gaining increasingly more attention
- Positive results in limited applications
- Integration with natural language models is being explored
- Most papers use ad-hoc custom made datasets

What is missing?

- Definition of standard benchmark downstream tasks and datasets
  - Definition of standard baselines (e.g., traditional algorithms)
- Work on making these models actually usable in practice
  - How to integrate into existing systems
  - How to deal with low-latency/memory/etc. constraints
- Lack of (large scale) real-world datasets
- Scaling to SOTA LLM sizes

Part 6

Future Directions and Open Challenges

- Large datasets
- Standardized benchmarks
- On-device inference optimization
- Federated learning
- AI-native air interface

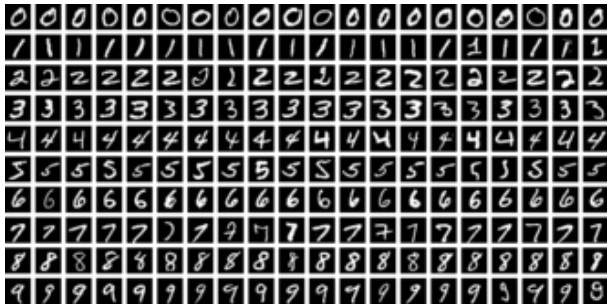
Related Work

- “The Role of Federated Learning in a Wireless World with Foundation Models”, Zihan Chen; Howard H. Yang; Y. C. Tay; Kai Fong Ernest Chong; Tony Q. S. Quek, IEEE Wireless Communications (2024)
- “The Convergence of Artificial Intelligence Foundation Models and 6G Wireless Communication Networks”, Mohamed R. Shoaib; Zefan Wang; Jun Zhao, IEEE 99th Vehicular Technology Conference (2024)
- “Large Generative AI Models for Telecom: The Next Big Thing?”, Lina Bariah, Qiyang Zhao, Hang Zou, Yu Tian, Faouzi Bader, and Merouane Debbah, IEEE Communications Magazine (Early Access) (2024)
- “Big AI Models for 6G Wireless Networks: Opportunities, Challenges, and Research Directions”, Zirui Chen; Zhaoyang Zhang; Zhaohui Yang, IEEE Wireless Communications (2024)
- “Implementation of Big AI Models for Wireless Networks with Collaborative Edge Computing”, Liekang Zeng, Shengyuan Ye, Xu Chen, and Yang Yang, IEEE Wireless Communications (2024)
- “Device-Edge Cooperative Fine-Tuning of Foundation Models as a 6G Service”, Hai Wu, Xu Chen, and Kaibin Huang, IEEE Wireless Communications (2024)
- “Semantic Communications Using Foundation Models: Design Approaches and Open Issues”, Peiwen Jiang; Chao-Kai Wen; Xinping Yi; Xiao Li; Shi Jin; Jun Zhang, IEEE Wireless Communications (2024)
- “Edge Large AI Models: Revolutionizing 6G Networks”, Zixin Wang, Yuanming Shi, Yong Zhou, Jingyang Zhu, Khaled B. Letaief, IEEE Communications Magazine (2025)

Lack of Large Datasets & Standardized Benchmarks

- Benchmarking & Comparability
  - Allow researchers to compare algorithms fairly.
- Accelerating Research & Innovation
  - Reduce duplication of effort in data collection.
  - Researchers can focus on model development rather than data gathering.
- Reproducibility & Transparency
  - Make experiments easier to replicate.
  - Ensures scientific rigor and trust in results.
- Driving Breakthroughs
  - Landmark advances (e.g., deep learning in vision) were enabled by large, well-labelled datasets.
  - ImageNet Challenge was pivotal in the rise of convolutional neural networks.
- Community Collaboration
  - Public datasets foster collaboration across academia, industry, and open-source communities.

MNIST (1998)



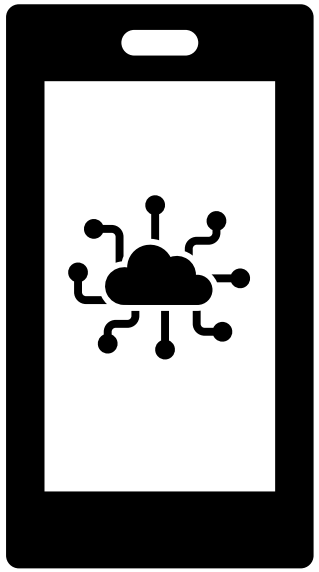
ImageNet (2012)

Laion (2021)

Optimization for On-Device Inference

Why On-Device?

- Real-time decision-making (e.g., adaptive modulation, interference mitigation).
- Reduced latency compared to cloud inference.
- Privacy: sensitive communication data stays on the device.
- Resilience: works even with limited or no connectivity to the core network.



Challenges:

- Foundation models are huge (hundreds of billions of parameters).
- Limited compute, memory, and energy on edge devices (UEs, IoT nodes, base stations).
- Need for efficient adaptation without retraining the full model.

Research Needs:

- Model compression: pruning, quantization, distillation.
- Parameter-efficient fine-tuning: LoRA, adapters, prefix tuning.
- Architecture optimization: lightweight transformer variants, sparse attention.
- Hardware-aware design: co-optimizing model and device hardware.

Federated Learning

- Distributed training where data stays local (e.g., at base stations, devices). Only model updates (gradients) are shared with a central server.

FL for Foundation Models in Communication Systems

- Communication data is often sensitive (user traffic, location, network logs).
- Data is naturally distributed across network nodes.
- FL allows pretraining or fine-tuning a large foundation model without centralizing raw data.

Example Workflow:

- Step 1: Start with a general-purpose foundation model pretrained on public/synthetic data.
- Step 2: Use FL to adapt it to real network conditions using local datasets at multiple sites.
- Step 3: Aggregate updates to improve the global model.

Challenges:

- Communication overhead for large models.
- Heterogeneous data distributions (non-IID).
- Synchronization and model convergence.

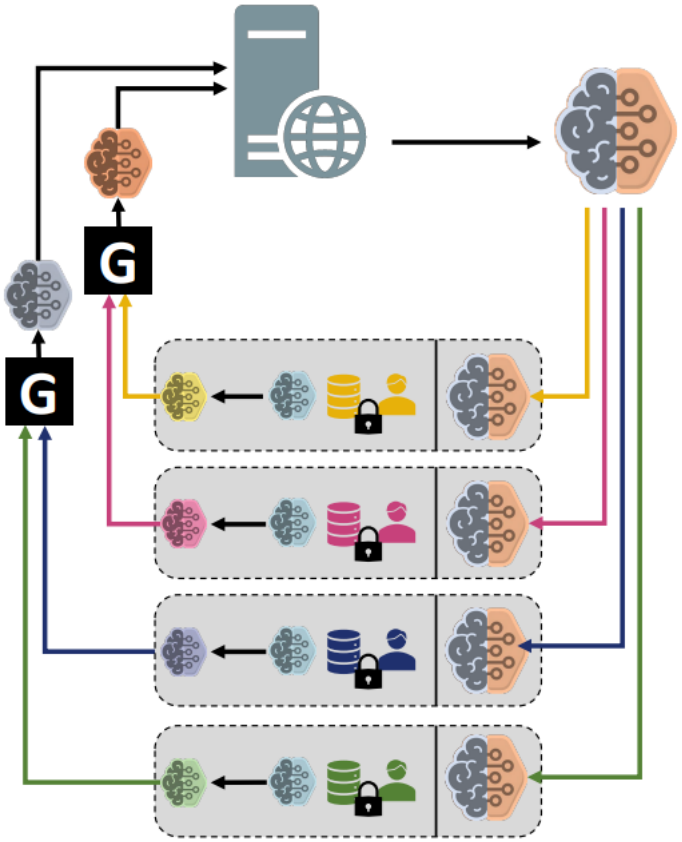


Image from “Federated Learning: A Cutting-Edge Survey of the Latest Advancements and Applications”, Akhtarshenas et al.

AI Native Air Interface

- An air interface designed from the ground up to integrate AI into its operation.

Why Now?

- Traditional air interfaces are designed with fixed algorithms and parameters.
- Increasing complexity of channels (e.g., mmWave, THz, RIS-assisted links) makes static designs suboptimal.
- AI can adapt in real time to changing conditions

Key Features:

- Learning-based waveform design: optimize modulation/coding jointly with channel conditions.
- Adaptive resource allocation: AI predicts traffic and interference patterns.
- Joint sensing and communication: AI integrates environmental sensing into PHY decisions.
- End-to-end optimization: from bits to antennas using deep learning.

Role of Foundation Models:

- Pretrained on massive synthetic + real channel data.
- Adaptable to multiple PHY tasks via fine-tuning or prompting.
- Could unify channel estimation, detection, and optimization in one model.

This could drastically change how algorithms are designed

A very open research direction!

Material

Link:
https://github.com/mtkresearch/foundation_models_comms_Globecom2025

Q&A and Discussion

(20 minutes)

MediaTek Proprietary and Confidential


MEDIATEK