

# Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Data Representation

**Tarun Mangla**

# ML for Networks

Module 1: Case studies of specific network learning tasks

Module 2: Task-agnostic automatic ML pipelines for networks

- Generalized data representation
- Generalized ML model(s)

Module 3: Beyond feature engineering and modeling

# Motivation

- **Current approach for ML in Networking:** Feature engineering to extract the most useful features
  - Significant effort per problem
  - Different features for different learning problems

- This is needed when:
  - Models are not expressive enough <sup>Deep learning</sup>  $\rightarrow$  False
  - Limited computation  $\rightarrow$  False
  - Limited data  $\rightarrow$  False <sup>What has changed?</sup>

$\rightarrow$  Generalized data representation

$\downarrow$   
Not necessarily true for labelled data

Can we develop standard representations of network data?

# Problem statement

- Given networking data, I want to construct standard data representation that is independent of the underlying problem?
- How would you approach this problem?

→ Packet sizes, IAT, Throughput, direction, protocol, (pps, bps) → Flow-level

① What is the underlying data?

→ Packet-level data

↓

② Packet is the fundamental unit

↓

Think about how to featureize it?

③ Are there common feature-sets  
are useful across many tasks?

# Case studies

- Flow-level representation -- [netML](#)
- Packet-level representation -- [nPrint](#)

# Flow-level Representation


- Extract flow summaries from packet traces [ netFlow/ IPFIX)
- netML: A Python-based framework to collect flow summaries

Which features could be extracted?

# Statistical Features (STAT)

*Application classification / NetMicroscope / Ketsune (Security)*

- flow duration,
- number of packets sent per second,
- number of bytes per second,
- various statistics on packet sizes within each flow: mean, standard deviation, inter-quartile range, minimum, and maximum, *skew, kurtosis*
- *# of modes*



## Other Features

- SIZES: A flow is represented as a timeseries of packet sizes in bytes, with one sample per packet.
- IAT: A flow is represented as a timeseries of inter-arrival times between packets, i.e., elapsed time in seconds between any two packets in the flow.
- SAMP-NUM: A flow is partitioned into small intervals of equal length  $\delta t$ , and the number of packets in each interval is recorded; thus, a flow is represented as a timeseries of packet counts in small time intervals, with one sample per time interval. Here,  $\delta t$  might be viewed as a choice of sampling rate for the timeseries, hence the nomenclature.
- SAMP-SIZE: A flow is partitioned into time intervals of equal length  $\delta t$ , and the total packet size (i.e., byte count) in each interval is recorded; thus, a flow is represented as a timeseries of byte counts in small time intervals, with one sample per time interval.

How might each of these features be useful?



# Use of Features in Different Anomaly Detection Approaches

packet size /  
9

Reference	Duration	IAT	SIZE	FFT	SAMP-NUM	SAMP-SIZE
<b>Network Intrusion Detection</b>						
Lakshari <i>et al.</i> [16]	✓	✓	✓	✗	✗	✗
Mukar <i>et al.</i> [12]	✗	✓	✗	✗	✗	✗
Mirsky <i>et al.</i> [21]	✗	✗	✗	✗	✓	✓
<b>IoT Anomaly Detection</b>						
Al Jawarneh <i>et al.</i> [3]	✓	✗	✓	✗	✗	✗
Doshi <i>et al.</i> [6]	✗	✓	✓	✗	✗	✗
Meidan <i>et al.</i> [20]	✗	✓	✗	✗	✓	✓
Bhatia <i>et al.</i> [4]	✗	✗	✓	✗	✓	✗
Thamilarasu <i>et al.</i> [33]	✗	✗	✓	✗	✗	✗
<b>DDoS Detection</b>						
Fouladi <i>et al.</i> [7]	✗	✗	✗	✓	✗	✗
Lima <i>et al.</i> [18]	✗	✗	✓	✗	✗	✗

# Empirical Results

Detector	Dataset	STATS	SIZE	IAT	SAMP- NUM
<u>OCSVM</u>	UNB(PC1)	0.48	0.77	0.81	0.76
	UNB(PC4)	0.55	0.73	0.86	0.75
	CTU	0.64	0.77	0.76	0.91
	MAWI	0.88	0.47	0.45	0.59
	TV&RT	1.00	1.00	0.96	0.93
	SFrig	0.98	0.89	0.82	0.96
	BSTch	0.95	0.98	0.97	0.95

*All or more  
than one*

# Discussion: What are the limitations

- more than one set of feature could be used
  - there could be other set of features
    - ↳ TCP flags / port #s
    - ↳ ~~Other~~ protocol details /
    - ↳ Payload information ( RTP headers / TLS SNI / DNS )
  - (does not necessarily generalize across datasets / problems)
  - works at flow-level & not packet-level
- python library

## Advantage

- ① Task-agnostic
- ② Light weight → system cost
- ③ Interpretable
- ④ less prone to overfitting ( Fair bit of distillation )

# This Class

- Flow-level representation
- **Packet-level representation**

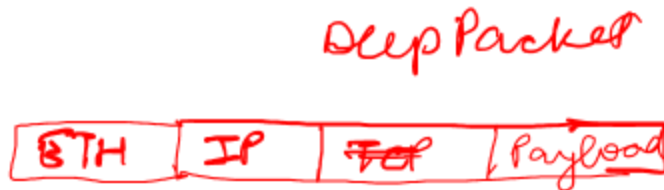
*n-bytes  
is system cost  $\nearrow$  reduced accuracy*

# Why Packet-level Representation?

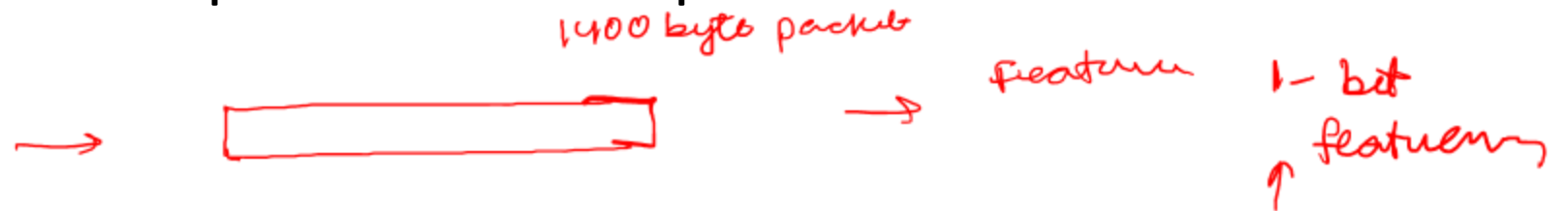
- Motivation: standard representation of image data
- Insight: Packet is the atomic unit
  - Every traffic analysis task can be represented by a group of packets
- What could be a standard representation of packet data?

① Modularization

*20-bytes  
into a single feature  
 $\hookrightarrow$  reduced system cost*



*Data unit*  
*N/w layer*  
*Transport*  
*SRK / DST IP*  
*Removed*



- ① Zero-padding for equal size
- ② Removed non-useful features
- ③ *semantic* Alignment of features
- ④ Byte-sized features

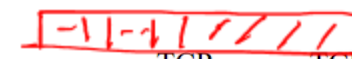
IPV4 & IPV6

- IPv4 | IPv6



40 - byls

IPV (



## IPv6



- |  | IP<br>Bit 1                    | IP Options<br>Bit 1                                | TCP<br>Bit 1 | TCP Options<br>Bit 1 | Payload<br>Bit 1 |
|--|--------------------------------|--|--------------|----------------------|------------------|
| Naive Binary: (TCP / IP)                 | 0 1 0 0 0 1 0 1 ... .. 0 1 1 1 | 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 ... .. 1 1 0 0 | 1 0 1 1      |                      |                  |
| Naive Binary: (TCP / IP)<br>(No Options) | 0 1 0 0 0 1 0 1 ... ..         | 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 ... ..         | 1 0 1 1      |                      |                  |
| Naive Binary: (UDP / IP)                 | 0 1 0 0 0 1 0 1 ... ..         | 1 0 0 1 0 0 1 0 ... ..                             | 1 0 1 1      |                      |                  |

Seq #s → ( )

Port #s

# Problems with Semantic Representation

- Semi-structured fields: TCP options

(Name, value pairs)



- Domain expertise: how to semantically represent each protocol?

- Normalization: expensive engineering

- Payload: not clear how to represent

# Problems with naïve binary representation

- Mis-alignment of headers
- Interpretability



# nPrint

IPv4 480 Features	TCP 480 Features	UDP 64 Features	ICMP 64 Features	Payload <i>n</i> Features
Maximum Size of IPv4 Header (60 Bytes)	Maximum Size of TCP Header (60 Bytes)	Size of UDP Header (8 Bytes)	Size of ICMP Header (8 Bytes)	User Defined Number of Bytes

nPrint (TCP / IP) Packet

0	1	0	0	0	1	0	1	1	1	...	...	...	...	0	1	0	1	0	1	0	1	...	...	...	...	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	0	...	...	...	...
---	---	---	---	---	---	---	---	---	---	-----	-----	-----	-----	---	---	---	---	---	---	---	---	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	-----	-----	-----	-----

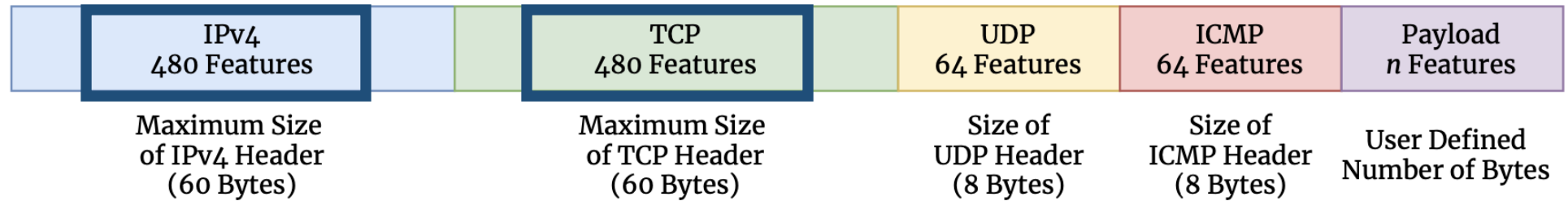
nPrint (UDP / IP) Packet

0	1	0	0	0	1	0	1	1	1	...	...	...	...	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	1	1	...	...	...	...	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	0	...	...	...	...
---	---	---	---	---	---	---	---	---	---	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	-----	-----	-----	-----

# nPrint Hybrid Representation Features

- Complete
- Aligned
- Modular
- Constant size per problem
- Inherently normalized

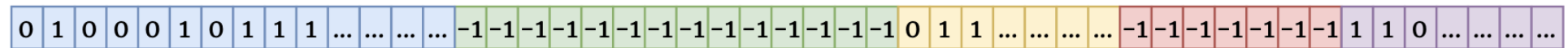
# Complete



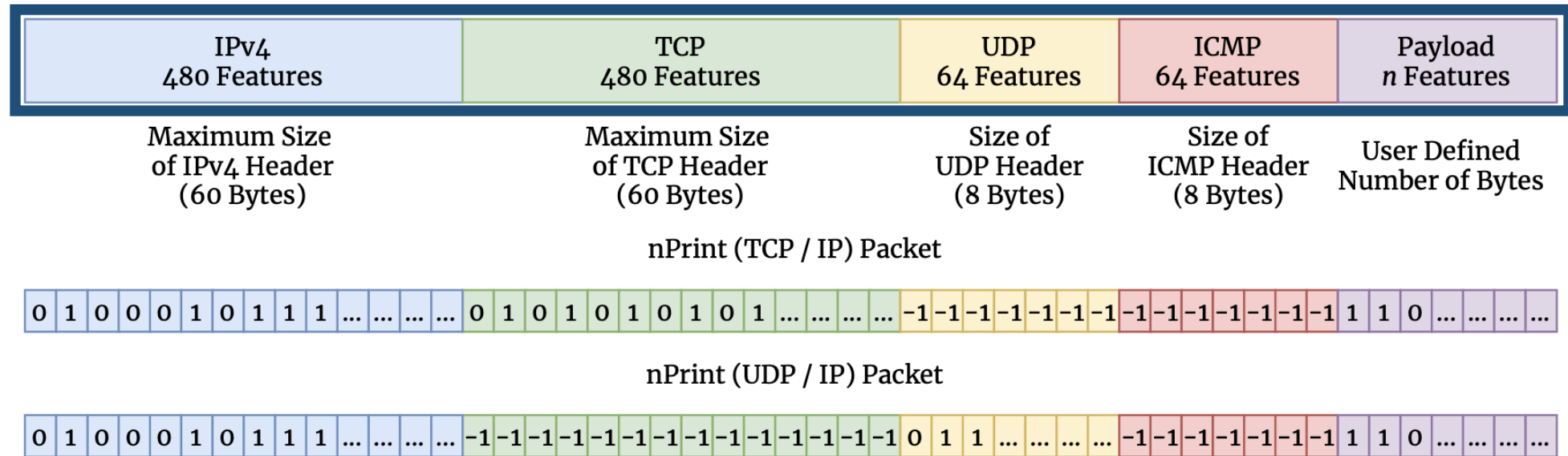
nPrint (TCP / IP) Packet



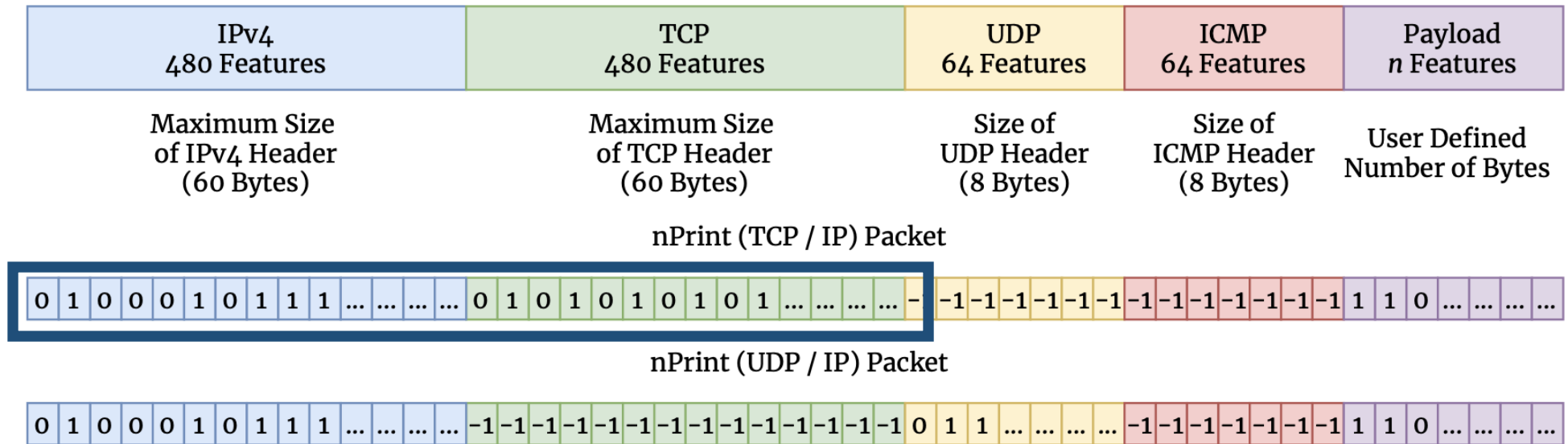
nPrint (UDP / IP) Packet



# Constant Size Per Problem



# Inherently Normalized



# Aligned

IPv4 480 Features	TCP 480 Features	UDP 64 Features	ICMP 64 Features	Payload <i>n</i> Features
Maximum Size of IPv4 Header (60 Bytes)	Maximum Size of TCP Header (60 Bytes)	Size of UDP Header (8 Bytes)	Size of ICMP Header (8 Bytes)	User Defined Number of Bytes

nPrint (TCP / IP) Packet

0	1	0	0	0	1	0	1	1	1	...	...	...	...	0	1	0	1	0	1	0	1	...	...	...	...	-	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	1	0	...	...	...	...
---	---	---	---	---	---	---	---	---	---	-----	-----	-----	-----	---	---	---	---	---	---	---	---	-----	-----	-----	-----	---	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	-----	-----	-----	-----

nPrint (UDP / IP) Packet

0	1	0	0	0	1	0	1	1	1	...	...	...	...	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	1	1	...	...	...	...	-1	-1	-1	-1	-1	-1	-1	-1	1	1	0	...	...	...	...
---	---	---	---	---	---	---	---	---	---	-----	-----	-----	-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	---	---	---	-----	-----	-----	-----	----	----	----	----	----	----	----	----	---	---	---	-----	-----	-----	-----

# Evaluation

- **Task:** Device fingerprinting
- **Baseline:** Nmap – an active measurement tool that does device fingerprinting by sending probes to devices and analyzes their response

Representation	Balanced Accuracy	ROC AUC	F1
nPrint	95.4	99.7	95.5
Nmap	92.7	99.3	92.9

# Evaluation: Other tasks

Problem Overview			nPrintML					Comparison	
Description	Dataset	# Classes	Configuration eAppendix A.4)	Sample Size (# Packets)	Balanced Accuracy	ROC AUC	Macro F1	Score	Source
Active Device Fingerprinting (§5.1)	Network Device Dataset [22]	15	-4 -t -i	21	95.4	99.7	95.5	92.9 (Macro-F1)	ML-Enhanced Nmap [31]
Passive OS Detection (§5.2)	CICIDS 2017 [48]	3	-4 -t	1	99.5	99.9	99.5	81.3 (Macro-F1)	p0f [40]
				10	99.9	100	99.9		
				100	99.9	100	99.9		
		13		100	77.1	97.5	76.9	No Previous Work	
Application Identification via DTLS Handshakes (§5.3)	DTLS Handshakes [32]	7	-4	43	99.8	96.9	99.7	99.8 (Average Accuracy)	Hand-Curated Features [32]
			-u		99.9	99.7	99.5		
			-p 10		95.0	78.8	77.4		
			-p 25		99.9	99.7	99.7		
			-p 100		99.9	99.7	99.7		
			-4 -u -p 10		99.8	99.9	99.8		
Malware Detection for IoT Traces (§5.4.1)	netML IoT [6, 28]	2	-4 -t -u	10	92.4	99.5	93.2	99.9 (True Positive Rate)	NetML Challenge Leaderboard [37]
		19			86.1	96.9	84.1	39.7 (Balanced F1)	
Type of Traffic in Capture (§5.4.1)	netML Non-VPN [6, 12]	7	-4 -t -u -p 10	10	81.9	98.0	79.5	67.3 (Balanced F1)	
					76.1	94.2	75.8	42.1 (Balanced F1)	
		18	-4 -t -u		66.2	91.3	63.7	34.9 (Balanced F1)	
		31			60.9	92.2	57.6		
Intrusion Detection (§5.4.1)	netML CICIDS 2017 [6, 48]	2	-4 -t -u	5	99.9	99.9	99.9	98.9 (True Positive Rate)	
		8			99.9	99.9	99.9	99.2 (Balanced F1)	
Determine Country of Origin for Android & iOS Application Traces (§5.4.2)	Cross Platform [44]	3	-4 -t -u -p 50	25	96.8	90.2	90.4	No Previous Work	
Identify streaming video (DASH) service via device SYN packets (§5.4.3)	Streaming Video Providers [10]	4	-4 -t -u -R	10	77.9	96.0	78.9	No Previous Work	
				25	90.2	98.6	90.4		
				50	98.4	99.9	98.6		



# **Discussion: What are the limitations?**

# Different way of thinking about data representation

- Assume the downstream model is a deep learning model
- Different kinds of data possible
  - Scalar data
  - Timeseries data
  - Graphical data
- How do you feed each of these data (individually or together?) into a neural network?

# Next Class

- Visitor talk: Attendance will be taken

**Title:** Designing Networked Systems for Spatial Intelligence

**Abstract:** Spatial intelligence is set to reshape how we interact with and understand physical environments by enabling real-time immersive experiences. The ability to seamlessly digitize, stream, and render 3D spaces in real-time can transform industries such as entertainment, remote collaboration, healthcare, and supply chain sectors. However, achieving this vision requires overcoming significant technical challenges related to 3D data acquisition, processing, and transmission. In this talk, I will discuss the networking and systems challenges involved in capturing 3D spaces with high fidelity, focusing on live streaming and adaptive transmission strategies. Traditional 3D capture systems often face scalability, bandwidth efficiency, and latency limitations, which hinder their deployment in wide-area settings. I will introduce approaches such as MeshReduce and RenderFusion that address these challenges by leveraging distributed processing and adaptive streaming techniques. MeshReduce employs a decentralized architecture with hierarchical merging and adaptive bitrate control to efficiently process and transmit 3D content, reducing the computational burden on centralized servers and enabling scalable real-time 3D scene capture using edge devices. Complementing this, RenderFusion dynamically balances local and remote rendering to optimize user experience by intelligently selecting objects for rendering based on network conditions and device capabilities. This hybrid approach mitigates latency challenges while maintaining high frame rates and rendering fidelity, making it ideal for resource-constrained XR devices.

**Bio:** Mallesham Dasari is an Assistant Professor at Northeastern University and the Director of the Spatial Intelligence Research Group ([sinrg.org](http://sinrg.org)). He is affiliated with both the Institute for the Wireless Internet of Things and the Department of Electrical and Computer Engineering. His research interests span Augmented and Virtual Reality (AR/VR) systems, computer networks, wireless sensing and communications, and mobile and wearable computing. Mallesham has published extensively in premier conferences, including IEEE VR, ACM SIGCOMM, IEEE INFOCOM, and USENIX NSDI. He served as a program committee member for conferences such as USENIX NSDI, IEEE VR, COMSNETS, ACM IMC, ACM CoNEXT, ACM MM, and ACM MMSys. He has also contributed as a Program Co-Chair for the ACM SIGCOMM Workshop on Emerging Multimedia Systems and ACM MobiCom workshop on Immersive Computing. Before joining Northeastern, Mallesham received a PhD in computer science from Stony Brook University and was a postdoctoral researcher at Carnegie Mellon University. (edited)

# Special Topics: Machine Learning (ML) for Networking

COL867

Holi, 2025

Data Representation

**Tarun Mangla**

*Slides adapted from CMU and MIT's course on Deep Learning*

# Recap

Flow level  
Packet level

①. Using domain knowledge  
↳ con

- Last lecture: Can we go beyond **feature engineering** by creating standard representation of network data?
- Two solutions:
  - Flow-level (NetML)
  - Packet-level (nPrint)
- What after that?

**Question: Can we go beyond basic model engineering?**

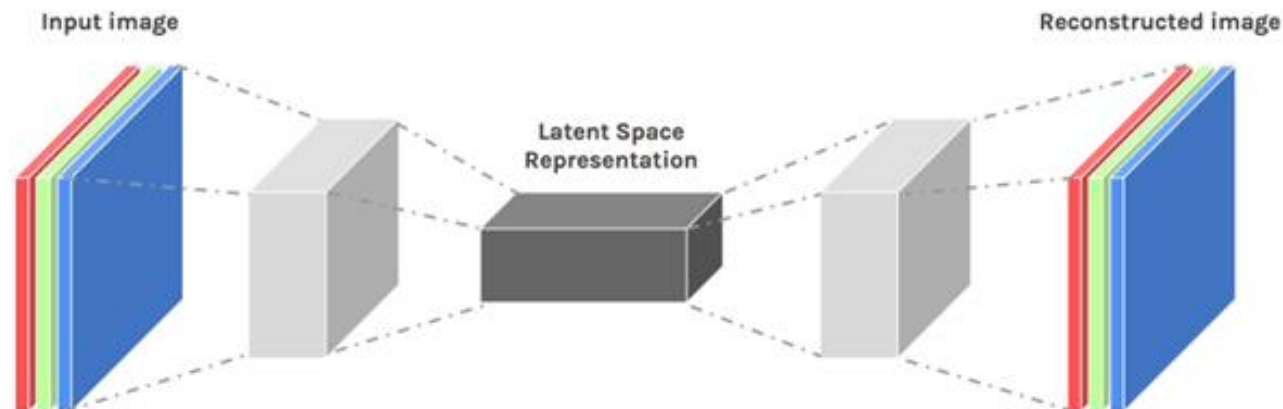
# One model to rule them all

- Build a foundation model for networking data
- What is a foundation model?



# Background on Foundation Models

- Use **self-supervised** learning techniques to learn a meaningful representation of the data
- Can you give an example of a self-supervised learning model?
  - Autoencoders
- Useful for various tasks
  - Generative, discriminative etc.



# Promise of Foundation Models

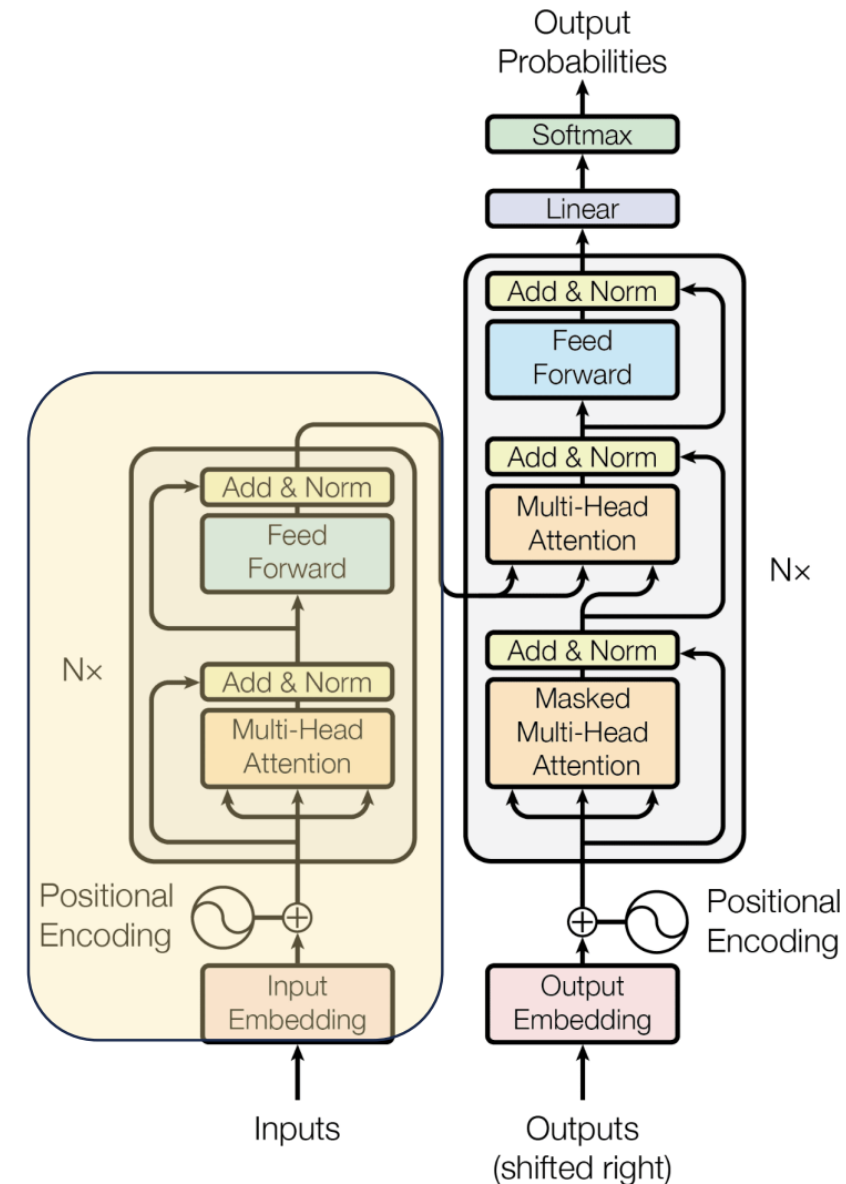
- Foundation models: trained on large corpora of unlabeled data, adapted to different downstream tasks with minimal tuning
- Advantages:
  - Performance improvements
  - Minimal data labeling
  - Reduced manual effort

How do you build a foundation model?



# BERT: Bidirectional Encoder Representation

- Goal: Build effective language representation that can be applied for a variety of downstream tasks
- Pre-trained on a large corpus
  - English Wikipedia – 2500M words
  - Book Corpus – 800M words
- Uses transformer as the underlying model

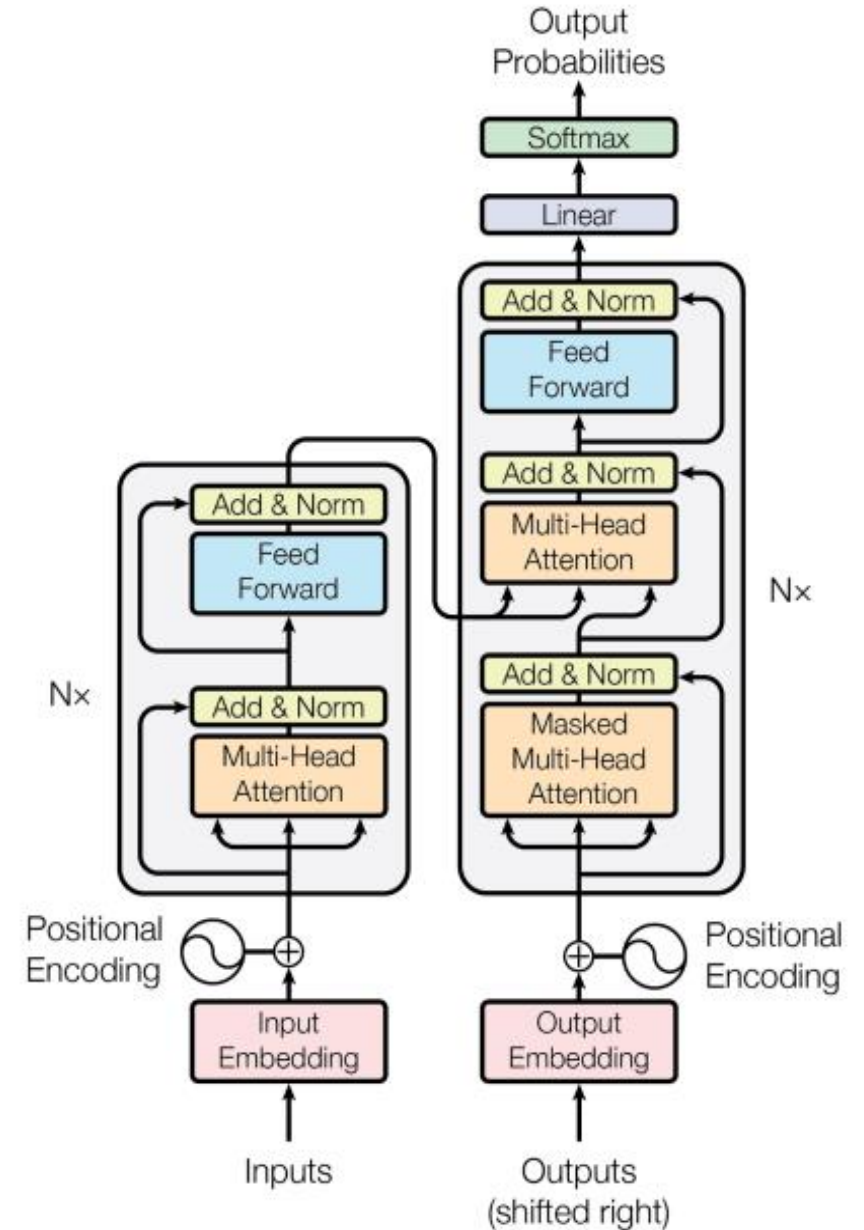


# Transformer Architecture

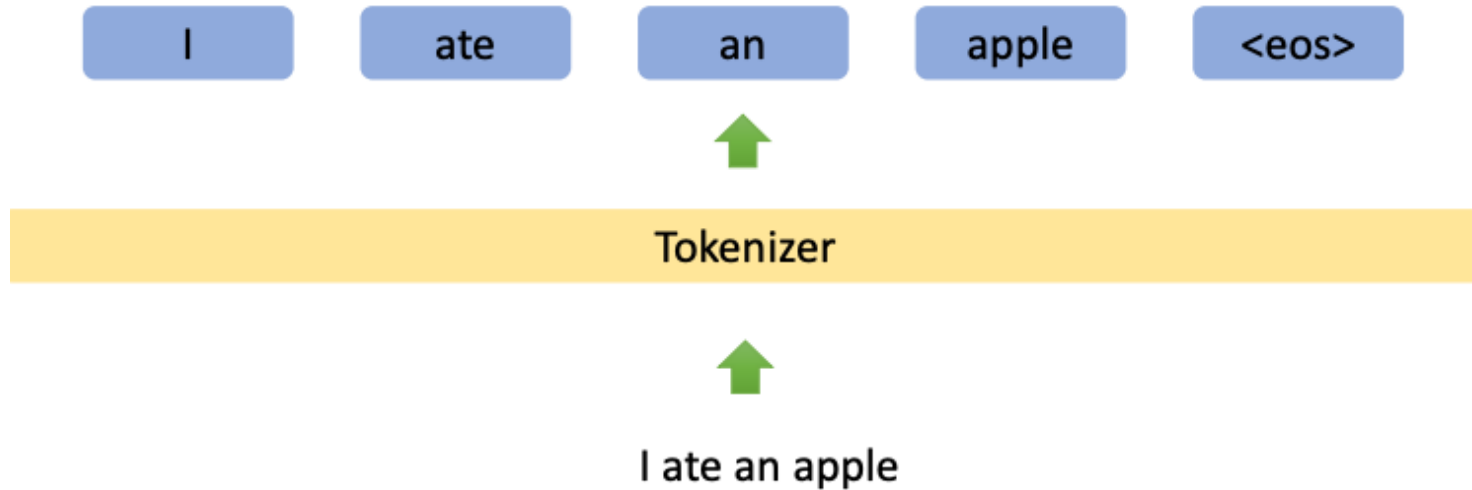
- Tokenization
- Input Embedding
- Positional Encoding
- Residual Connection
- Query
- Key
- Value
- Add & Norm
- Encoder
- Decoder



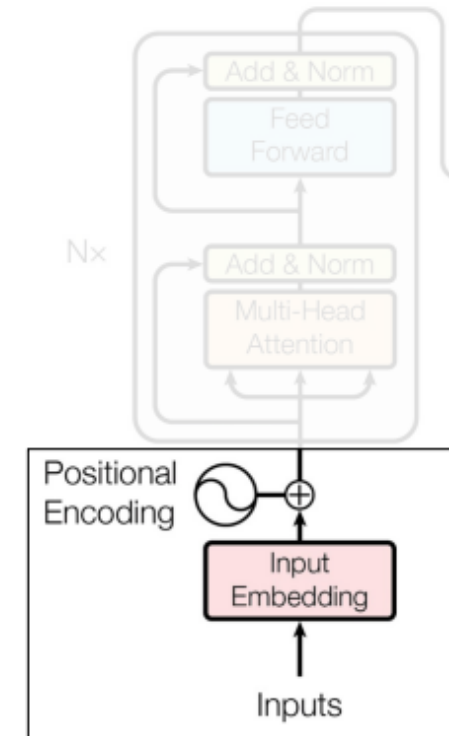
- Attention
- Self-Attention
- Multi-Head Attention
- Encoder Attention
- Probabilities / Logits
- Encoder models
- Decoder only models



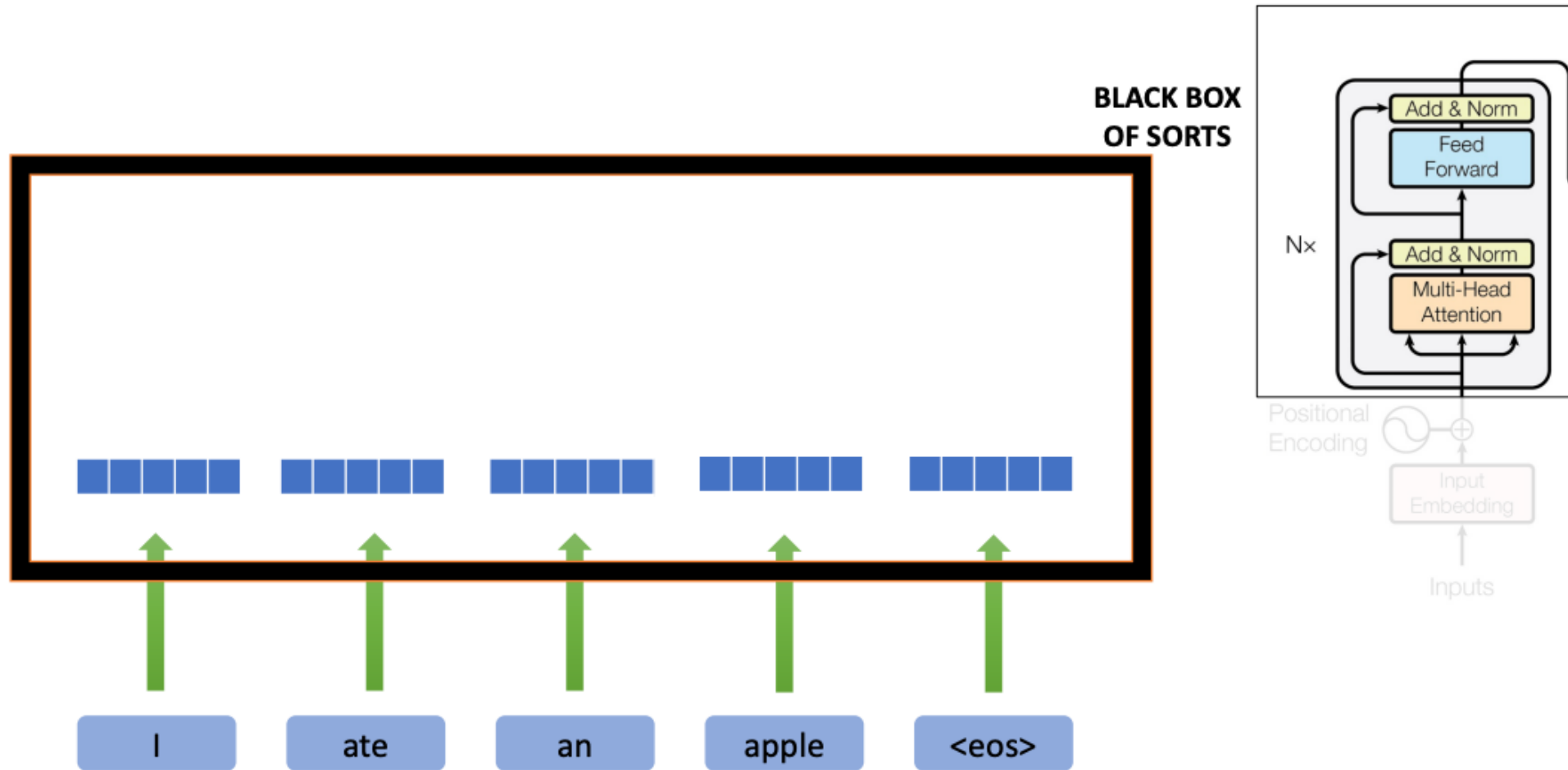
# Processing Input



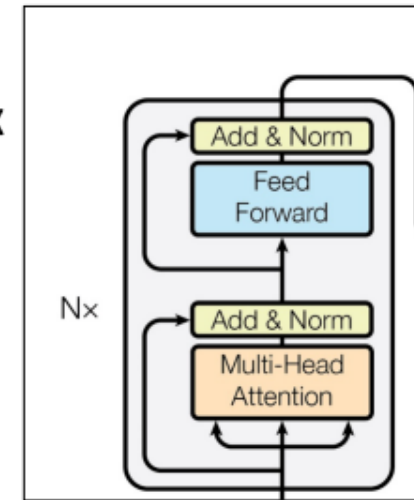
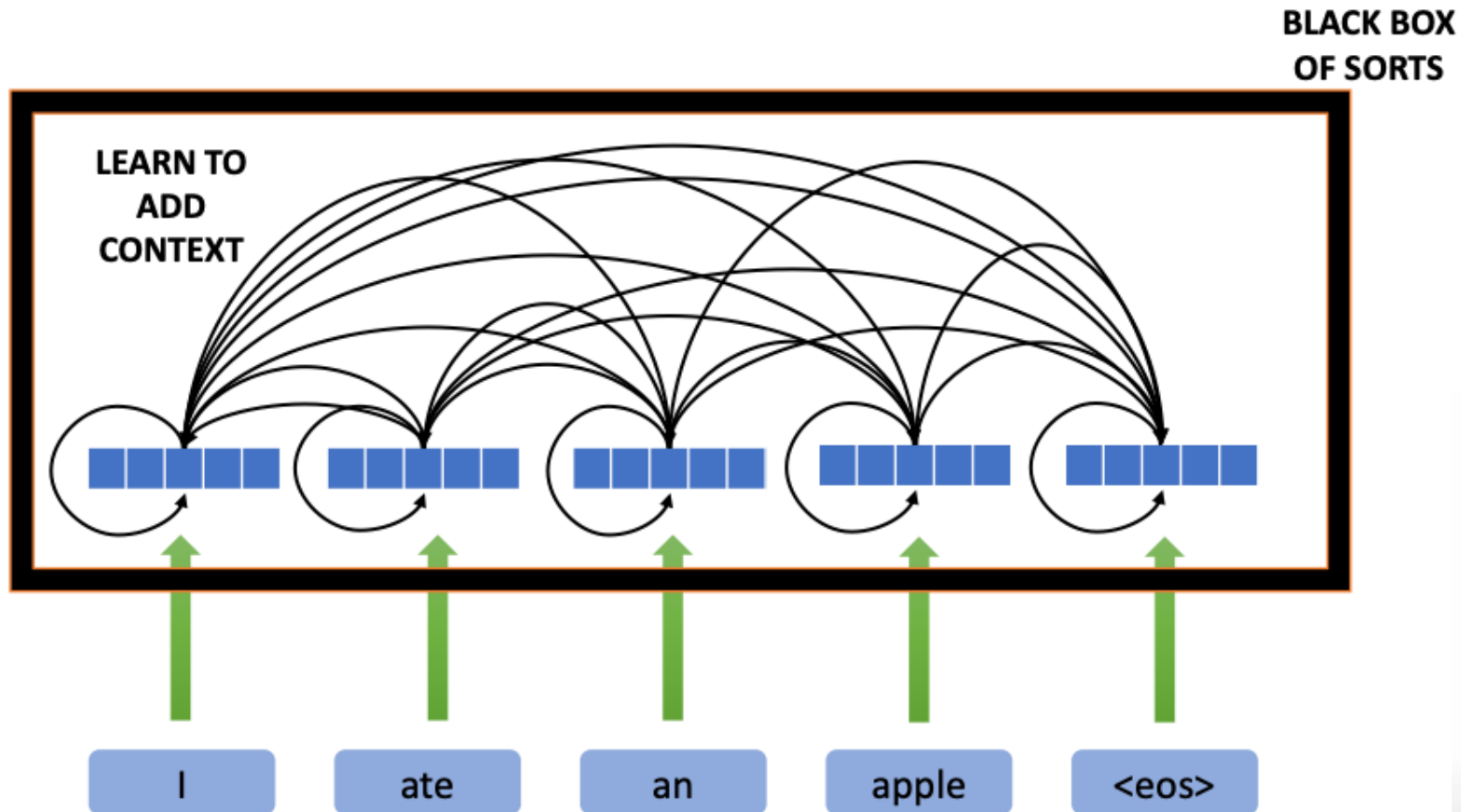
Generate Input Embeddings



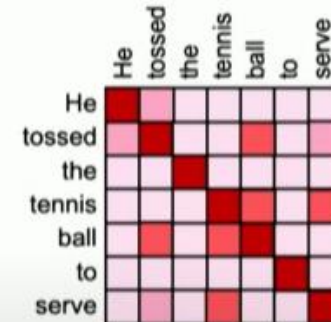
# Capturing Context



# Capturing Context



Attention weighting: where to attend to!  
How similar is the key to the query?

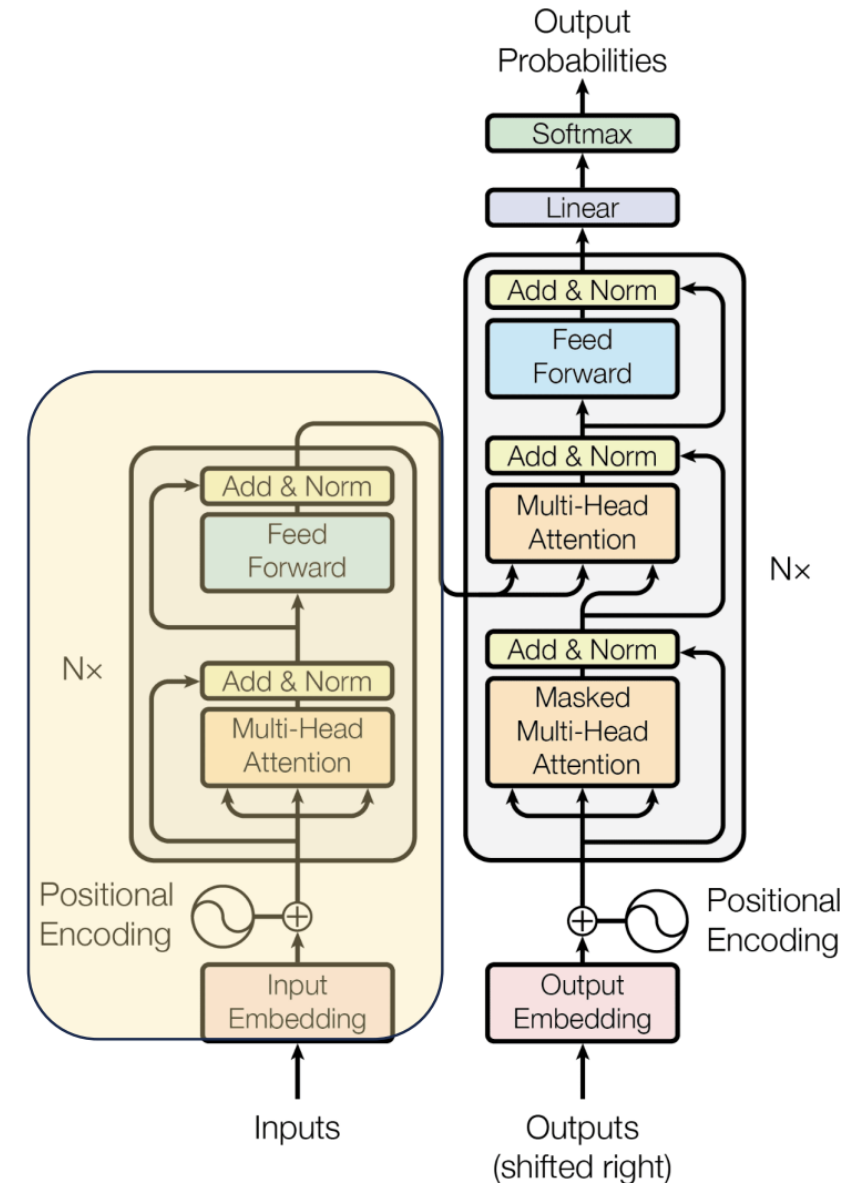
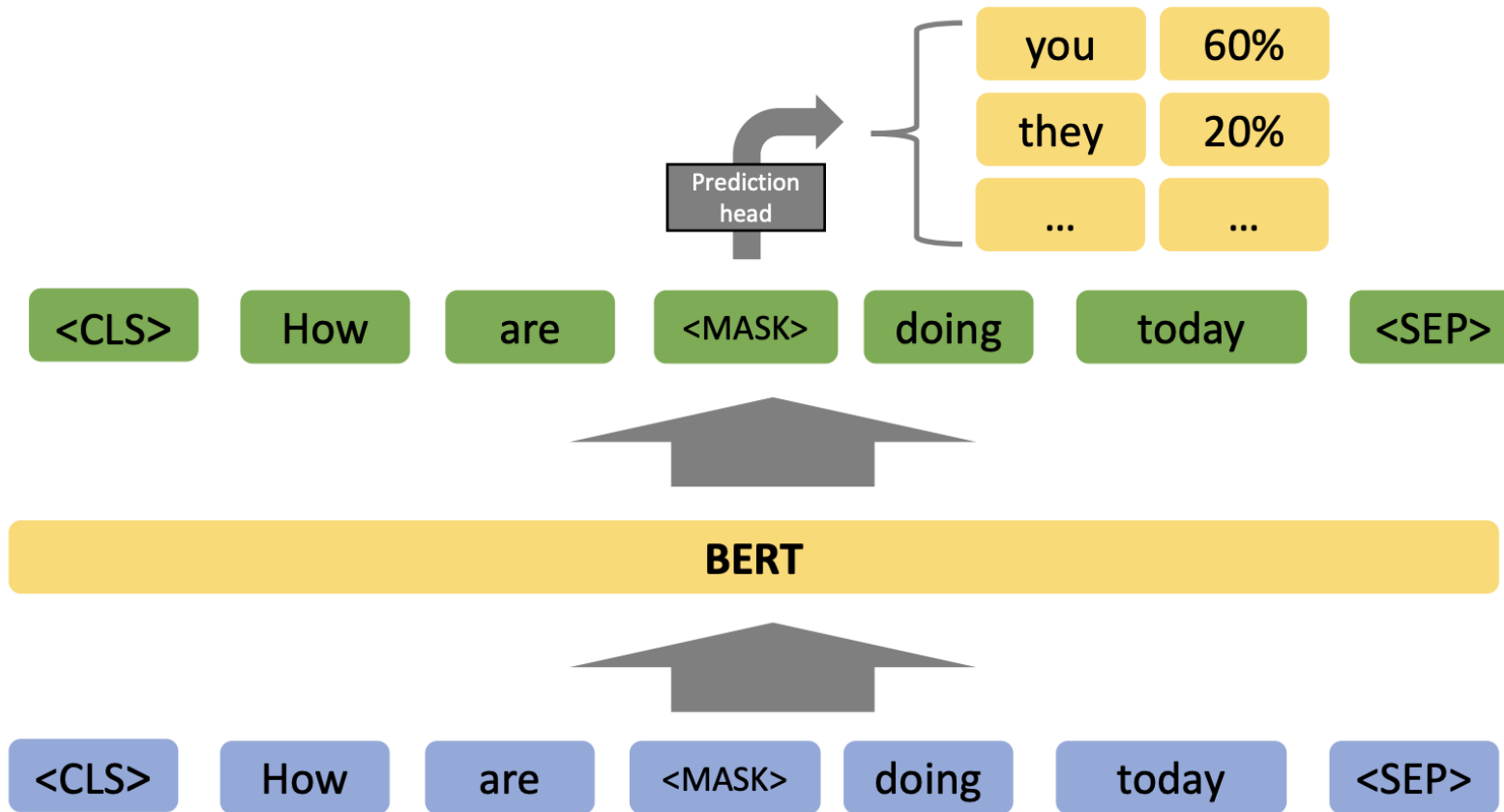


$$\text{softmax} \left( \frac{Q \cdot K^T}{\text{scaling}} \right)$$

Attention weighting

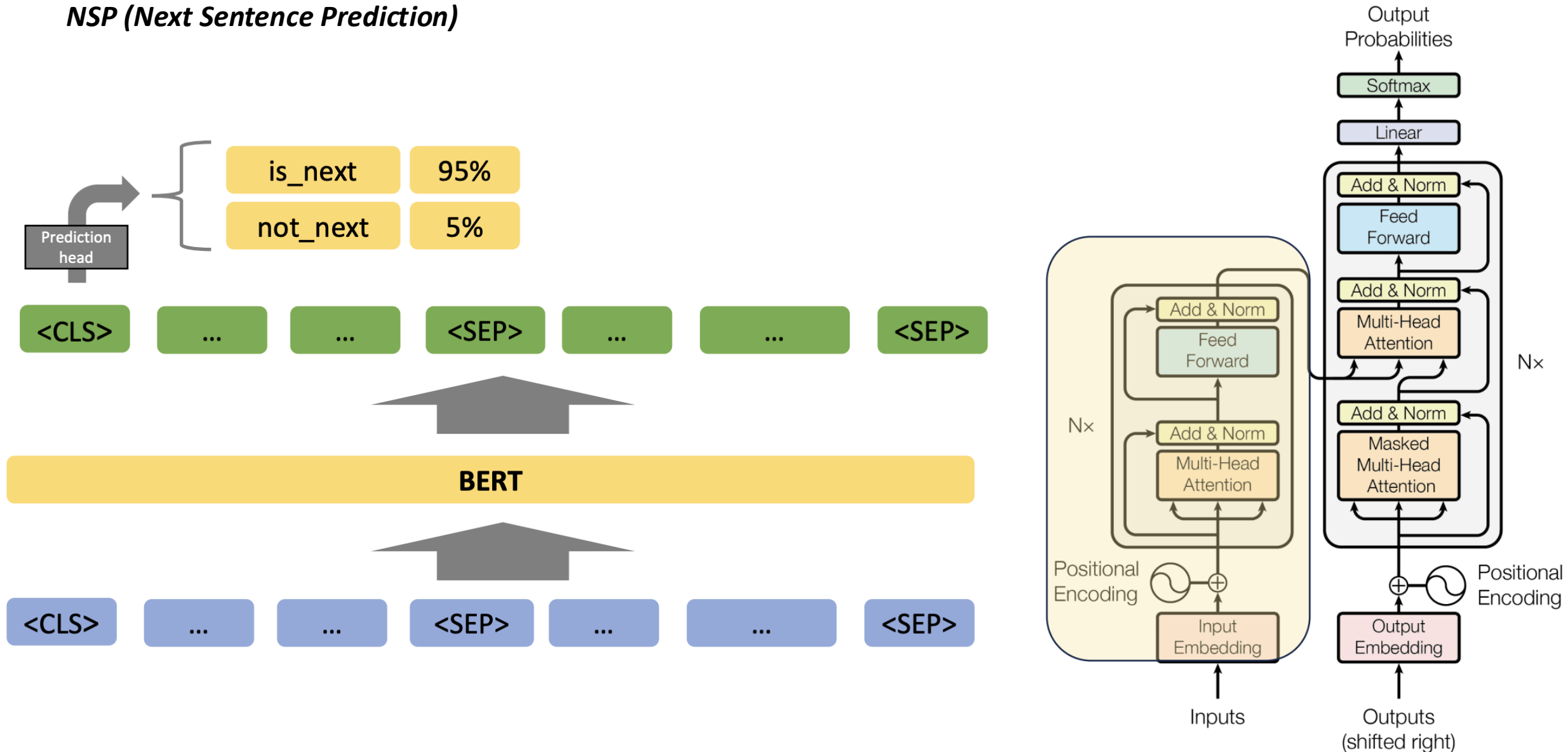
# BERT: Bidirectional Encoder Representation

## MLM (Masked Language Modeling)



# BERT: Bidirectional Encoder Representation

*NSP (Next Sentence Prediction)*



# Why Foundation Models for Networking Data?

- Network learning approaches consist of classification, reinforcement learning, anomaly detection, generative
  - Foundation models have been successfully applied to these problems
- Abundant unlabeled sequential data
  - Campus networks
  - Data center networks
  - Transit/ISP networks
- Rich semantic content (like text)
  - Well-defined protocols



# Challenges

- Tokenizer
  - Text: Tokens can be characters or words
  - What is a token for network data?
- Context
  - How do you define context?
  - What are the pre-training tasks?
- **Post mid-term:** Two papers on design of foundation model for networks
  - netFound
  - netLLM

# Resources

- CMU Deep Learning lecture: <https://deeplearning.cs.cmu.edu/S24/index.html>
- MIT Deep Learning course: <https://introtodeeplearning.com/>
- Explanation with code: <https://nlp.seas.harvard.edu/2018/04/03/attention.html>
- Jay Alammar, The illustrated transformer: <http://jalammar.github.io/illustrated-transformer/>