nPrint: Standard Packet-level Network Traffic Analysis

nprint.github.io/nprint/

Jordan Holland, Paul Schmitt, Nick Feamster, Prateek Mittal

Overview

- Introduce nPrint, a standard data representation for network traffic analysis problems

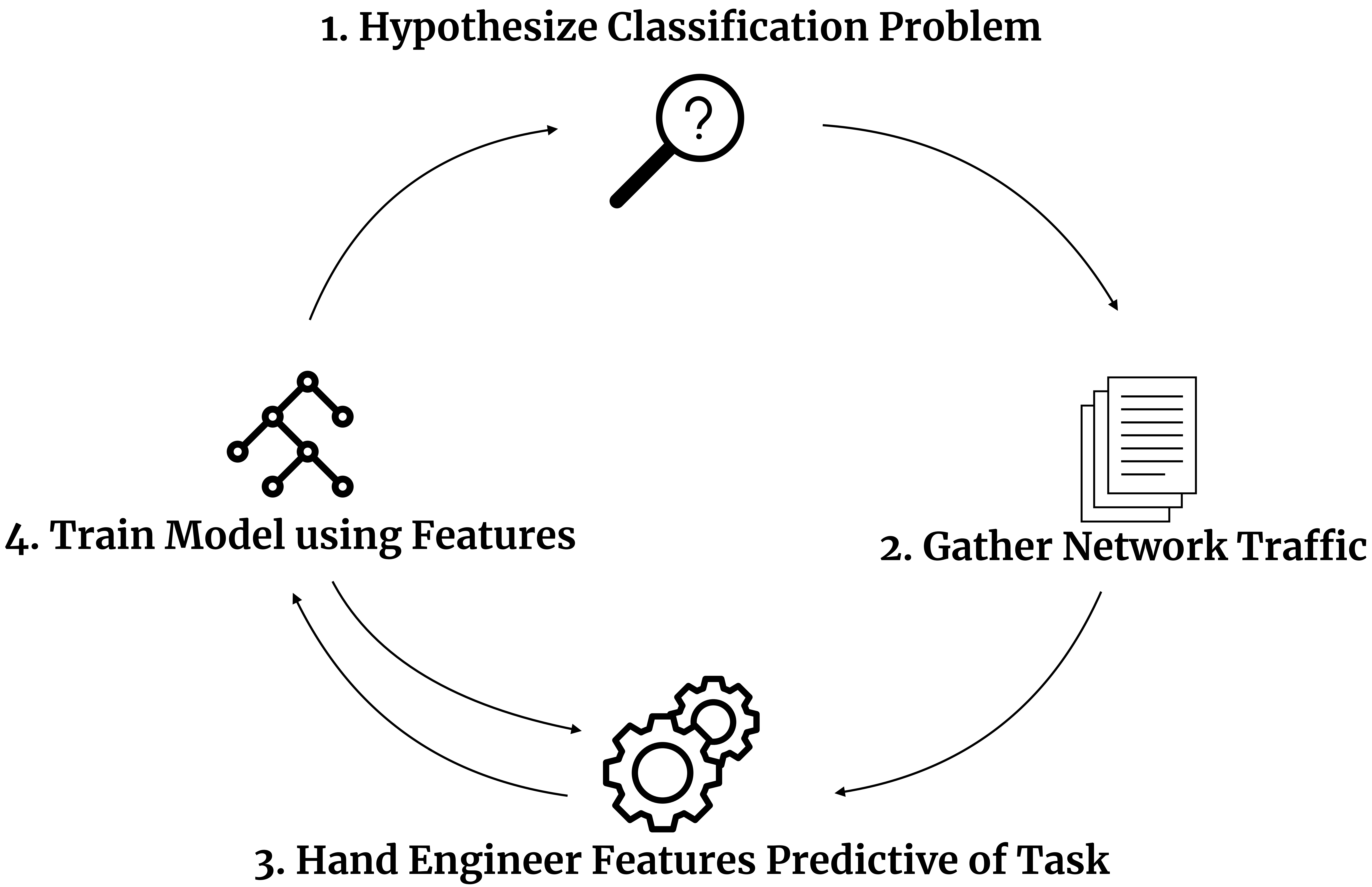
- Remove human driven feature engineering step from the typical machine learning workflow for a class of network traffic classification problems

- Show equivalent or better performance than widely used device and Passive OS Detection tools in passive fingerprinting (p0f), active fingerprinting (Nmap), and application identification

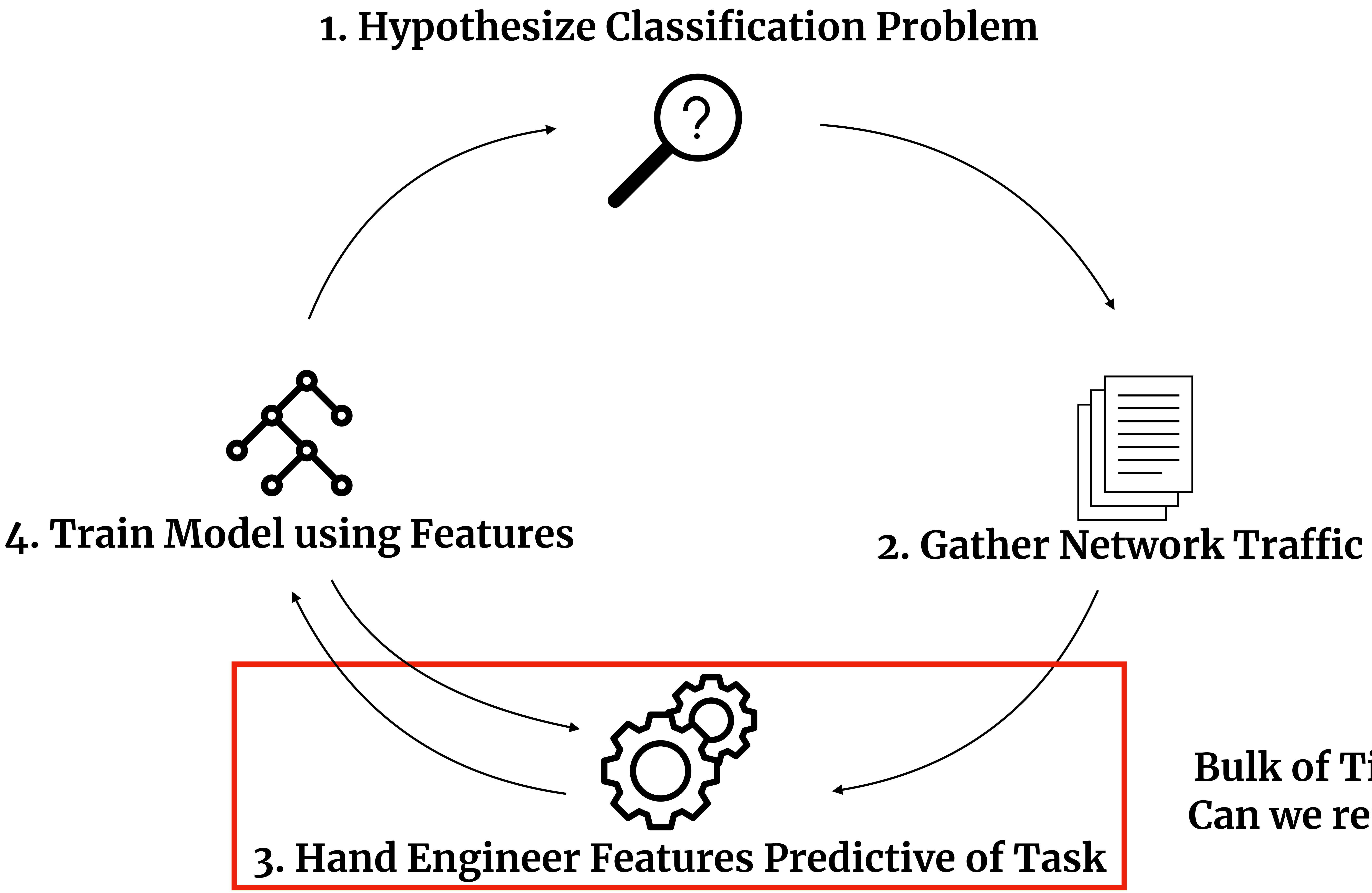
Machine Learning in Networking

- Device fingerprinting
- Passive OS Detection
- Website fingerprinting
- Protocol fingerprinting
- Application identification

Machine Learning in Networking



Machine Learning in Networking



Bulk of Time spent here:
Can we remove this step?

Outline

- Motivation
- Methodology
- nPrint - Active Device Fingerprinting
- nPrint - Application Identification
- Conclusions

Inspiration

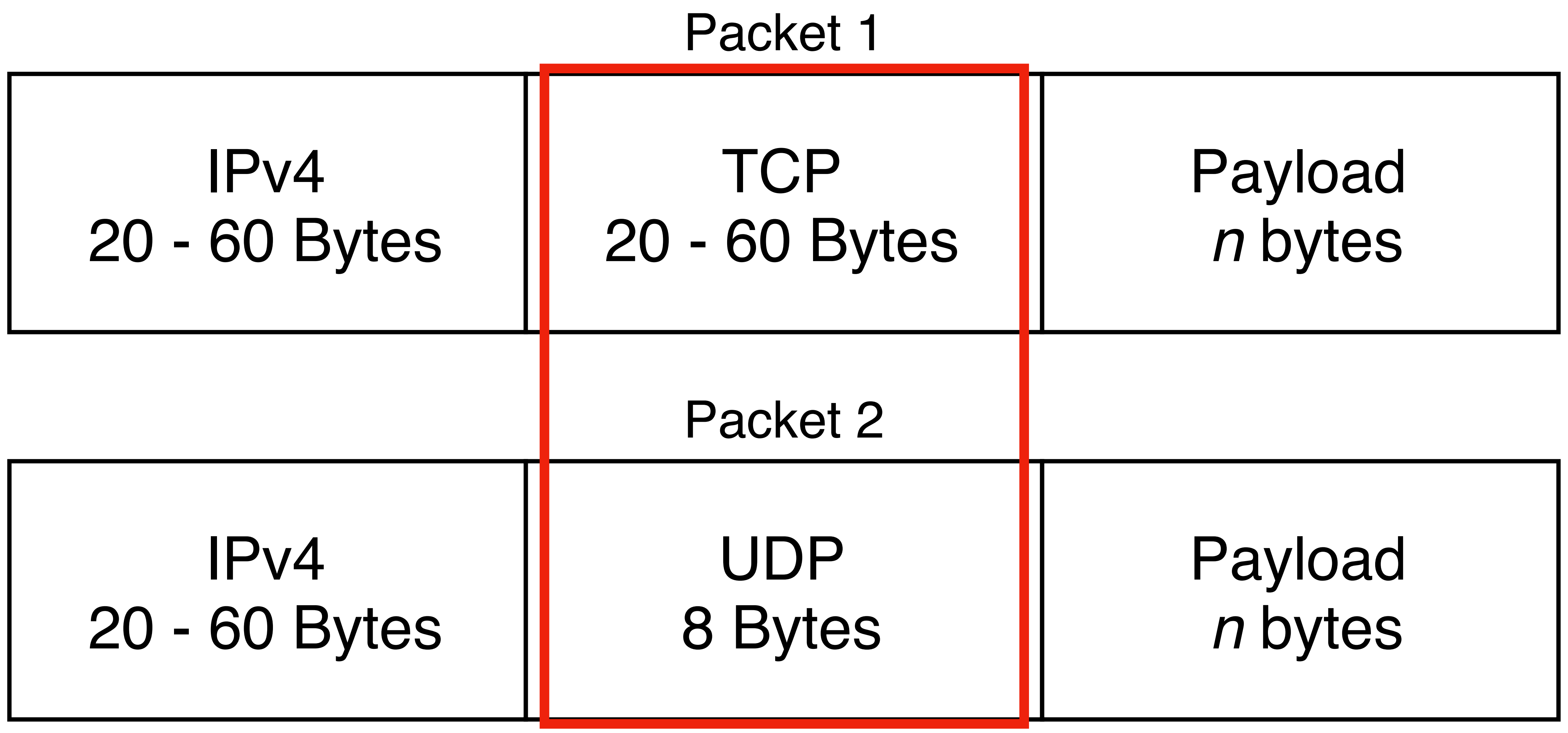
- Inspiration - deep learning techniques perform well with problems related to image classification, where pictures have a standard representation of values
  - Related work shows deep learning techniques effective in website fingerprinting for Tor [1,2]

- Problem! Outside of Tor, network traffic is not as simple

Packet Representation - Requirements

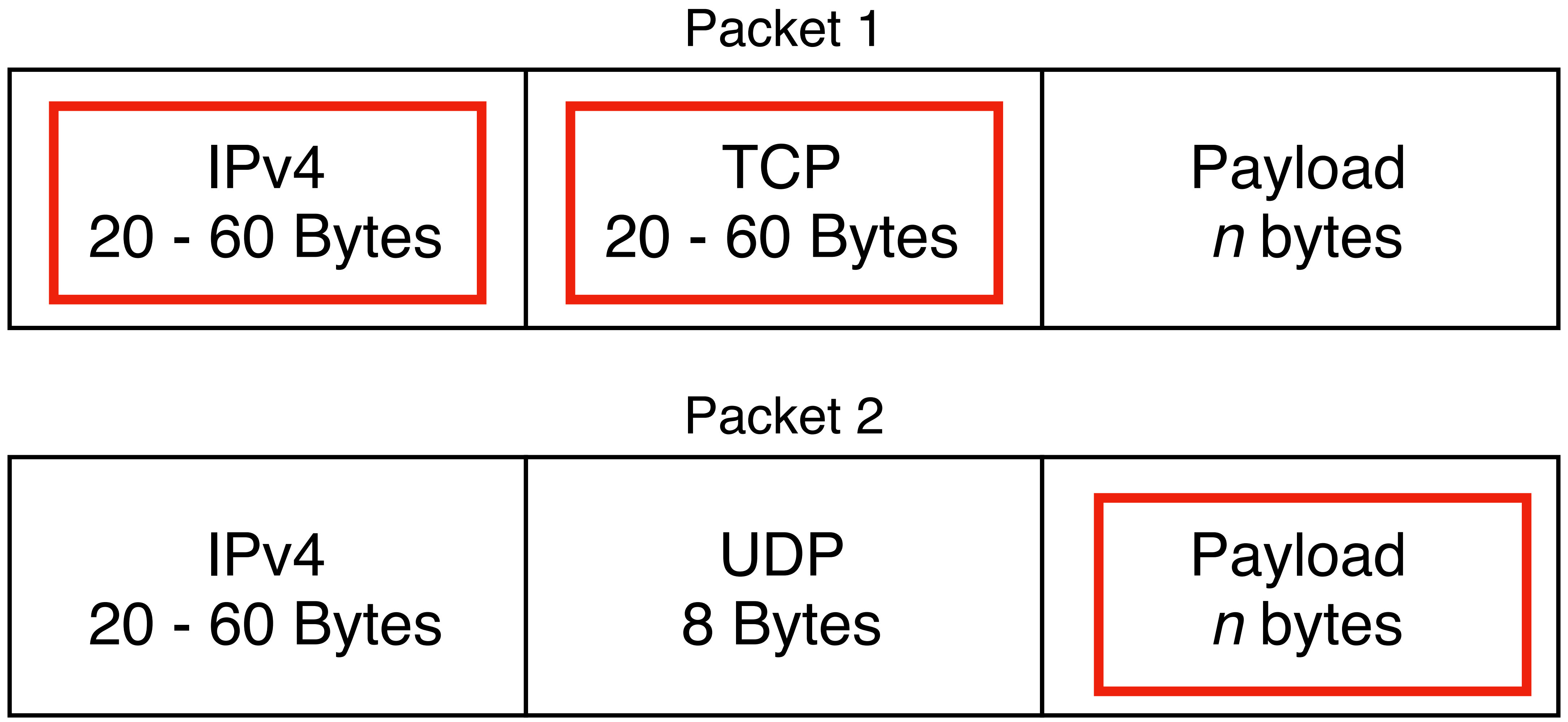
Packet Representation - Completeness

- Complete: each feature is represented for every packet
  - Every field in each packet must be able to be included in the representation
  - Our intuition is that the models can determine which features are important for a given problem without human guidance



Packet Representation – Constant Size Per Problem

- Complete: each feature is represented for every packet
- Constant Size Per Problem: the size of the representation is the same for each packet
  - Requirement for many machine learning techniques



Packet Representation – Semantic

- Semantic view involves packets broken into headers
- Encode each header field as a feature
- <u>Constant size</u>

Semantic Representation: (IP / TCP) Packet

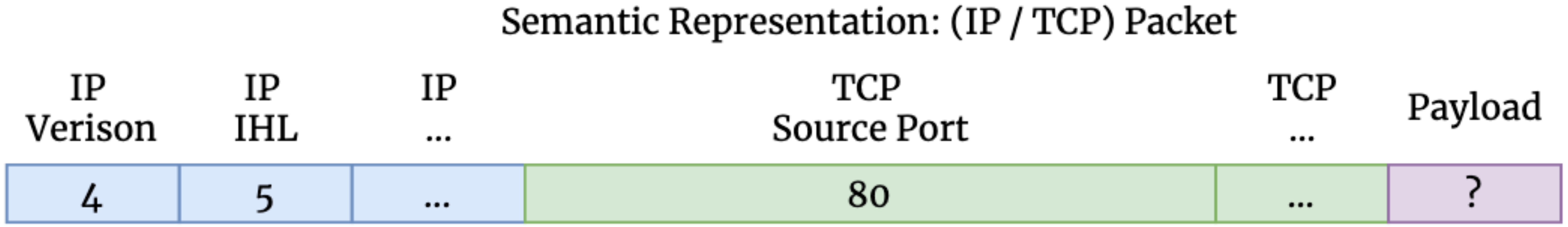
IP Verison	IP IHL	IP ...	TCP Source Port	TCP ...	Payload
4	5	...	80	...	?

Packet Representation – Semantic

- Problem – Loses the ordering of options!
  - Example – TCP Options

- Problem – How do we represent less structured parts of the packet?
  - Example: Payloads can’t really be represented numerically

- Problem –Normalization requires multiple passes of the data



Packet Representation –Inherently Normalized

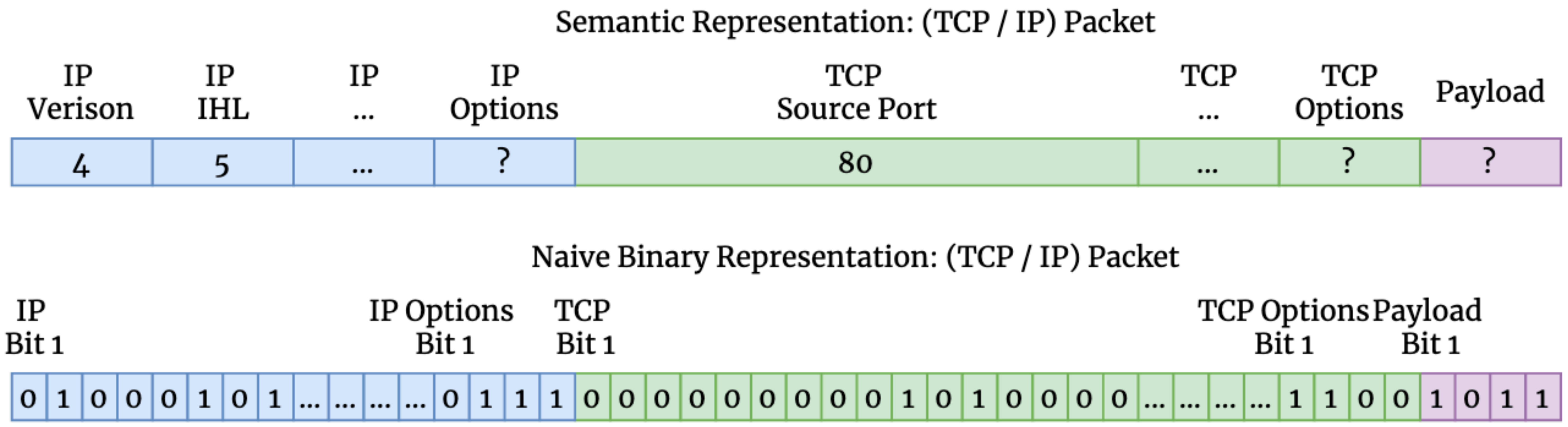
- Complete: each feature is represented for every packet
- Constant size per problem: the size of the representation is the same for each packet
- Inherently Normalized: each feature ranges between 0 and 1

Packet Representation – Naive Binary

- Turn classic semantic view of packets on its head
  - <u>think of packets as a collection of bits</u>

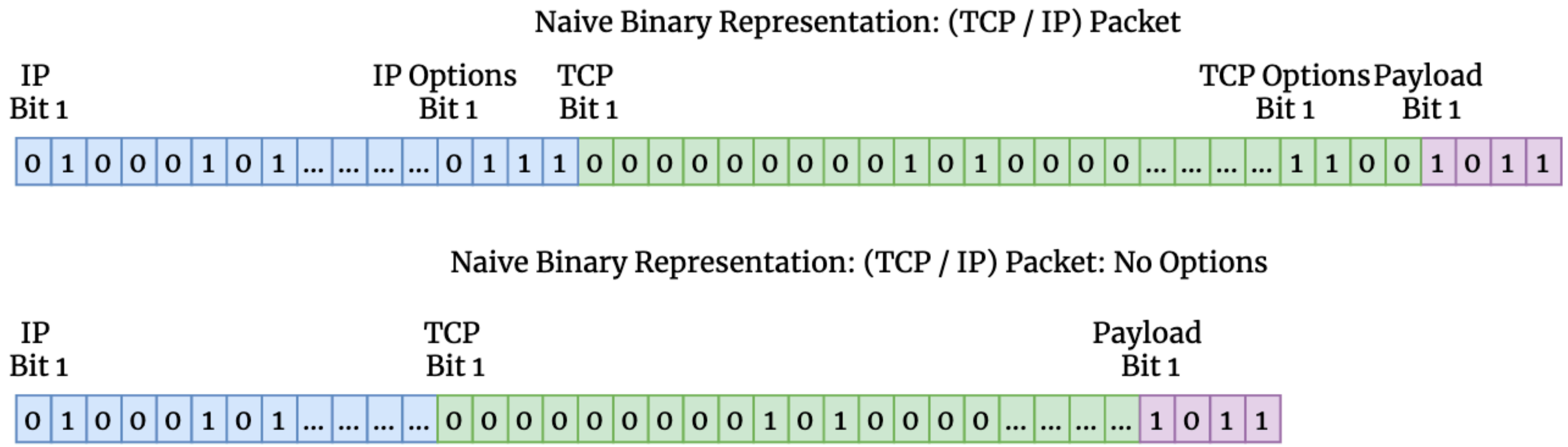
- Complete – each bit can be represented

- Inherently Normalized – each feature represented between 0 and 1



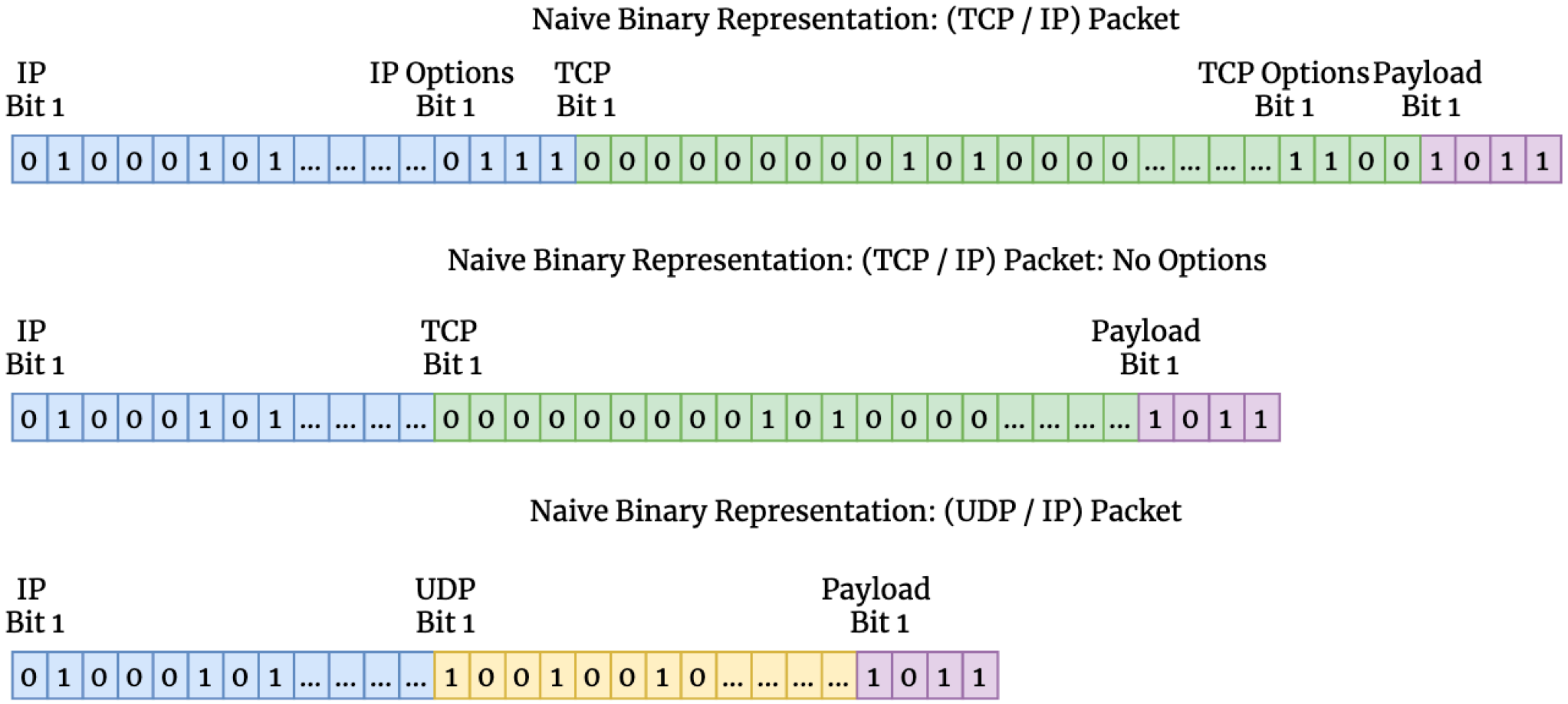
Packet Representation – Naive Binary

- Problem! – different bits (features) in different packets can have different meanings



Packet Representation –Naive Binary

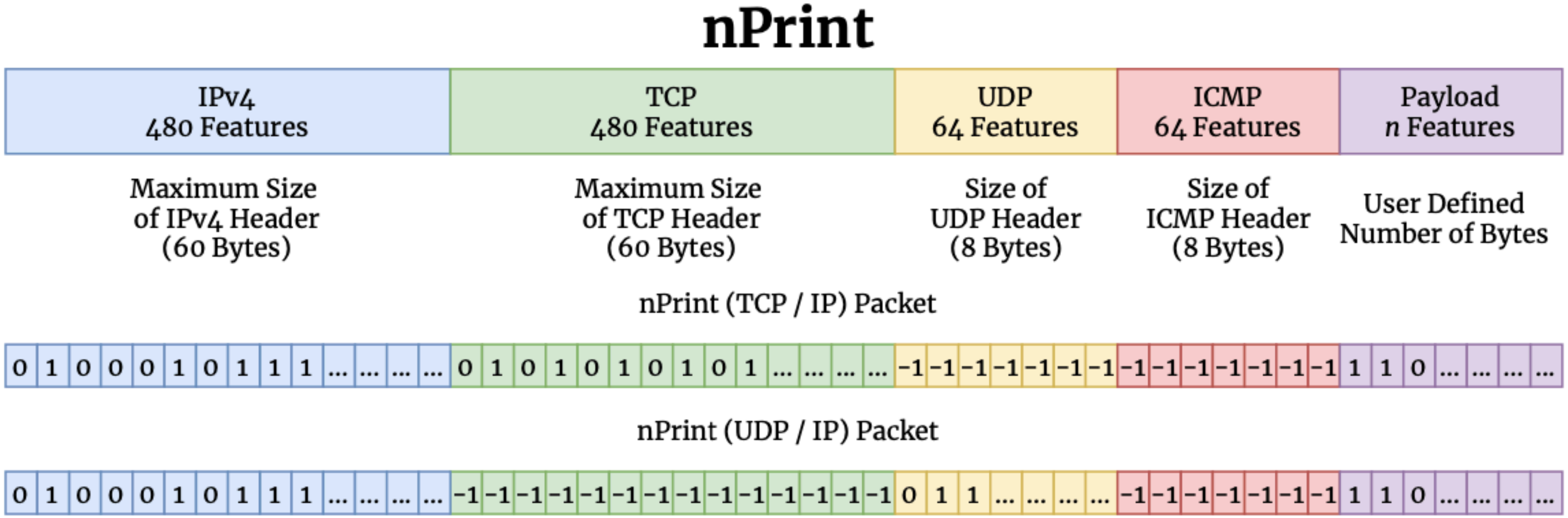
- Even worse when we have different types of packets…



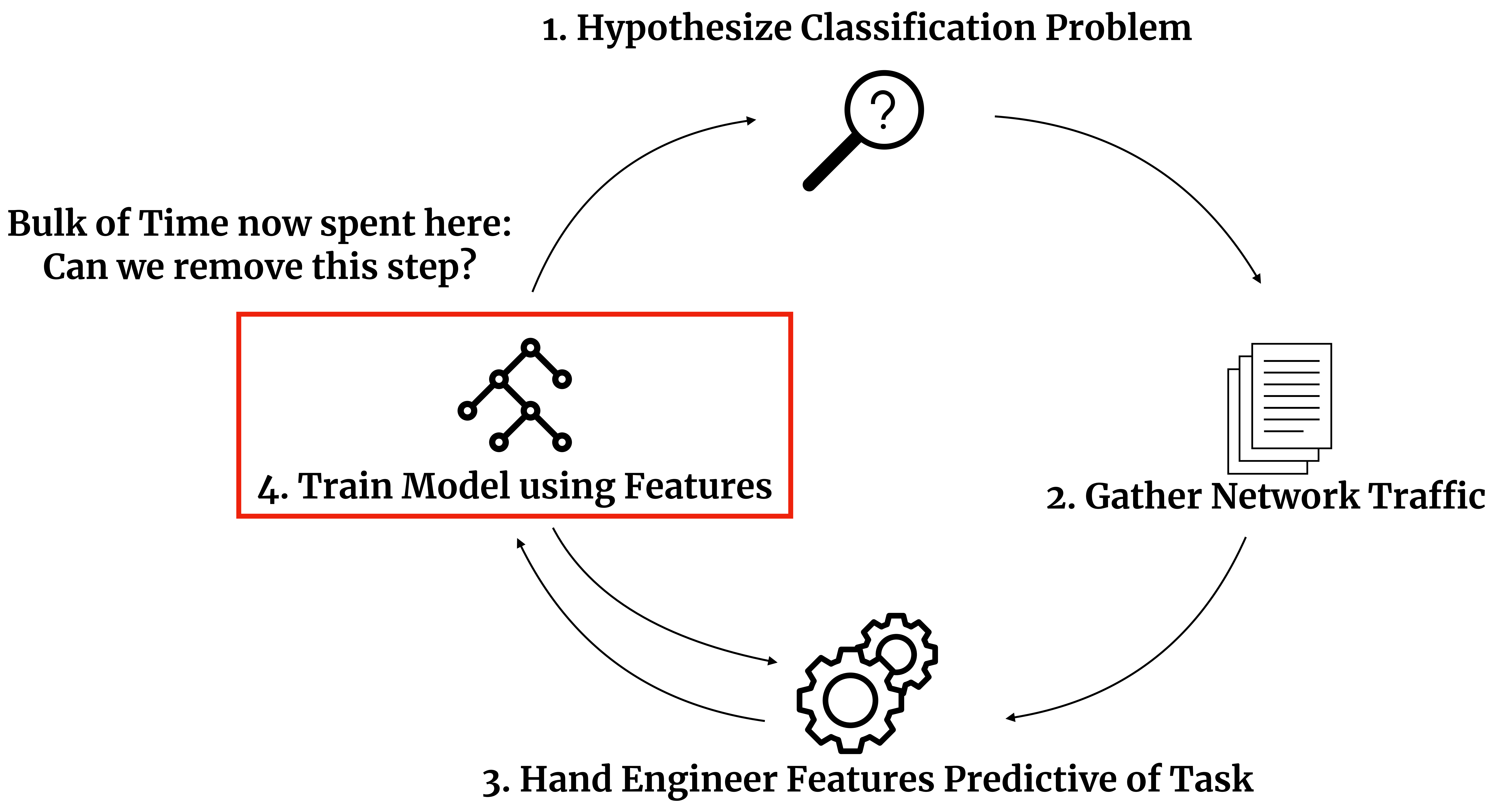
Packet Representation – Alignment

- Complete: each feature is represented for every packet
- Constant size: the size of the representation is the same for each packet
- Inherently Normalized: each feature ranges between 0 and 1
- Aligned: every location in the representation has the same meaning across all packets
  - Encodes semantic structure
  - Allows for interpretability

Packet Representation – nPrint



New Bottleneck



Training Models

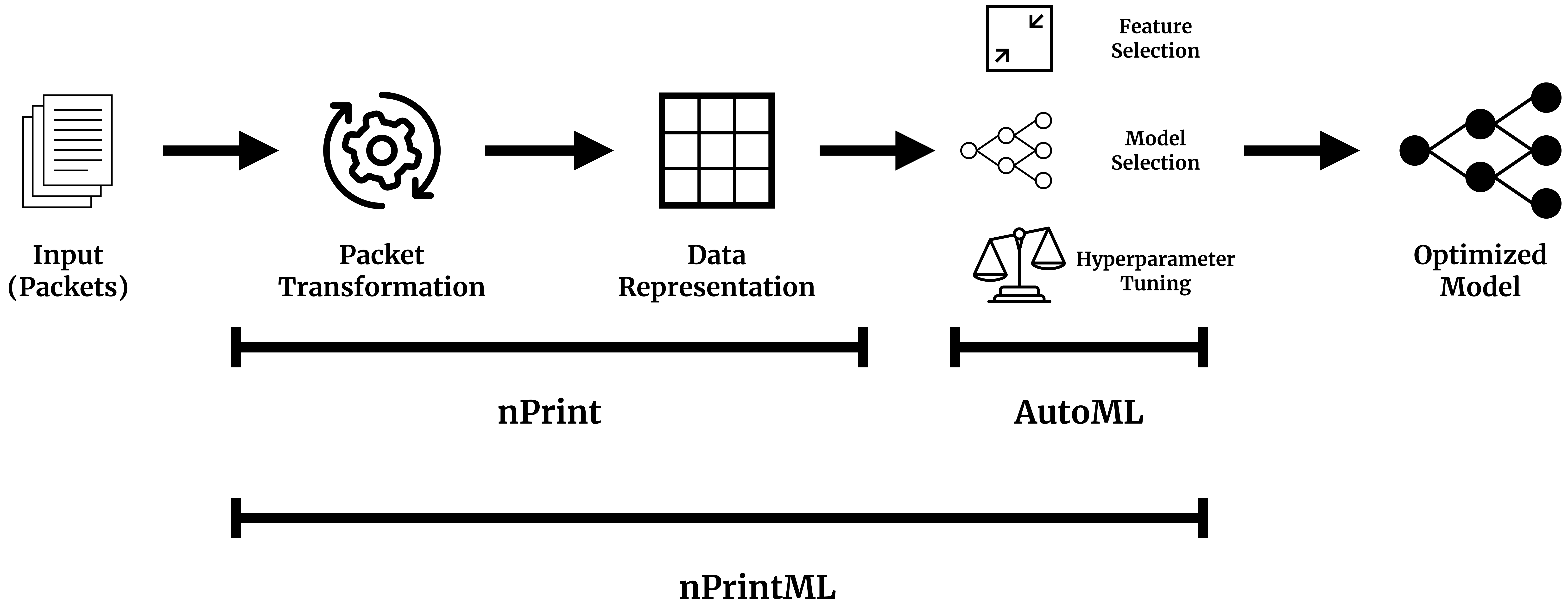
- Previous work (including ours):
  - Pick your favorite model(s)
  - Write code to search some hyperparameters for that model
  - Choose the best one in your search space

- AutoML
  - Do the following in a more principled manner:
    - Model selection
    - Feature selection
    - Hyperparameter search

AutoGluon AutoML

- Allow state-of-the-art AutoML to perform all model optimizations
- Achieves higher performance than other tools due to model ensembling.
- Allows us to train, optimize, and test **over 50 models** from 5 different base model classes

The nPrint Pipeline



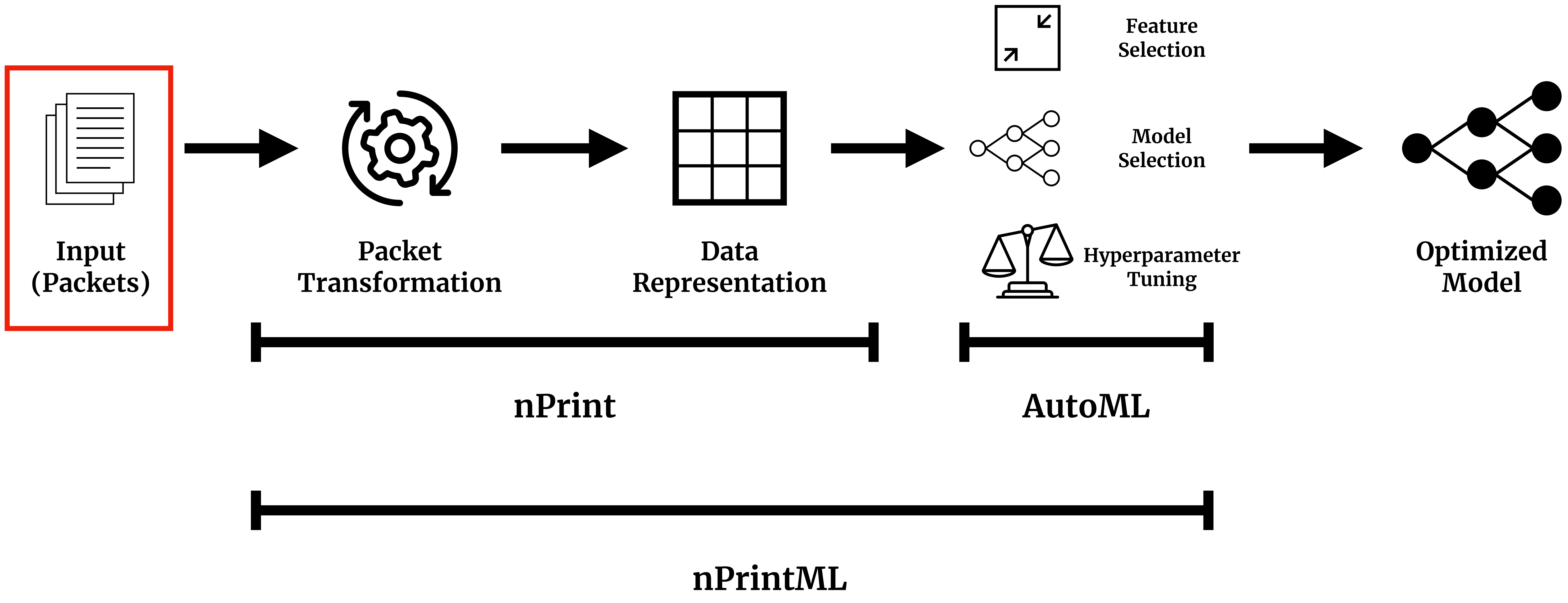
Outline

- Motivation
- Methodology
- nPrint - Active Device Fingerprinting
- nPrint - Application Identification
- Conclusions

Active Device Fingerprinting– Dataset

Vendor	Device Type	Labeled Devices
Adtran	Network Device	1,449
Avtech	IoT Camera	2,152
Axis	IoT Camera	2,653
Chromecast	IoT Streaming	2,872
Cisco	Network Device	1,451
Dell	Network Device	1,449
H3C	Network Device	1,380
Huawei	Network Device	1,409
Juniper	Network Device	1,445
Lancom	Network Device	1,426
Miktrotik	Network Device	1,358
NEC	Network Device	1,450
Roku	IoT Streaming	2,403
Ubiquoss	Network Device	1,476
ZTE	Network Device	1,425

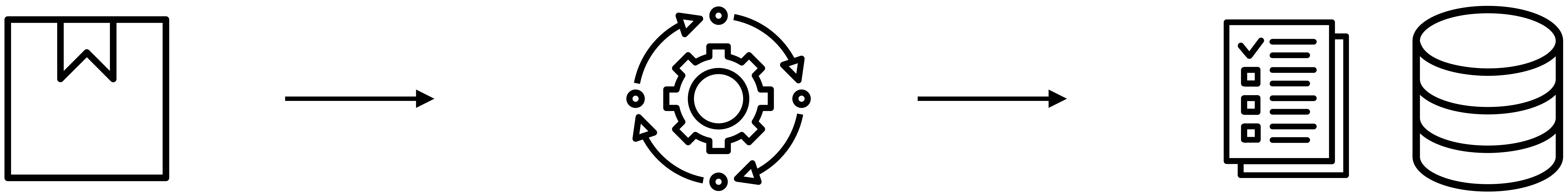
The nPrint Pipeline



Active Device Fingerprinting- Gathering Traffic

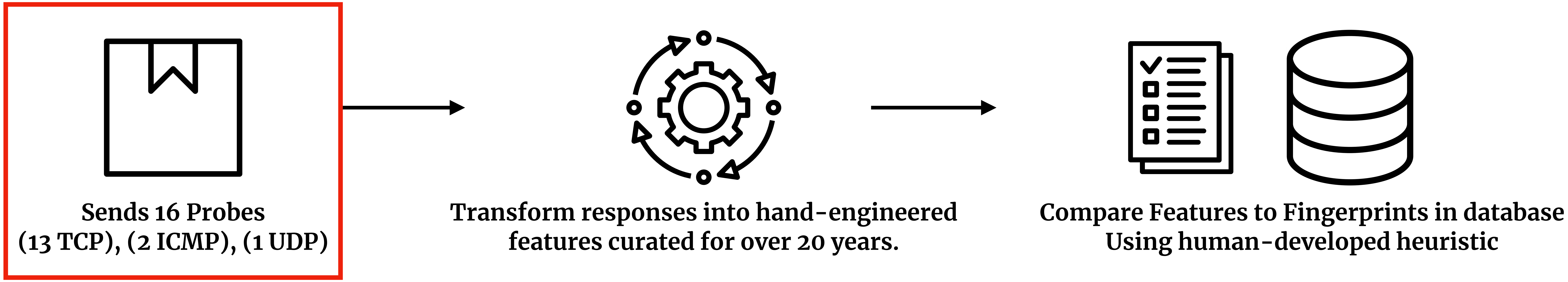
- Leverage Nmap, well known and used remote device fingerprinting tool

- Over 20 years of hand curated features and a hand-developed heuristic to fingerprint remote devices

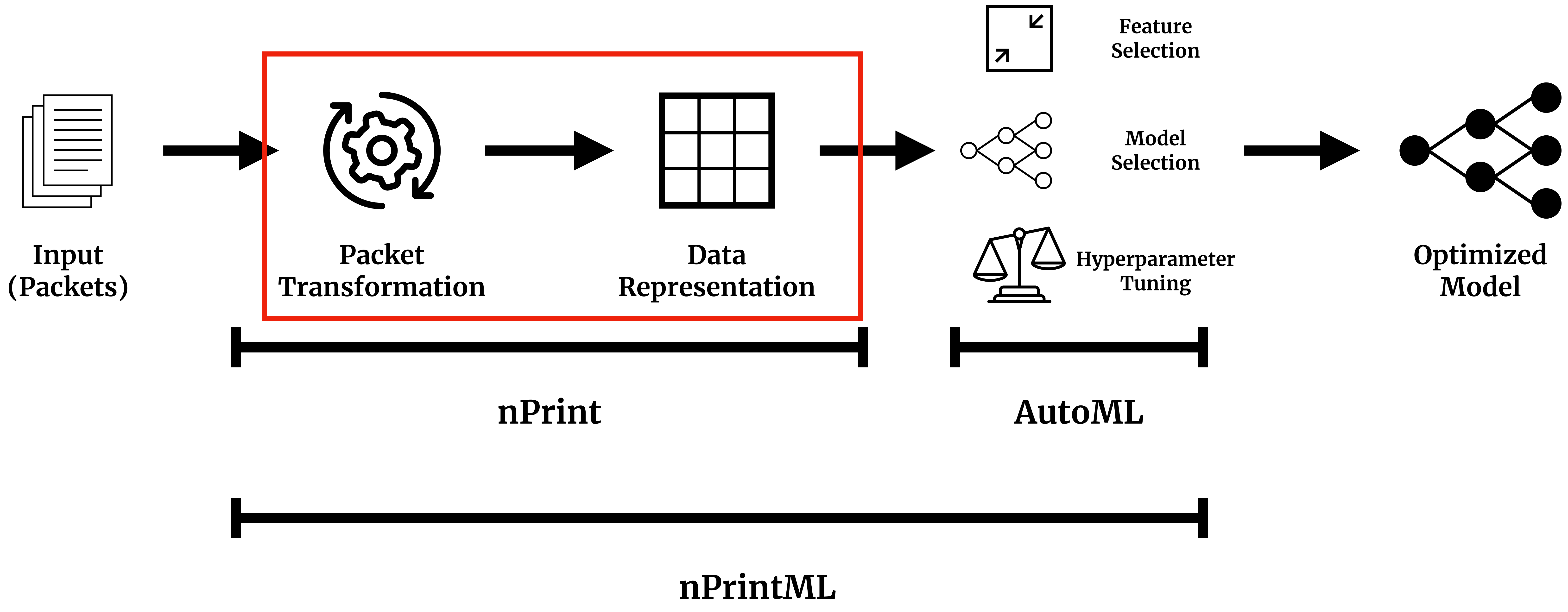


Active Device Fingerprinting- Gathering Traffic

- Leverage Nmap, well known and used remote device fingerprinting tool
- Over 20 years of hand curated features and a hand-developed heuristic to fingerprint remote devices



The nPrint Pipeline

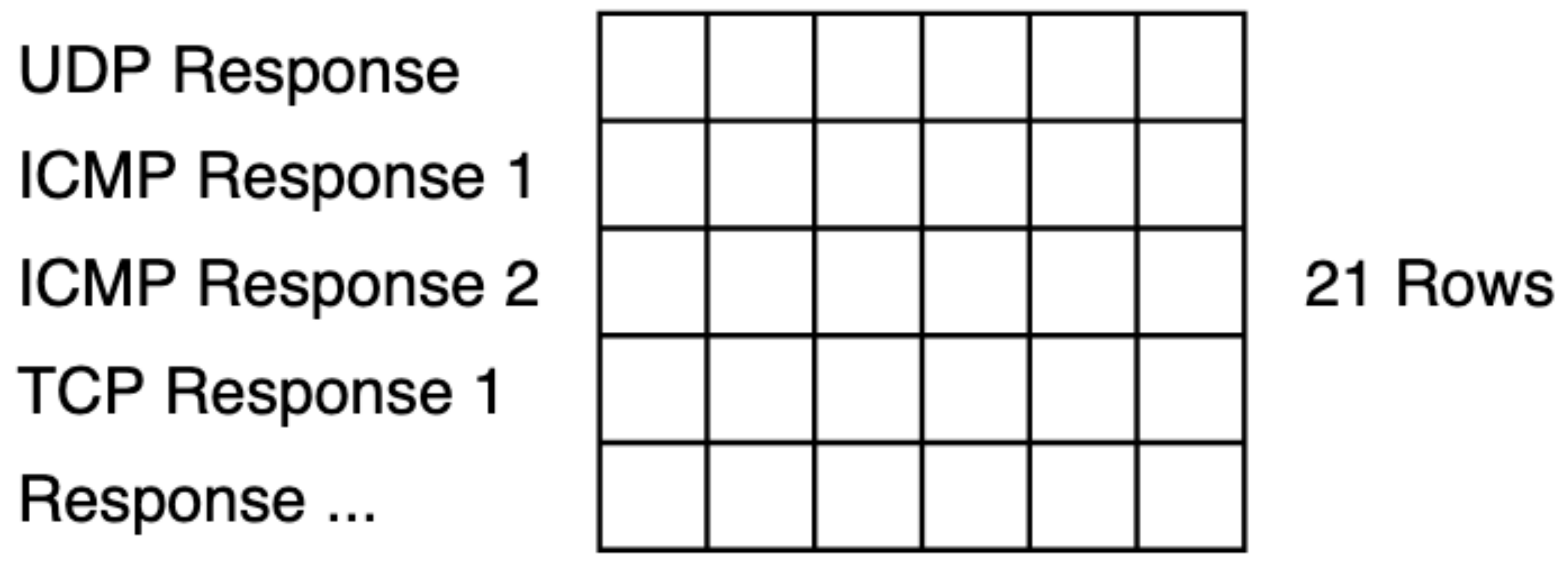


Active Device Fingerprinting– Packet Transformation

- 21 uniquely named responses from the sent Nmap probes

- Create fingerprint picture by sorting the responses and concatenating them

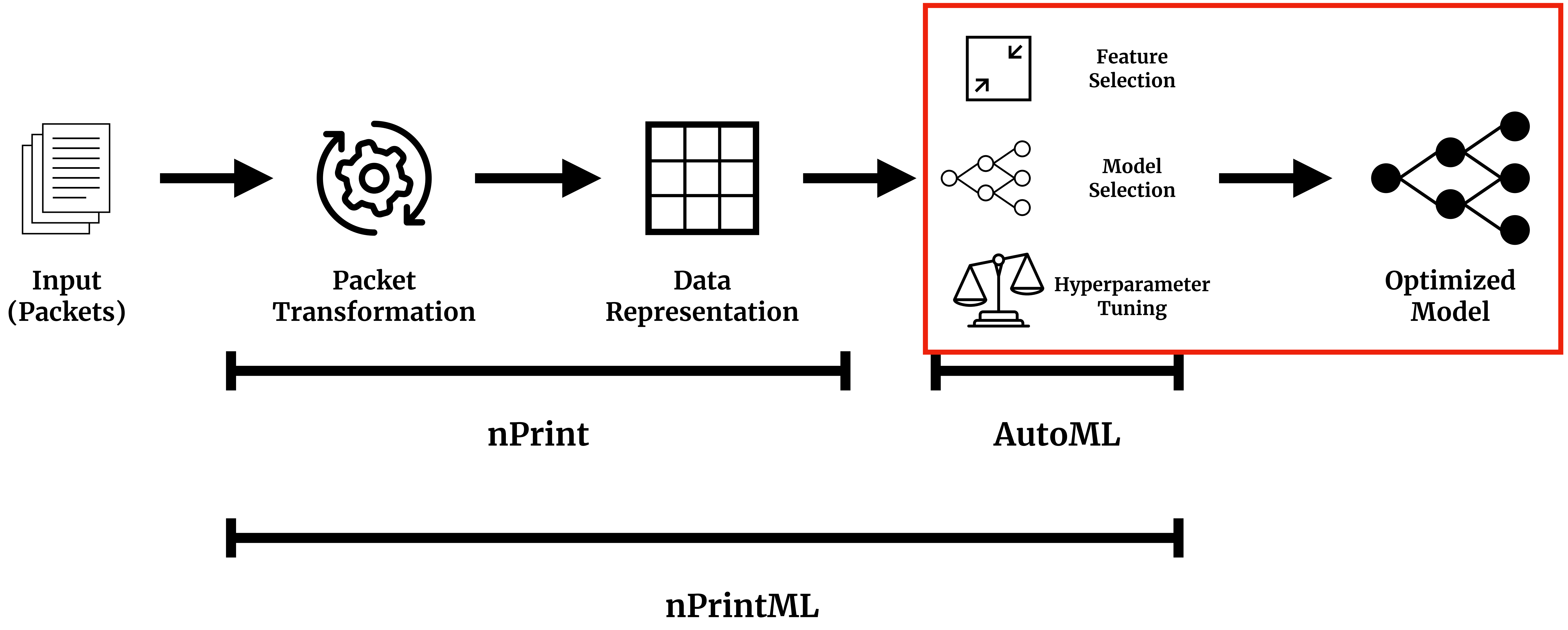
- Flattened version of the 2D image to the right



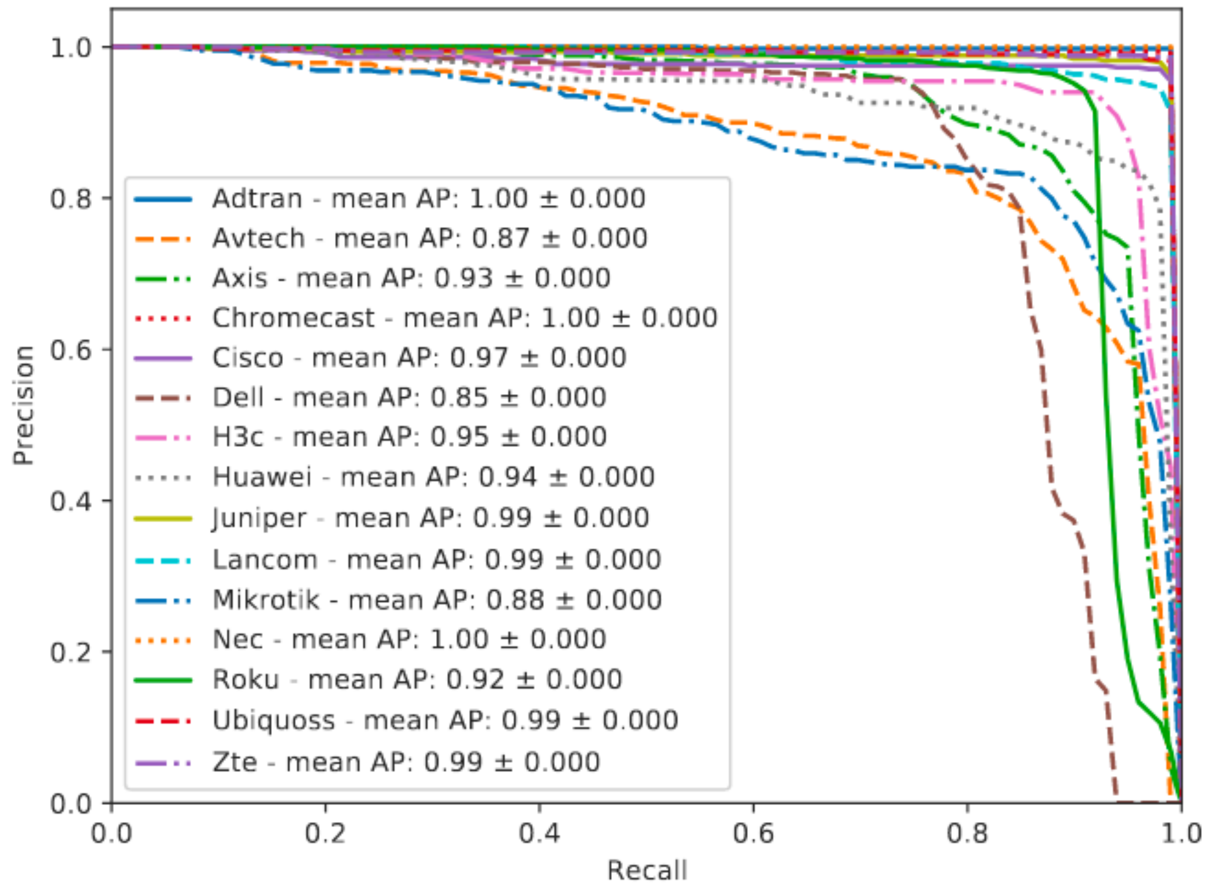
Nmap – Features

Test Name	Summary	Nmap Weight
Explicit Congestion Notification	TCP Explicit Congestion control flag.	100
ICMP Response Code	ICMP Response Code.	100
Integrity of returned probe IP Checksum	Valid checksum in an ICMP port unreachable.	100
Integrity of returned probe UDP Checksum	UDP header checksum received match.	100
IP ID Sequence Generation Algorithm	Algorithm for IP ID.	100
IP Total Length	Total length of packet.	100
Responsiveness	Target responded to a given probe.	100
Returned probe IP ID value	IP ID value.	100
Returned Probe IP Total Length	IP Length of an ICMP port unreachable.	100
TCP Timestamp Option Algorithm	TCP timestamp option algorithm.	100
Unused Port unreachable Field Nonzero	Last 4 bytes of ICMP port unreachable message not zero.	100
Shared IP ID Sequence Boolean	Shared IP ID Sequence between TCP and ICMP.	80
TCP ISN Greatest Common Divisor	Smallest TCP ISN increment.	75
Don’t Fragment ICMP	IP Don’t Fragment bit for ICMP probes.	40
TCP Flags	TCP flags.	30
TCP ISN Counter Rate	Average rate of increase for the TCP ISN.	25
TCP ISN Sequence Predictability Index	Variability in the TCP ISN.	25
IP Don’t Fragment Bit	IP Don’t Fragment bit.	20
TCP Acknowledgment Number	TCP acknowledgment number.	20
TCP Miscellaneous Quirks	TCP implementations, e.g, reserved field in TCP header.	20
TCP Options Test	TCP header options, preserving order.	20
TCP Reset Data Checksum	Checksum of data in TCP reset packet.	20
TCP Sequence Number	TCP sequence number.	20
IP Initial Time-To-Live	IP initial time-to-live.	15
TCP Initial Window Size	TCP window size.	15

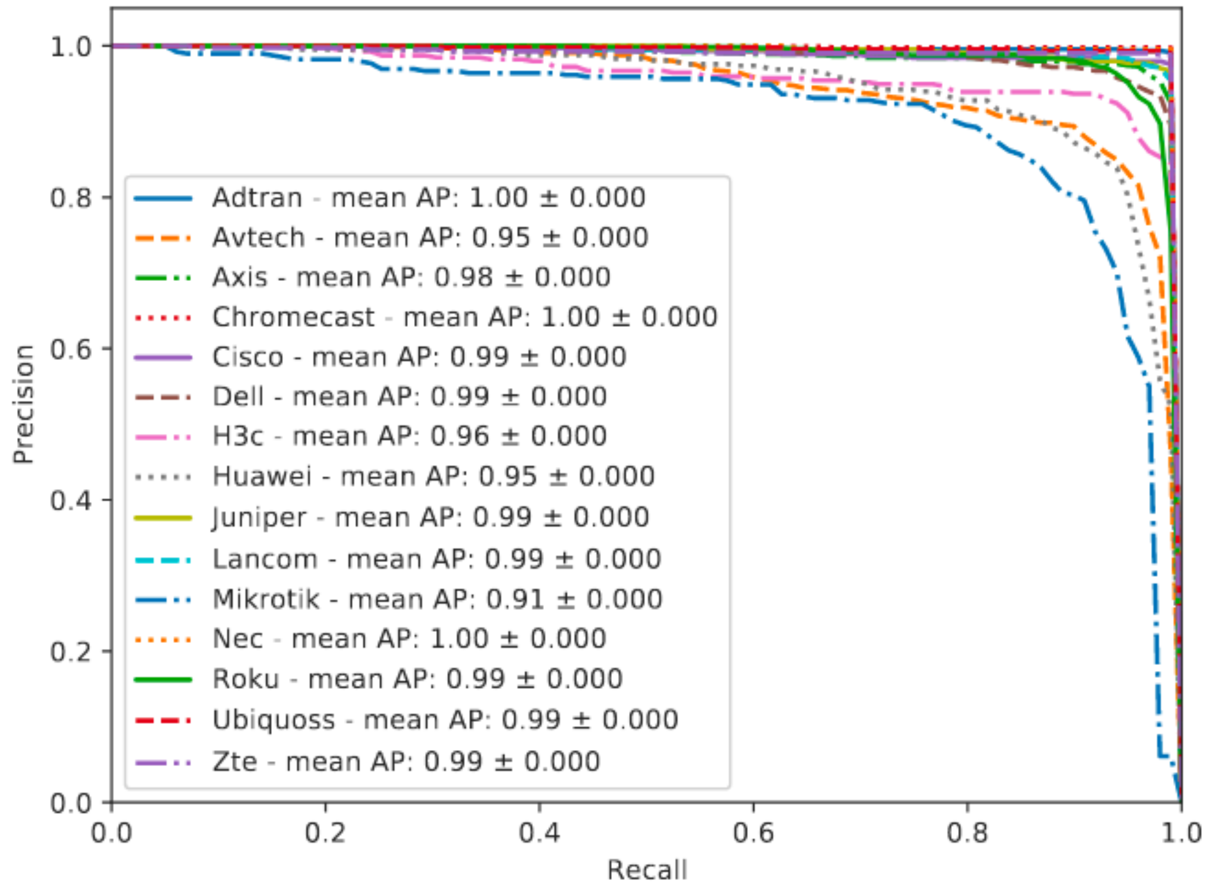
The nPrint Pipeline



Active Device Fingerprinting–Outperforming Nmap



(a) *Nmap PR*



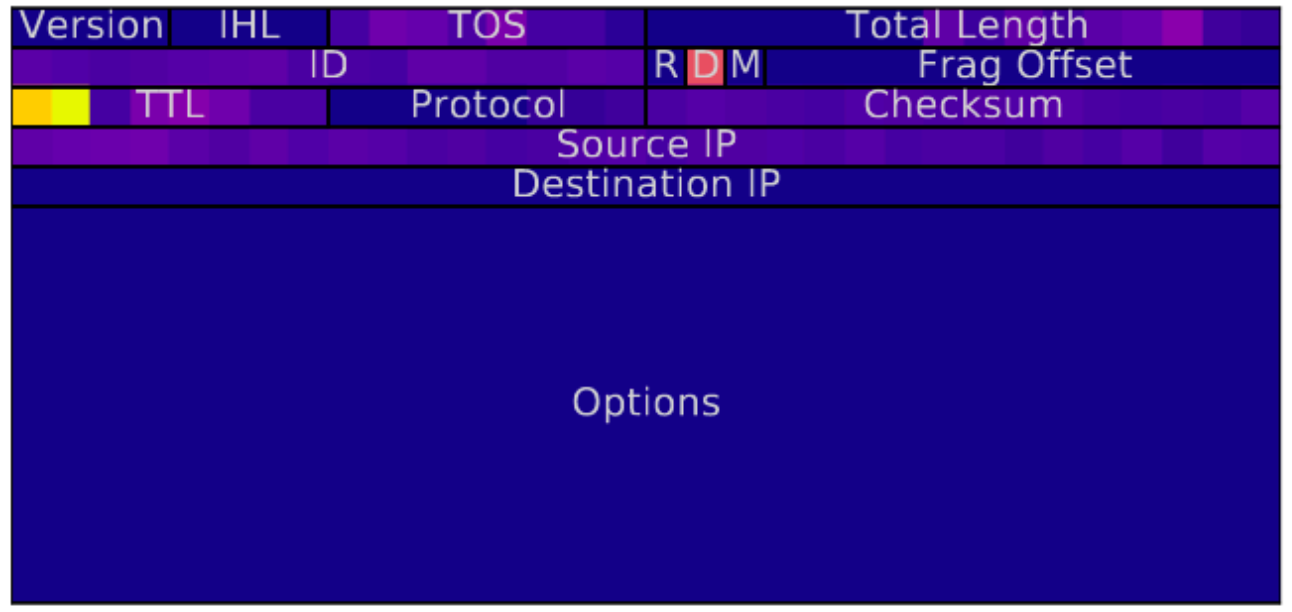
(b) *nPrint PR*

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

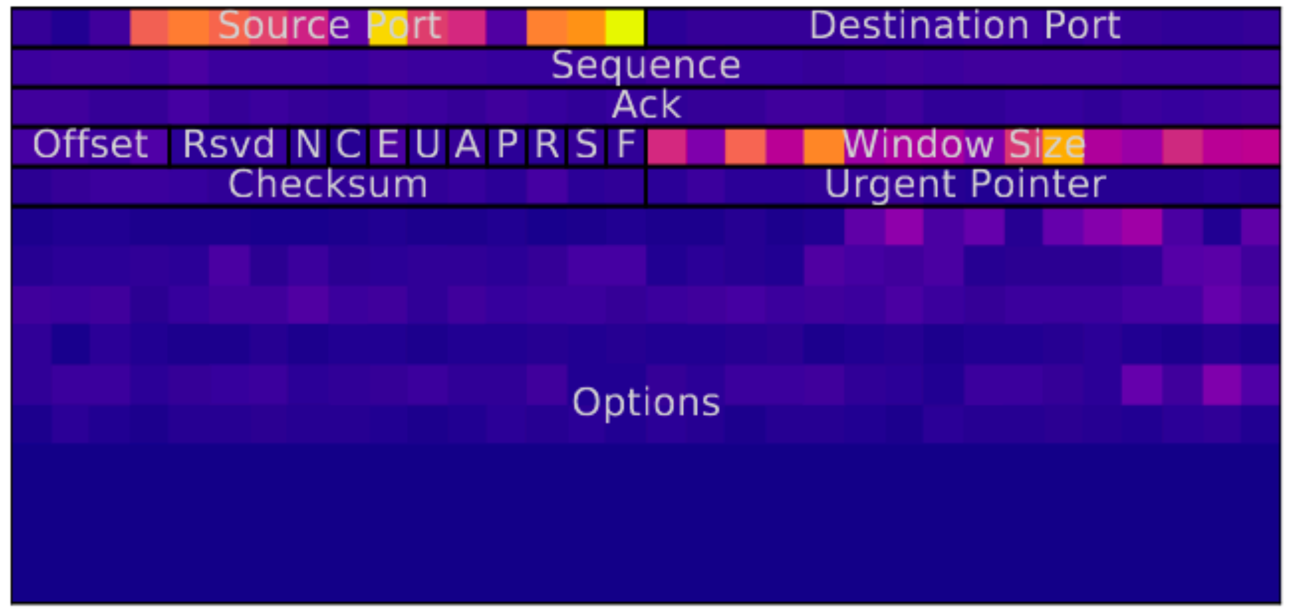
Active Device Fingerprinting– Feature Importance

- <u>Interpretable machine learning</u>

- Automatically learns features encoded into device fingerprinting tools
  - IP TTL
  - TCP options, window size



(a) *IPv4*



(b) *TCP*

(c) *ICMP*

Outline

- Motivation
- Methodology
- nPrint - Active Device Fingerprinting
- <u>nPrint - Application Identification</u>
- Conclusions

Application Identification - Dataset

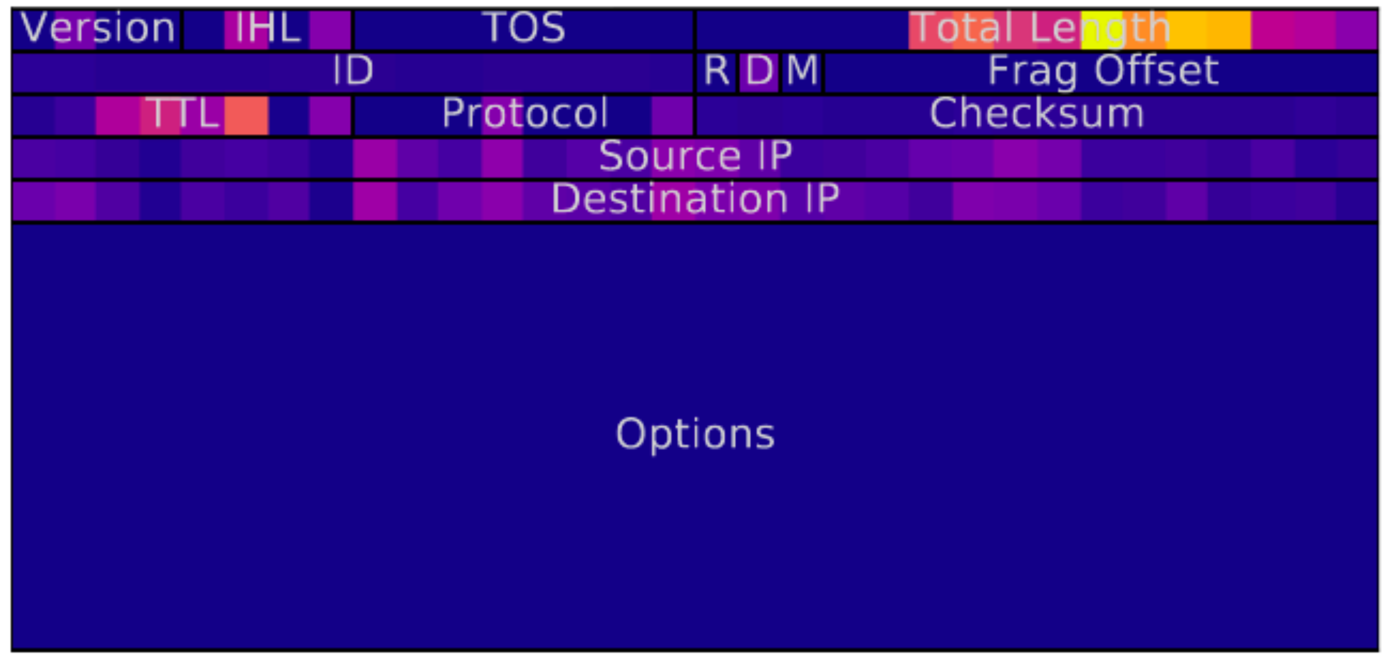
- Collection of ~7000 dTLS handshakes in an effort to identify Snowflake, a new pluggable Transport in Tor
- Manual feature engineering leads to 99% accuracy on just the application. Can we match this with the nPrint pipeline on a harder problem?

	Application Handshakes			
Browser	Snowflake	Facebook	Google	Discord
Firefox	991	796	1000	992
Chrome	0	784	995	997

Application Identification - Performance

- nPrint can automatically detect features in a noisy environment
- nPrint performs well across models and trains quickly

Model Architecture	Fit Time (Seconds)	Total Inference Time (Seconds)	F1
Random Forest	3.69	0.37	99.8
ExtraTrees	3.89	0.43	99.9
KNeighbors	3.90	8.95	96.0
LightGBM	5.21	0.15	99.8
Catboost	9.00	0.38	99.7
Weighted Ensemble	46.1	0.45	99.9
Neural Network	85.58	29.9	99.7



(a) *IPv4*

(b) *UDP*

(c) *DTLS*

Go try it!

- **nPrint is publicly available**
  - Ethernet, IPv4, fixed IPv6 headers, UDP, TCP, ICMP, payloads
  - Relative timestamps
  - Absolute timestamps
  - Formats: live capture, PCAPs, CSV scan data (Zmap)

- **nPrint runs in a variety of environments**
  - ~300KB memory footprint
  - Roughly zero loss on 10 GbE university link with pf_ring

- **nPrintML is publicly available**
  - Combines nPrint and AutoGluon
  - Application Identification case study:
    - **"nprintml —pcap-dir pcaps/ -L labels.csv -a pcap -4 -u -p 10"**
  - Passive OS detection case study:
    - **"nprintml -P traffic.pcap -L labels.csv -a index -4 -t"**

Conclusions

- Introduce nPrint, a standard data representation for network traffic analysis problems

- Remove human driven feature engineering step from typical machine learning workflow for network traffic classification problems

- Show better performance than widely used device and OS fingerprinting tools

References

1. Rimmer, Vera, et al. "Automated website fingerprinting through deep learning." 25th *Annual Network & Distributed System Security Symposium*. NDSS, 2018
2. Oh, Se Eun, Saikrishna Sunkam, and Nicholas Hopper. "p1-FP: Extraction, Classification, and Prediction of Website Fingerprints with Deep Learning." *Proceedings on Privacy Enhancing Technologies* 2019.3 (2019): 191-209.
3. https://skminhaj.wordpress.com/2016/02/15/tcp-segment-vs-udp-datagram-header-format/
4. https://arxiv.org/pdf/2006.13086.pdf
5. https://www.shodan.io/
6. Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization", 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018