KDD 2022 Tutorial:

Deep Learning for Network Traffic Data

https://github.com/manish-marwah/KDD2022Tutorial

Manish Marwah, Martin Arlitt

MICRO FOCUS

Network traffic data

- Network traffic data is an extremely rich source of information
- It is highly voluminous and complex

- It has a lot of applications, particularly in areas of
  computer security, e.g.,
  - intrusion detection
  - malware detection

  and network management, e.g.,
  - device identification
  - application identification

- These can be cast as ML problems, but there are lots of challenges

Outline

- Part 1: Network traffic background and challenges
  - What is network traffic data?
  - Where and how is it collected?
  - Packet vs flow-based data
  - Data sets
  - Applications
  - Challenges
- Part 2: Representation learning for network traffic data
  - Background
  - Representation learning for IP addresses
  - Traffic as an image
  - Traffic as a sequence
  - Traffic as a graph
- Part 3: Synthetic data generation for network traffic data
  - Generative model
  - GAN-based generators
  - Auto-regressive neural generator

Part 1: Network Traffic Data

Part 1 Outline

- Network traffic data
- Applications
- Data sets
- Challenges

Background: the Internet Protocol stack

- Application: supports end-user services and network applications
  - HTTP, SMTP, DNS, FTP, NTP
- Transport: end to end data transfer
  - TCP, UDP
- Network: routing of datagrams from source to destination
  - IPv4, IPv6, BGP, RIP
- Data Link: channel access, framing, flow/error control, hop by hop basis
  - PPP, Ethernet, IEEE 802.11b WiFi
- Physical: transmission of bits

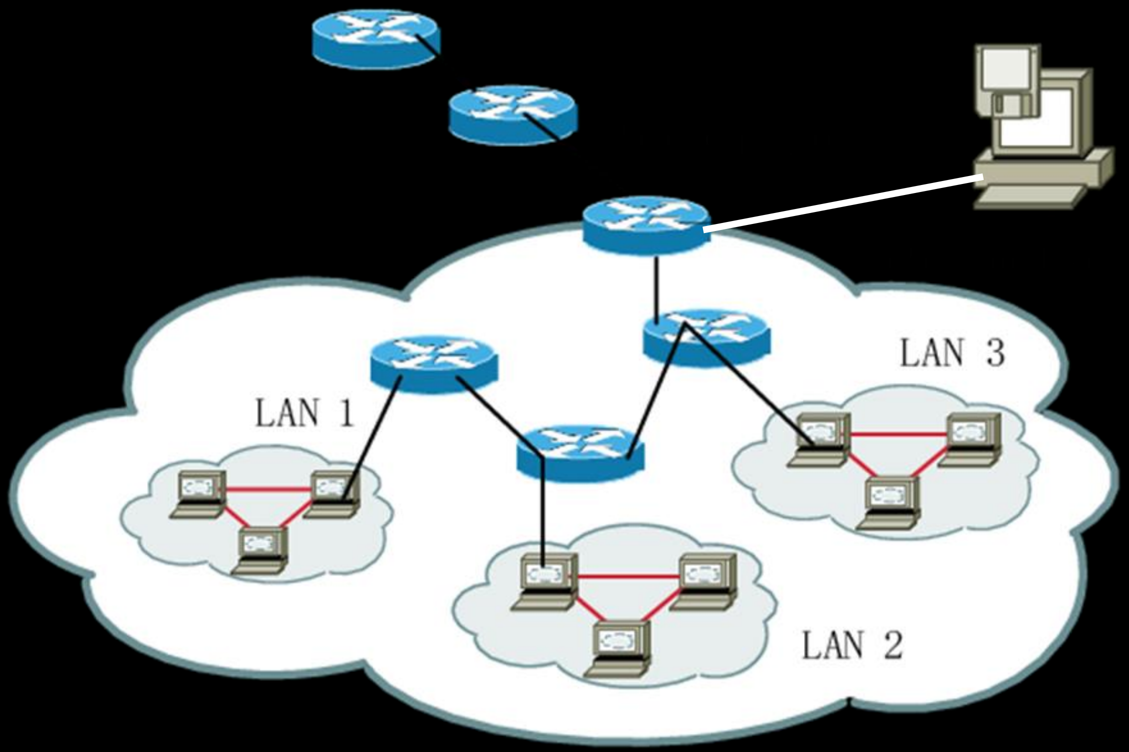
What is Network traffic Data?

- Traffic packets captured on the network
- Contain protocols headers and data
- Different types of traffic – text, video, audio, …

www.istockphoto.com

Where is Network Traffic Data Collected?

- Usually at an intranet edge where it can capture ingress / egress data
  - This simplifies collection
  - It may however not observe all traffic
    - Internal traffic that doesn’t pass the collection point
    - Packets that are dropped due to capacity limits of the collection infrastructure



https://www.researchgate.net/figure/The-topology-of-network-traffic-captured-environment_fig2_275415518

How is network traffic data collected?

- Either hardware or software measurement tools

- Hardware: specialized equipment
  - Examples: HP 4972 LAN Analyzer, DataGeneral Network Sniffer, NavTel InterWatch 95000, Endace DAG, others...
  - These are faster, but more expensive ($$$)

- Software: special software tools
  - Examples: tcpdump, wireshark, others...
  - These are cheaper (free!), but also slower (miss packets)

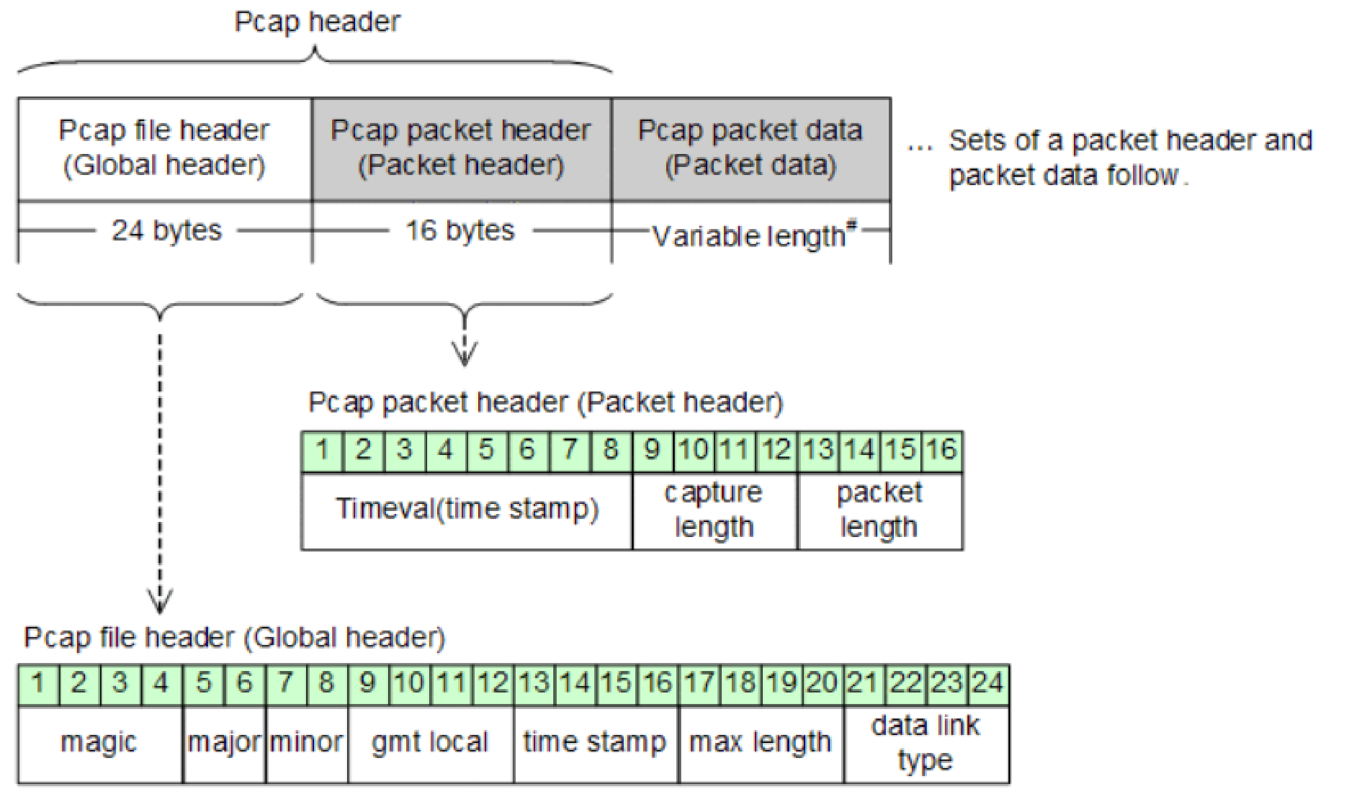
Tools for packet capture / parsing

- tcpdump https://www.tcpdump.org
  - Unix-based tool from mid-to-late 1980's
  - Distributed with BSD Unix (Berkeley Software Distribution)
  - Command-line interface; must be root to run it
  - Uses the Berkeley Packet Filter (BPF) in operating system
  - Writes to a PCAP file format; uses libpcap library

- Wireshark https://www.wireshark.org
  - PC-based tool from the early 2000's
  - Formerly called Ethereal (name change in May 2006)
  - Free and open-source tool
  - Multi-layer visualization and analysis of packet traces
  - Also supports PCAP file format

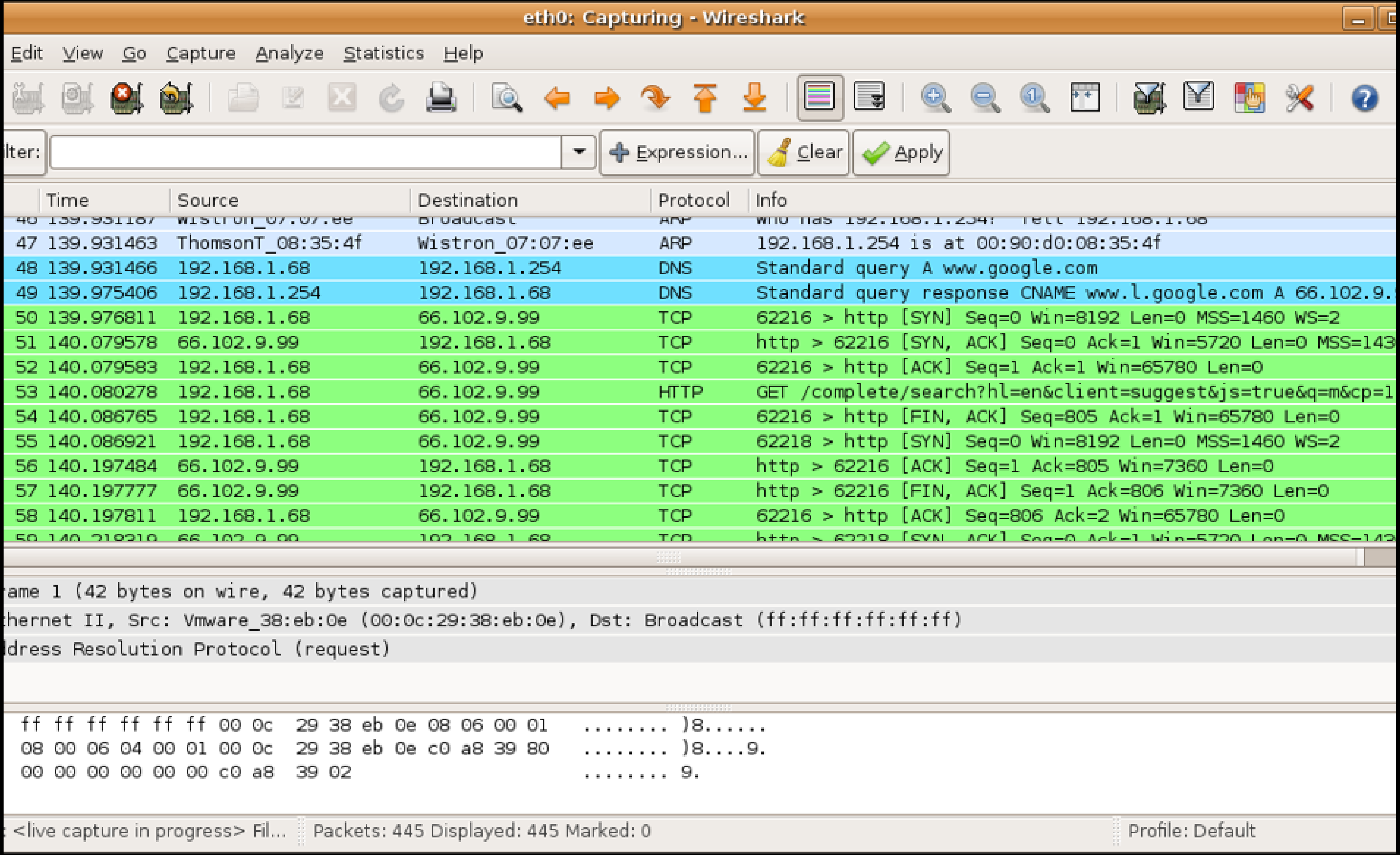
Collection format

- pcap ("packet capture")
  - Defines format and API for capturing network packets
  - Libpcap available from tcpdump.org



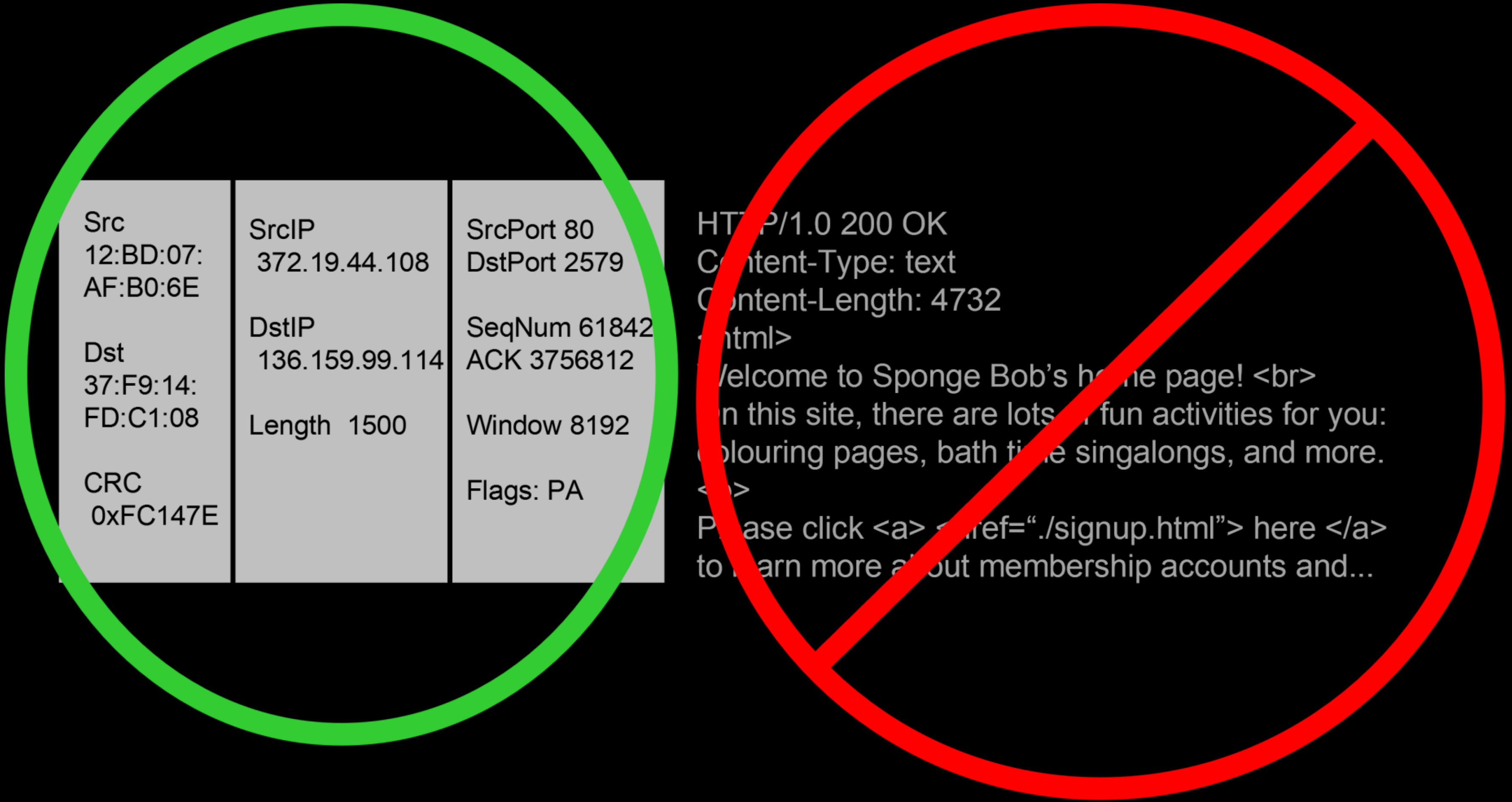
http://itdoc.hitachi.co.jp/manuals/3020/30203V0200e/BV020176.HTM

Wireshark example



Packet headers vs Deep Packet Inspection

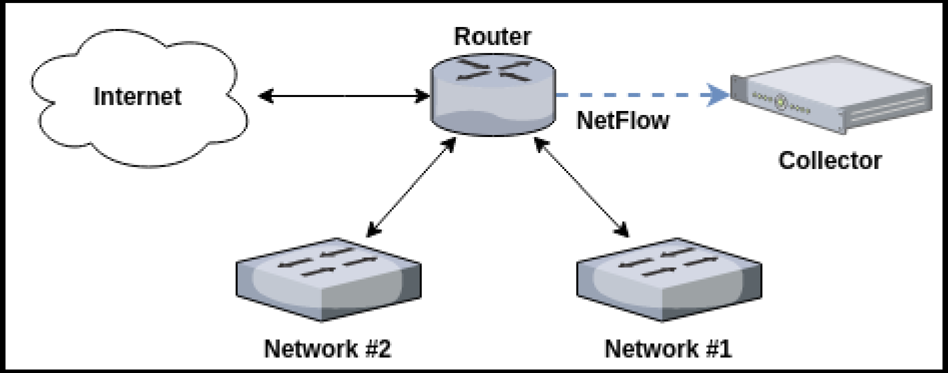
- Sometimes it makes sense to only examine a packet’s headers rather than its payload



Flow-based traffic data

Netflow

- Introduced on Cisco routers in 1996
- Multiplexes one or more packets into a “flow”
- Each record summarizes a **unidirectional** flow



https://cylab.be/blog/42/collecting-and-processing-netflow-on-ubuntu

Main Fields

- Connection 5-tuple
  - Source IP
  - Source Port
  - Destination IP
  - Destination Port
  - Transport protocol
- Start timestamp
- Duration
- Number of bytes
- Number of packets
- TCP flags

Netflow example

Terminal File Edit View Search Terminal Help

```
$ nfdump -R /var/cache/nfdump/
Date first seen          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2019-10-01 18:51:06.726     0.013 TCP    193.190.205.209:443   ->     192.168.0.14:42438        3      187     1
2019-10-01 18:51:06.726     0.002 TCP       192.168.0.14:42438 ->  193.190.205.209:443          2      104     1
2019-10-01 18:51:15.843     0.000 UDP      192.168.0.100:5353  ->      224.0.0.251:5353         1      155     1
2019-10-01 18:51:08.044     7.730 UDP       192.168.0.14:5353  ->      224.0.0.251:5353         2      162     1
2019-10-01 18:51:30.381     6.389 TCP       192.168.0.14:42504 ->  193.190.205.209:443         20     6279     1
2019-10-01 18:51:30.392     6.389 TCP    193.190.205.209:443   ->     192.168.0.14:42504       21     7486     1
2019-10-01 18:51:49.061     0.043 TCP      192.168.0.100:6690  ->     192.168.0.14:59660       31     9971     1
2019-10-01 18:51:49.059     0.045 TCP       192.168.0.14:59660 ->    192.168.0.100:6690        17     5497     1
2019-10-01 18:51:45.465     7.238 TCP       192.168.0.14:38968 ->      10.67.1.60:3128          4      240     1
2019-10-01 18:51:49.659     3.001 UDP       192.168.0.14:59044 ->  239.255.255.250:1900         4      800     1
2019-10-01 18:51:32.223    24.580 TCP       192.168.0.14:38112 ->    67.202.110.12:443          3      120     1
2019-10-01 18:51:55.528     3.031 TCP       192.168.0.14:38992 ->      10.67.1.60:3128          3      180     1
2019-10-01 18:51:32.327    24.576 TCP      67.202.110.12:443   ->     192.168.0.14:38112        4      206     1
2019-10-01 18:52:01.399     6.168 TCP    193.190.205.209:443   ->     192.168.0.14:42574       22     7538     1
2019-10-01 18:52:01.391     6.166 TCP       192.168.0.14:42574 ->  193.190.205.209:443         19     6227     1
2019-10-01 18:51:19.919    62.567 UDP       192.168.0.14:45062 ->     109.88.203.3:53           3      186     1
2019-10-01 18:51:19.935    62.568 UDP       109.88.203.3:53    ->     192.168.0.14:45062        3      618     1
2019-10-01 18:51:19.919    62.567 UDP       192.168.0.14:45062 ->    62.197.111.140:53          3      186     1
2019-10-01 18:51:19.919    62.567 UDP       192.168.0.14:45062 ->          8.8.8.8:53           3      186     1
2019-10-01 18:51:19.939    62.564 UDP            8.8.8.8:53    ->     192.168.0.14:45062        3      282     1
2019-10-01 18:51:54.333    28.169 UDP     62.197.111.140:53    ->     192.168.0.14:45062        2      412     1
2019-10-01 18:51:28.128    60.395 TCP       192.168.0.14:47118 ->     54.154.86.41:443         13      556     1
2019-10-01 18:52:28.442     0.000 UDP       192.168.0.14:46582 ->  239.255.255.250:1900         1      154     1
2019-10-01 18:51:28.162    60.361 TCP       54.154.86.41:443   ->     192.168.0.14:47118       12      887     1
2019-10-01 18:52:31.383     8.121 TCP       192.168.0.14:42636 ->  193.190.205.209:443         20     7115     1
```

Comparing a packet trace to a flow record

```
0.000000 192.168.1.201 -> 192.168.1.200   60 TCP  4105   80 1315338075 : 1315338075 0 win: 5840 S
0.003362 192.168.1.200 -> 192.168.1.201   60 TCP    80 4105 1417888236 : 1417888236 1315338076 win: 5792 SA
0.009183 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338076 : 1315338076 1417888237 win: 5840 A
0.010854 192.168.1.201 -> 192.168.1.200  127 TCP  4105   80 1315338076 : 1315338151 1417888237 win: 5840 PA
0.014309 192.168.1.200 -> 192.168.1.201   52 TCP    80 4105 1417888237 : 1417888237 1315338151 win: 5792 A
0.049848 192.168.1.200 -> 192.168.1.201 1500 TCP    80 4105 1417888237 : 1417889685 1315338151 win: 5792 A
0.056902 192.168.1.200 -> 192.168.1.201 1500 TCP   80  4105 1417889685 : 1417891133 1315338151 win: 5792 A
0.057284 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338151 : 1315338151 1417889685 win: 8688 A
0.060120 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338151 : 1315338151 1417891133 win: 11584 A
0.068579 192.168.1.200 -> 192.168.1.201 1500 TCP    80 4105 1417891133 : 1417892581 1315338151 win: 5792 PA
0.075673 192.168.1.200 -> 192.168.1.201 1500 TCP    80 4105 1417892581 : 1417894029 1315338151 win: 5792 A
0.076055 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338151 : 1315338151 1417892581 win: 14480 A
0.083233 192.168.1.200 -> 192.168.1.201 1500 TCP   80 4105 1417894029 : 1417895477 1315338151 win: 5792 A
0.096728 192.168.1.200 -> 192.168.1.201 1500 TCP   80 4105 1417896925 : 1417898373 1315338151 win: 5792 A
0.103439 192.168.1.200 -> 192.168.1.201 1500 TCP   80 4105 1417898373 : 1417899821 1315338151 win: 5792 A
0.103780 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338151 : 1315338151 1417894029 win: 17376 A
0.106534 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338151 : 1315338151 1417898373 win: 21720 A
0.133408 192.168.1.200 -> 192.168.1.201  776 TCP   80 4105 1417904165 : 1417904889 1315338151 win: 5792 FPA
0.139200 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338151 : 1315338151 1417904165 win: 21720 A
0.140447 192.168.1.201 -> 192.168.1.200   52 TCP  4105   80 1315338151 : 1315338151 1417904890 win: 21720 FA
0.144254 192.168.1.200 -> 192.168.1.201   52 TCP   80 4105 1417904890 : 1417904890 1315338152 win: 5792 A
```

Flow summary (e.g., NetFlow record or Bro connection log entry):

0.000000 192.168.1.201 4105 192.168.1.200 80 0.144254 10 77 11 16654 SF

Flow data in the cloud

- Cloud computing infrastructure (e.g., AWS) allows collection of flow logs
- AWX VPC Flow logs provide flow summaries similar to Netflow
  - A key distinction is the VPC Flow logs summarize network activity for a specific “virtual private cloud”
  - Activity from multiple VPCs could potentially traverse the same Netflow enabled device
  - In other words, VPC Flow logs are similar to network logs that have been demultiplexed

Encrypted vs Unencrypted

- Use of TLS for encrypting network data has been increasing
- TLS encrypts transport layer packet payload
- So, application layer protocols are encrypted, but network (IP) and transport (TCP/UDP) layer headers are not
- Netflow is unaffected

Network Traffic Data Formats

- Packets (e.g., pcap)
- Flows
  - Unidirectional (e.g., netflow)
  - Bidirectional (e.g., IDS (zeek), firewall (check point))
- Sampled flows (e.g., s-flow)

Data Sets

How to create a network traffic data set with attacks?

- Real
  - + it is real
  - – privacy issues
  - – very few attacks
  - – no labeling
- Emulated
  - – not real
  - + no privacy issues
  - + has attacks
  - + labels
- Hybrid
  - ± partially real
  - – privacy issues
  - + has attacks
  - ± partial labels
- Synthetic
  - – not real
  - + no privacy issues
  - + has attacks
  - + labels

How to label a data set?

- Through emulation
- Use rules
- Use a ML model
- Synthetic
- Automated labeling

Other data set considerations

- Duration and size
- Packets vs Flows
- Meta information
- Anonymized fields
- Unidirectional vs bidirectional flows
- Class imbalance

Examples of publicly available network traffic data sets

- NSL-KDD [Tavallaee et al. 2009]
- University of New Brunswick, CICIDS [Sharafaldin et al. 2018]
- Australian Centre for Cyber Security, UNSW-NB [Moustafa et al. 2015]
- University of Coburg, CIDDS-002 [Ring et al. 2017]
- CTU-13 [Garcia et al. 2014]
- UGR-16 [Maciá-Fernández et al. 2018]

Ring et al., A Survey of Network-based Intrusion Detection Data Sets, https://arxiv.org/pdf/1903.02460.pdf

Applications

Why is network traffic data collected?

- Network Management perspective
  - To understand the traffic on existing networks
  - For workload characterization and modeling
  - To guide the design of future networks
  - For performance evaluation of network protocols and applications
  - For debugging network protocols and applications

- Security perspective
  - For network security monitoring
  - For threat hunts and investigations

A lot of these can be cast as ML problems

- Intrusion detection and prevention
- Malware detection and prevention
- Other attack detection, e.g., DOS
- Search, e.g., find similar attacks
- Device Identification
- Traffic classification
- Application identification
- Website fingerprinting

Challenges

Challenges

- Lack of data sets that are
  - publicly available
  - representative
  - Labeled – requires both effort and expertise
- Privacy and security
- High volume
  - granularity

Challenges

- Complex
    - Difficult to understand
        - Due to encryption, virtualization, containerization, NAT’s, etc.
    - Heterogeneous
    - Temporal and within-row dependencies
    - Long-tailed
    - Non-stationarity
- Adversary
- Validation

References

- Sommer, Robin, and Vern Paxson. "Outside the closed world: On using machine learning for network intrusion detection." *2010 IEEE symposium on security and privacy*. IEEE, 2010.
- Anderson, Blake, and David McGrew. "Machine learning for encrypted malware traffic classification: accounting for noisy labels and non-stationarity." *Proceedings of the 23rd ACM SIGKDD International Conference on knowledge discovery and data mining*. 2017.
- Benoît Claise. 2004. Cisco Systems NetFlow Services Export Version 9. RFC 3954.
- TCAP, tcpdump.org
- Ring, Markus, et al. "A survey of network-based intrusion detection data sets." Computers & Security 86 (2019): 147-167.
- Maciá-Fernández, Gabriel, et al. "UGR ‘16: A new dataset for the evaluation of cyclostationarity-based network IDSs." Computers & Security 73 (2018): 411-424.
- Garcia, Sebastian, et al. "An empirical comparison of botnet detection methods." computers & security 45 (2014): 100-123.
- Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward generating a new intrusion detection dataset and intrusion traffic characterization." ICISSP 1 (2018): 108-116.
- Tavallaee, Mahbod, et al. "A detailed analysis of the KDD CUP 99 data set." 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, 2009.
- Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." 2015 military communications and information systems conference (MilCIS). IEEE, 2015.
- Ring, Markus, et al. "Creation of flow-based data sets for intrusion detection." Journal of Information Warfare 16.4 (2017): 41-54.

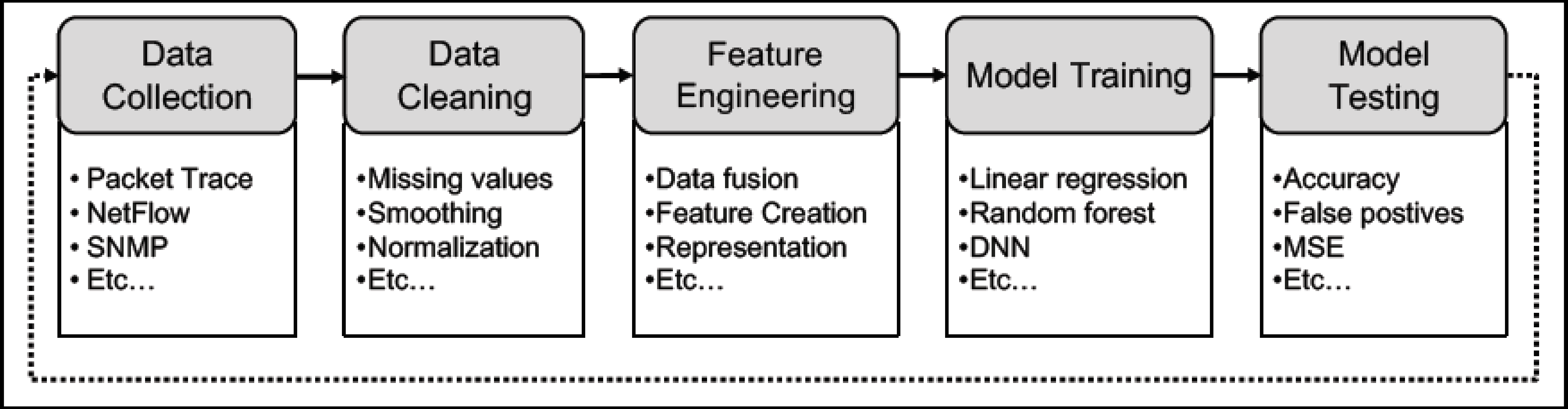
Part 2: Network Data Representation Learning

Manish Marwah, Martin Arlitt

Part 2 Outline

- Feature engineering
- Representation Learning
- Representation learning for IP address, port number
- Traffic as an image
- Traffic as a sequence
- Traffic as a graph

Typical Approach



[Bronzino et al 22]

Basic Features: Header Fields of Protocols

Protocol Fields	Fields used as Basic Features
802.11 headers	version, type, subtype, ToDS, FromDS, More Fragments, Retry, Power Mgmt, More Data, WEP, Order, Duration, RA, TA, MA, FCS
802.11 Calculated	isWepValid: flag indicating if WEP ICV check was successful DurationRange: discretize numerical duration to values low, average or high CastingType: unicast, multicast or broadcast destination address
Ethernet headers	size, dest hi, dest lo, src hi, src lo, protocol
IP headers	header length, TOS, Frag ID, Frag Ptr, TTL, Protocol, Checksum, Src ip, Dest ip
TCP headers	Src Port, Dest Port, Seq, Ack, Header Len, Flag UAPRSF, Window Sz, Checksum, URG Ptr, Option
UDP headers	Src Port, Dest Port, Len, Checksum
ICMP headers	Type, Code Checksum

[Davis et al 2011]

Examples of Features derived from a single flow / connection

Feature	Description
Contextual	Quad: combination of src ip, src port, dst ip, dst port define a single connection Service type (TCP, UDP or ICMP) and application protocol (HTTP, SMTP, SSH or FTP etc.) group similar traffic
Duration	Start time, end time, and the duration of the connection
Status	Normal or error status of connection e.g. valid TCP 3-way handshake, and FIN to end session
SCD timing	Number of questions per second Average size of questions Average size of answers Question answer idle time Answer question idle time
RTT	The round trip time (RTT) of a TCP packet sent through a connection chain is calculated from timestamps of TCP send and echo packets
Fingerprint	Percentage of packets with each of the TCP flags set Mean packet inter-arrival time Mean packet length Number of bytes and number of packets in connection Union of TCP flags
HTTPS session	For HTTPS traffic, a feature vector is created as a set of data sizes transferred during the session. The 10 largest values of request size and response size in the HTTPS session are used
TCP Flags	Each TCP flag combination is quantized as a symbol. A TCP session is then represented as a sequence of symbols, one symbol per packet transferred
TCP states	Create a feature vector listing the frequency of each TCP state transition
land	1 if src ip and port matches dest ip and port. 0 otherwise
wrong_fragment	Number of wrong fragments
urgent	Number of urgent packets

[Davis et al 2011]

Examples of features derived from a multiple flows / connections

Feature	Description
Joint Probability	$P(srcip, srcport, dstip, dstport)$, $P(srcip, dstip, dstport)$, $P(dstip, dstport)$
Conditional Probability	$P(srcIP\|dstIP)$, $P(srcIP\|dstIP, dstport)$, $P(TCPflags\|destport)$, $and P(keyword\|destport)$
Entropy Measures	Entropy of basic features over dataset: src ip, src port, dst ip, dst port
Association Rules	Mine rules from connection records containing: start time, quad, and connection status
Flow concentration	Count of TCP flows with same src ip, dst ip and dst port in this time slice
Data points per cluster	A cluster is a frequently occurring value for a feature, e.g. a common IP address
%Control, %Data	percentage of control/data packets
Av Duration	Average flow duration over all flows
Av Duration Dest	Average flow duration per destination
Max Flows	Maximum number of flows to a particular service
%_same_service_host	Percent of traffic from a particular src port to a particular dst ip
%_same_host_service	Percent of traffic from a particular src ip to a particular dst port
Count Variance	Variance measure for the count of packets for each src-dest pair
Wrong resent rate	Count of bytes sent event after being acknowledged
Duplicate ACK rate	Count of duplicate acknowledgment packets
Data bytes	Count of data bytes exchanged per flow
sdp statistics	Source-destination pairs (sdps) are unique combinations of src ip, dest ip and dest port Number of unique sdps in collection interval Number of new sdps in this data collection interval Number of new sdps which were not seen in last month Number of well known ports used in interval Variance of the count of packets seen against sdps Count of sdps which include hosts outside local network domain Number of successfully established TCP connections in time interval Total packets observed in collection interval
av_size	Average packet size over time window
av_packets	Average packets per flow over time window
count_single	Number of single packet flows over time window
ratio	Ratio of Number of flows to bytes per packet (TCP) over time window

[Davis et al 2011]

Examples of volume / count features

Volume-based Feature	Description
count-dest	Flow count to unique dest IP in the last T seconds from the same src
count-src	Flow count from unique src IP in the last T seconds to the same dest
count-serv-src	Flow count from the src IP to the same dest port in the last T seconds
count-serv-dest	Flow count to the dest IP using same src port in the last T seconds
count-dest-conn	Flow count to unique dest IP in the last N flows from the same src IP
count-src-conn	Flow count from unique src IP in the last N flows to the same dest IP
count-serv-src-conn	Flow count from the src IP to the same dest port in the last N flows
count-serv-dest-conn	Flow count to the dest IP using same source port in the last N flows
num_packets_src_dst/dst_src	Count of packets flowing in each direction
num_acks_src_dst/dst_src	Count of acknowledgment packets flowing in each direction
num_bytes_src_dst/dst_src	Count of data bytes flowing in each direction
num_retransmit_src_dst/dst_src	Count of retransmitted packets flowing in each direction
num_pushed_src_dst/dst_src	Count of pushed packets flowing in each direction
num_SYNs(FINs)_src_dst/dst_src	Count of SYN/FYN packets flowing in each direction
connection_status	Status of the connection (0 Completed; 1 - Not completed; 2 Reset)
count_src'	Connection count from same source as the current record
count_serv_src	Count of different services from the same source as the current record
count_serv_dest	Count of different services to the same destination IP as the current record
count_src_conn	Connection count from this src IP in the last 100 connections
count_dest_conn	Connection count to this dest IP in the last 100 connections
count_serv_src_conn	Connection count with same dst port and src IP in the last 100 connections
count_serv_dst_conn	Connection count with same dst port and dst IP in the last 100 connections

[Davis et al 2011]

Feature engineering is difficult

- Manual and laborious
- Requires deep domain knowledge
- Based on extensive trial and error
- Subjective (introduces bias of the domain expert – could be good or bad)

Feature Learning in Computer Vision

Traditional approach to image classification

[224x224x3]

Manual

Handcrafted features

GIST
HoG
SIFT
…

Learned

ML Classfier

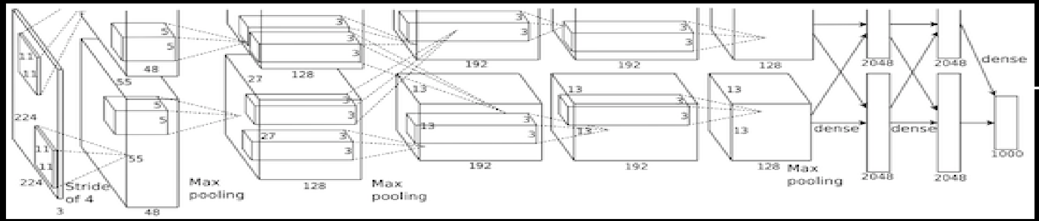
Shallow NN
SVM
Random Forest
…

“Cat”

Deep learning approach to image classification

[224x224x3]

Learned



Deep neural network

“Cat”

Is domain information necessary?

Tabula Rasa

Domain
Expert Rules

Where does the sweet spot lie?

How to use an IP address as a feature?

- Ignore it
  - May not be available
  - Usually anonymized
  - Use other features
- Indirect features
  - Internal / external
  - Various counts
- binarize, extract 32 features (IPv4)

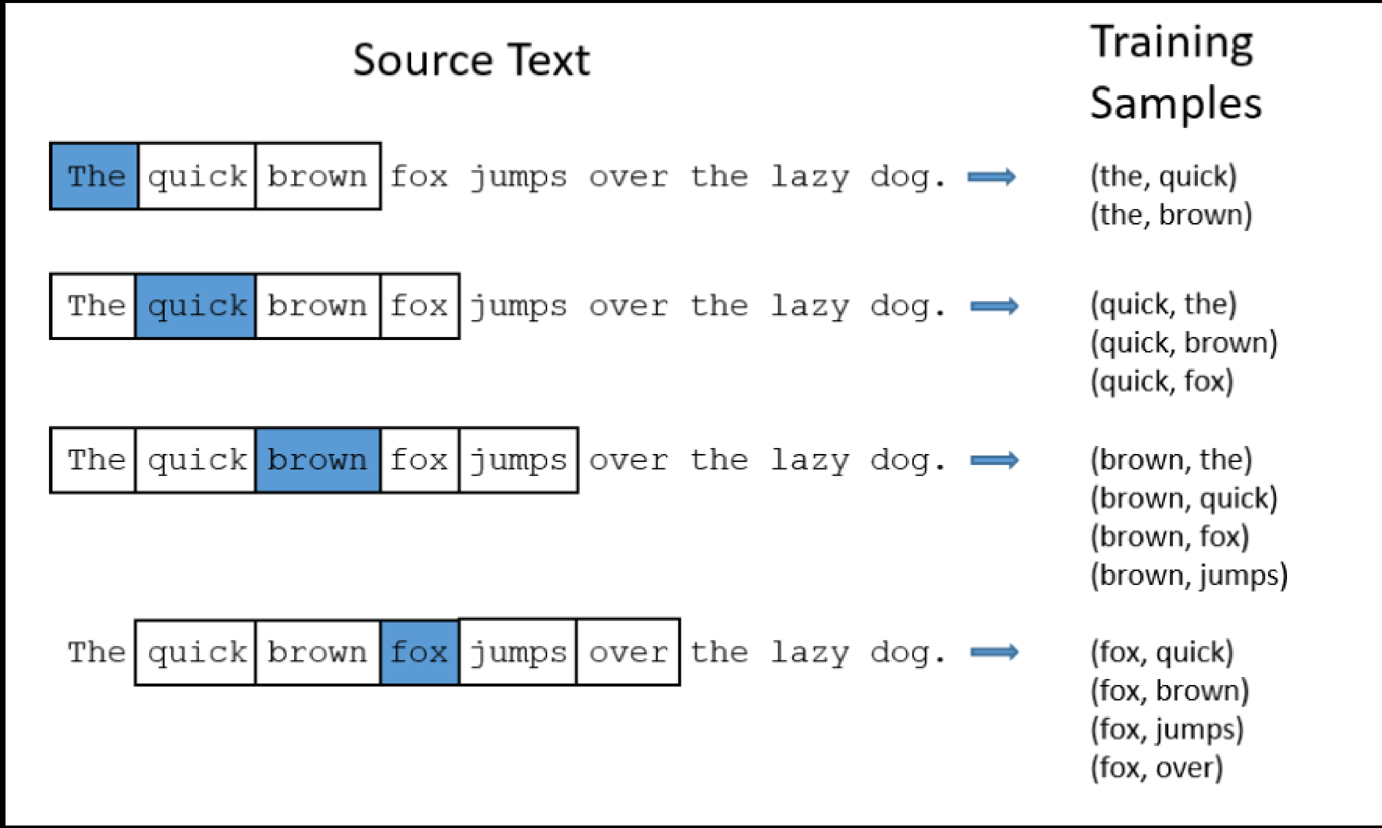
- convert to geo locations and use longitude, latitude

- define a hierarchy of IP addresses, compute distance based on this tree

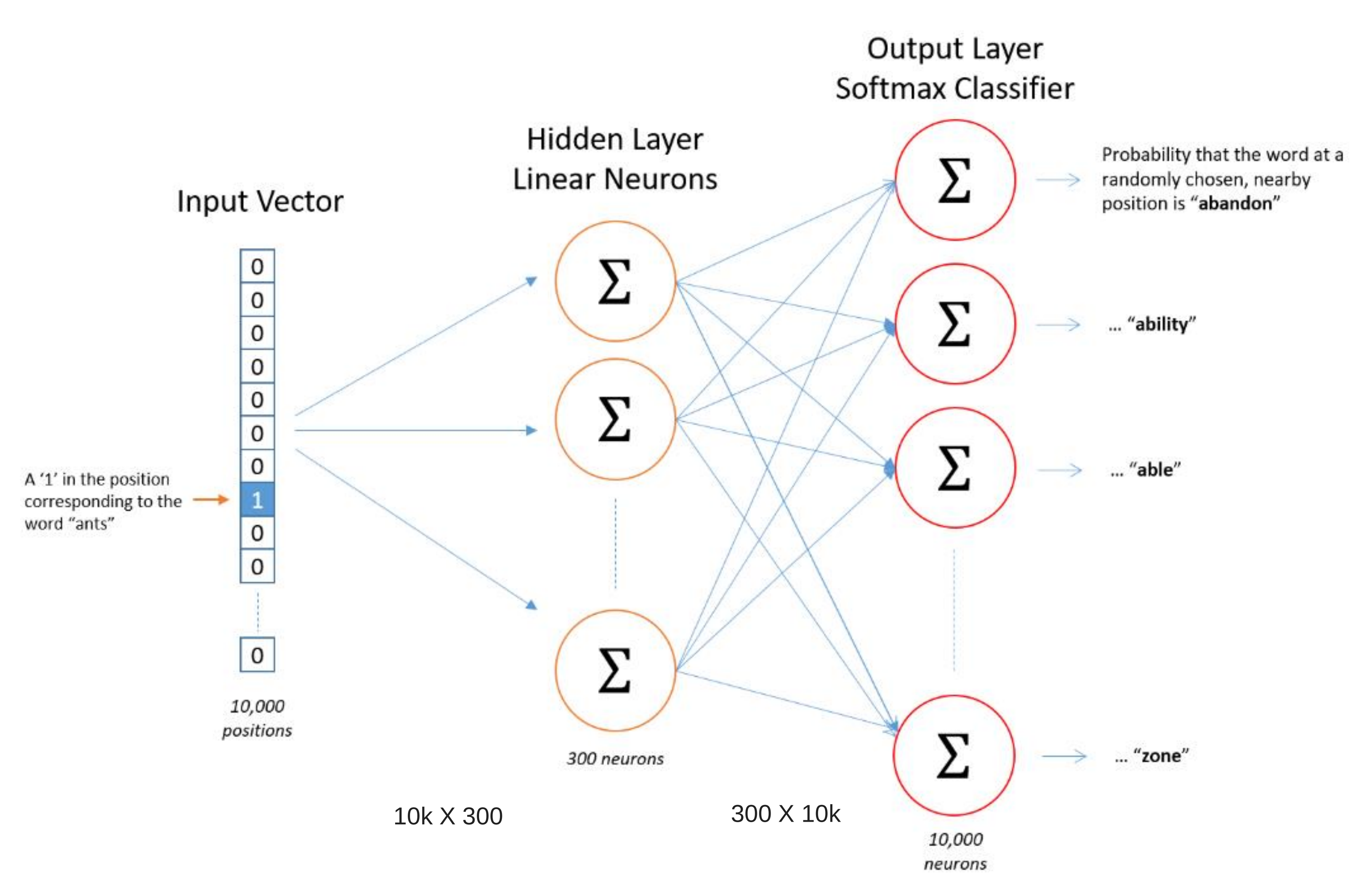
Can we learn features with minimal domain knowledge?

Learning Representation of IP addresses

word2vec Recap

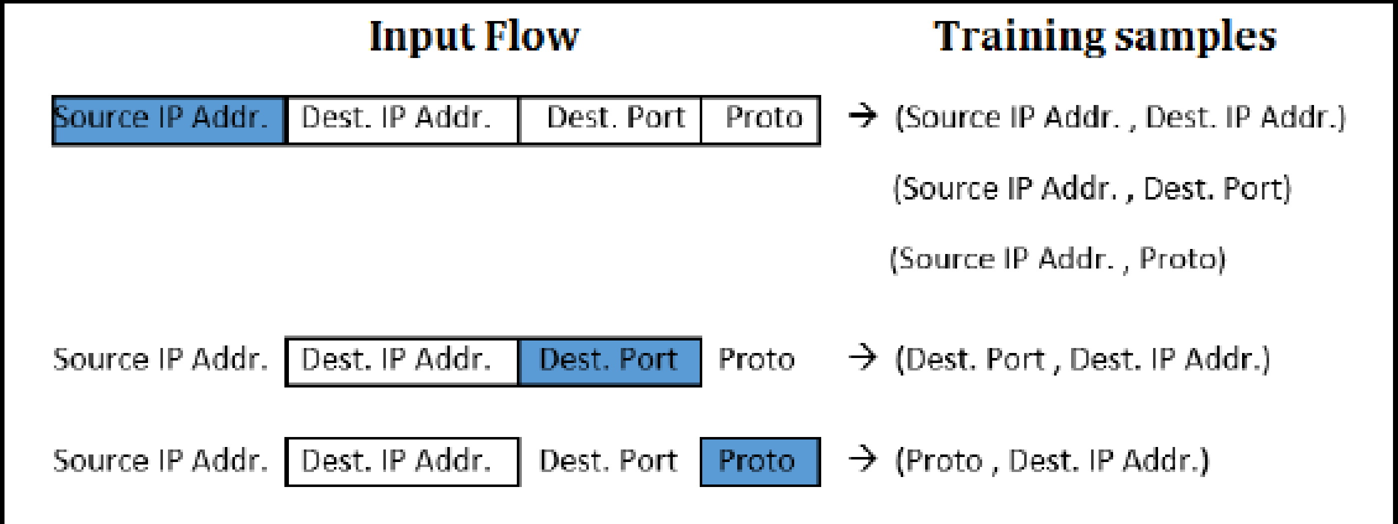


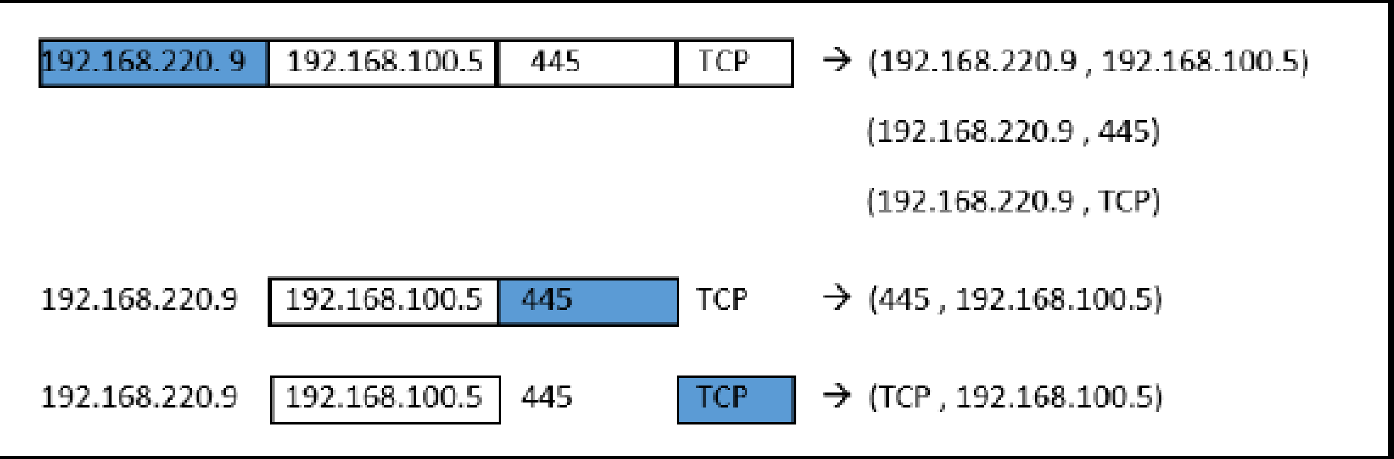
http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/



http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

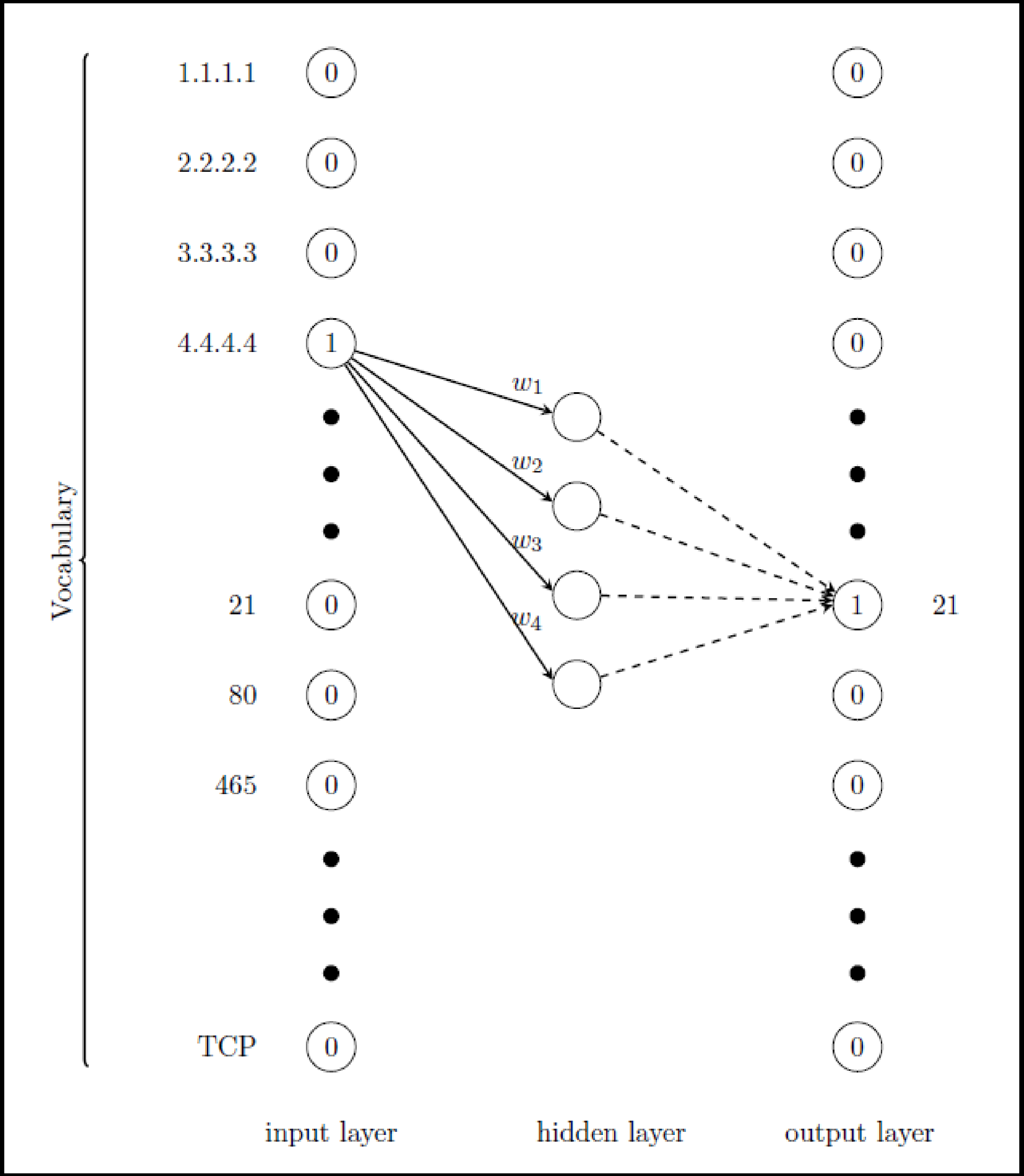
#	Source IP Addr.	Dest. IP Addr.	Dest. Port	Proto.
1	192.168.100.5	192.168.220.9	51479	TCP
2	192.168.220.9	192.168.100.5	445	TCP
3	216.58.210.19	192.168.200.8	44444	TCP
4	192.168.200.8	216.58.210.19	80	TCP
5	192.168.220.14	53.53.53.53	53	UDP





[Ring et al 2017]

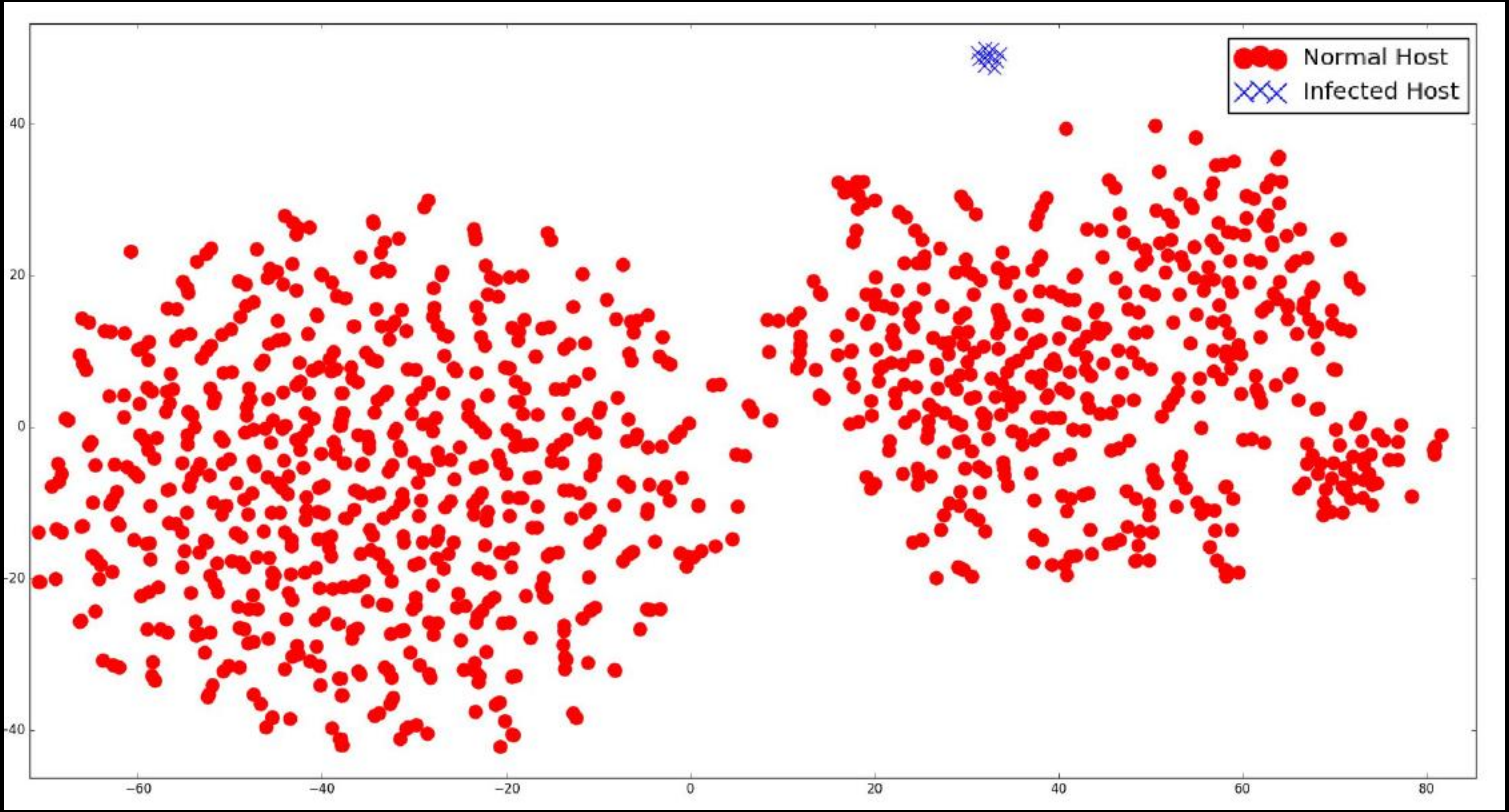
IP2Vec



Differences from work2vec

- Only a subset of inputs are used
- And on a subset of features are used as context
- Non-stationarity

[Ring et al 2017]



t-sne of embedding of IP addresses on CTU-13 data set

[Ring et al 2017]

Packet-based traffic data

- Direct application of word2vec to payload data [packet2vec, Goodman et al. 2020]
- TLS2vec [Ferriyan, 2022]

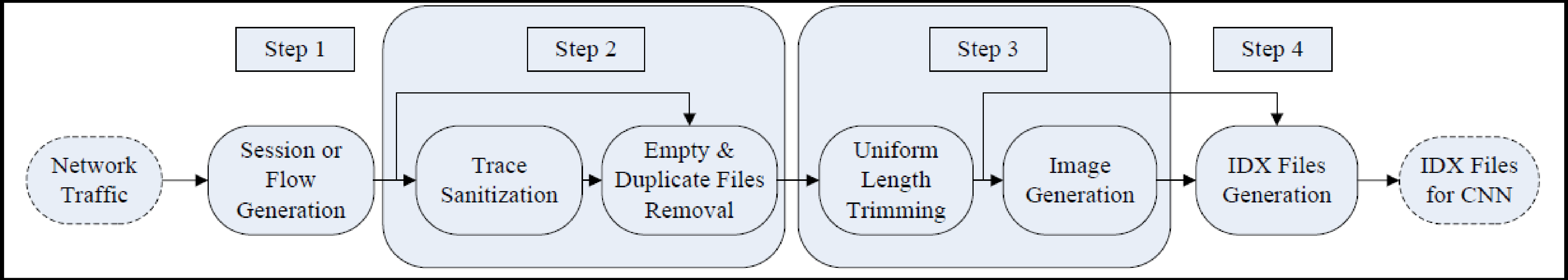
Traffic as an Image

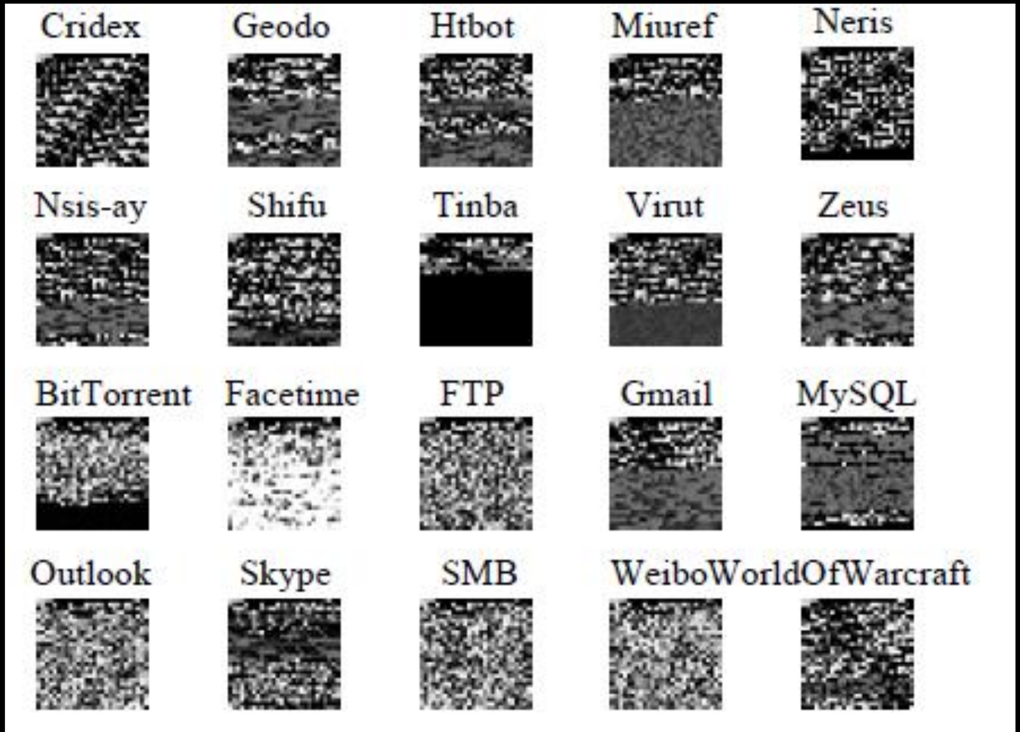
Traffic as an image

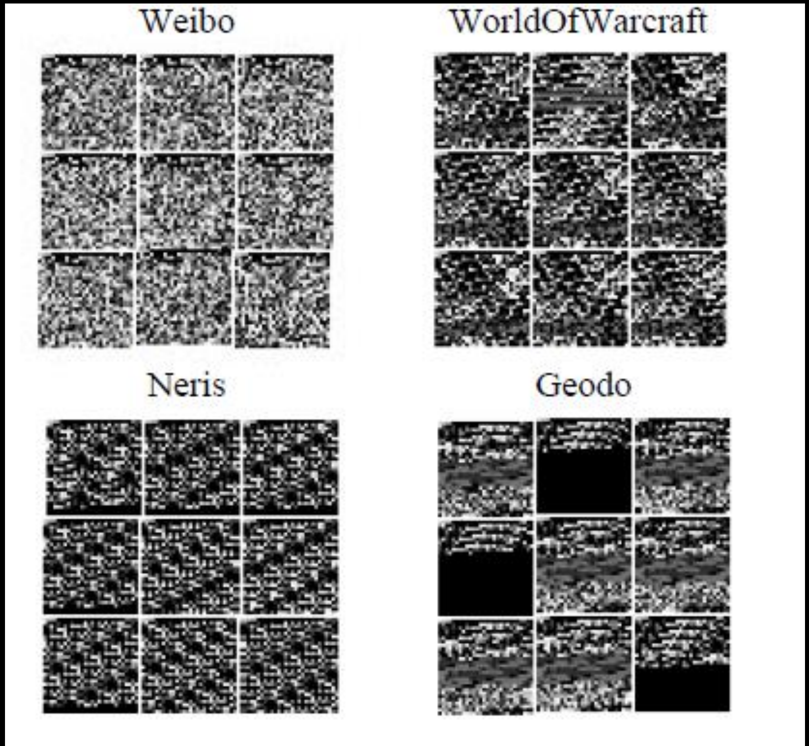
- Several researchers have proposed converting network traffic to images
- Uses raw packet or flow data or extracted features
- Mostly supervised, assuming availability of labeled data
- CNN-based architectures are used
- Temporal dependency is typically ignored

Malware Traffic Classification Using CNN for Representation Learning [Wang et al. 2017]

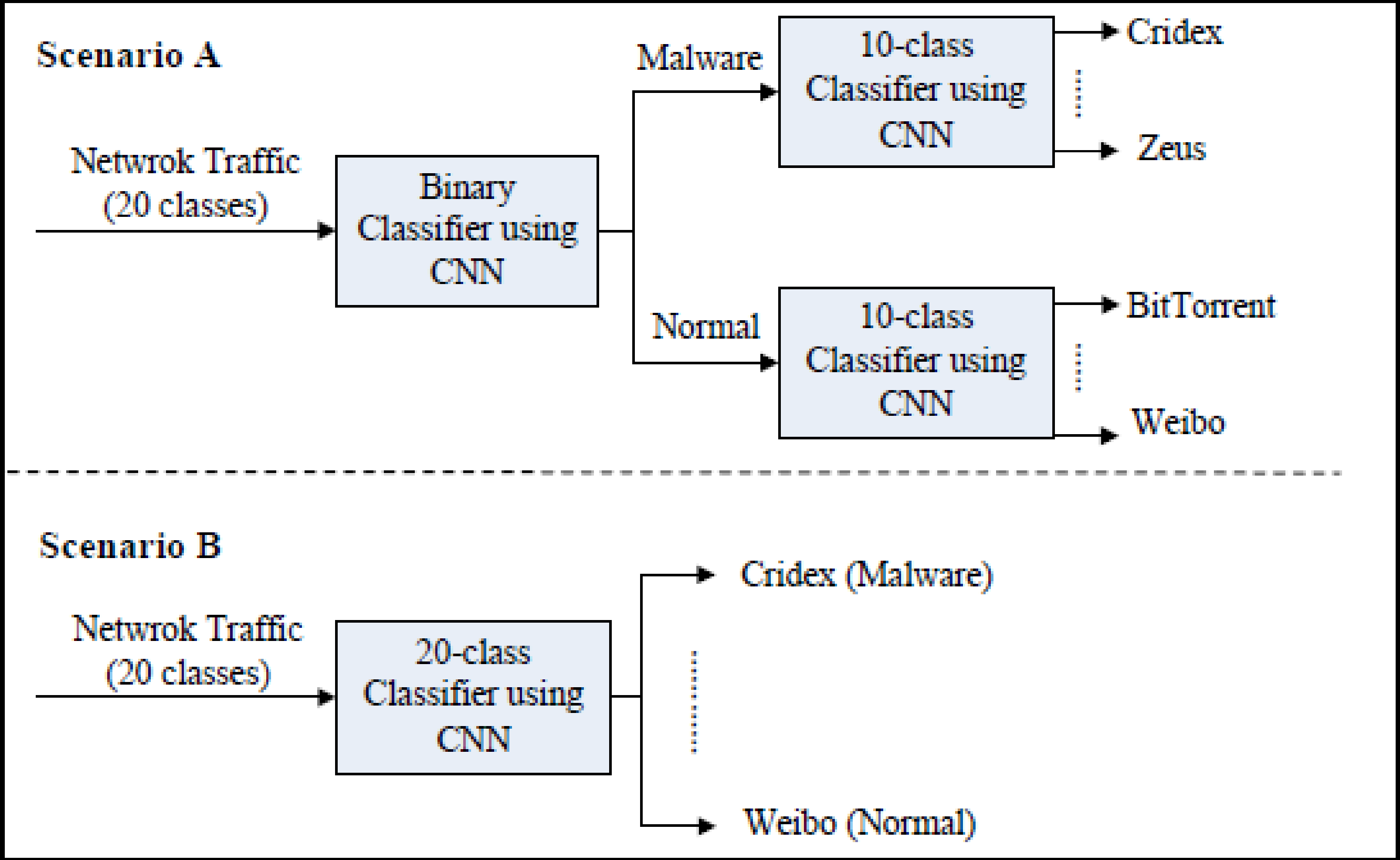
- Detect known malware from network packet data
- Created labeled data set with 10 types of malware traffic and 10 types of normal traffic





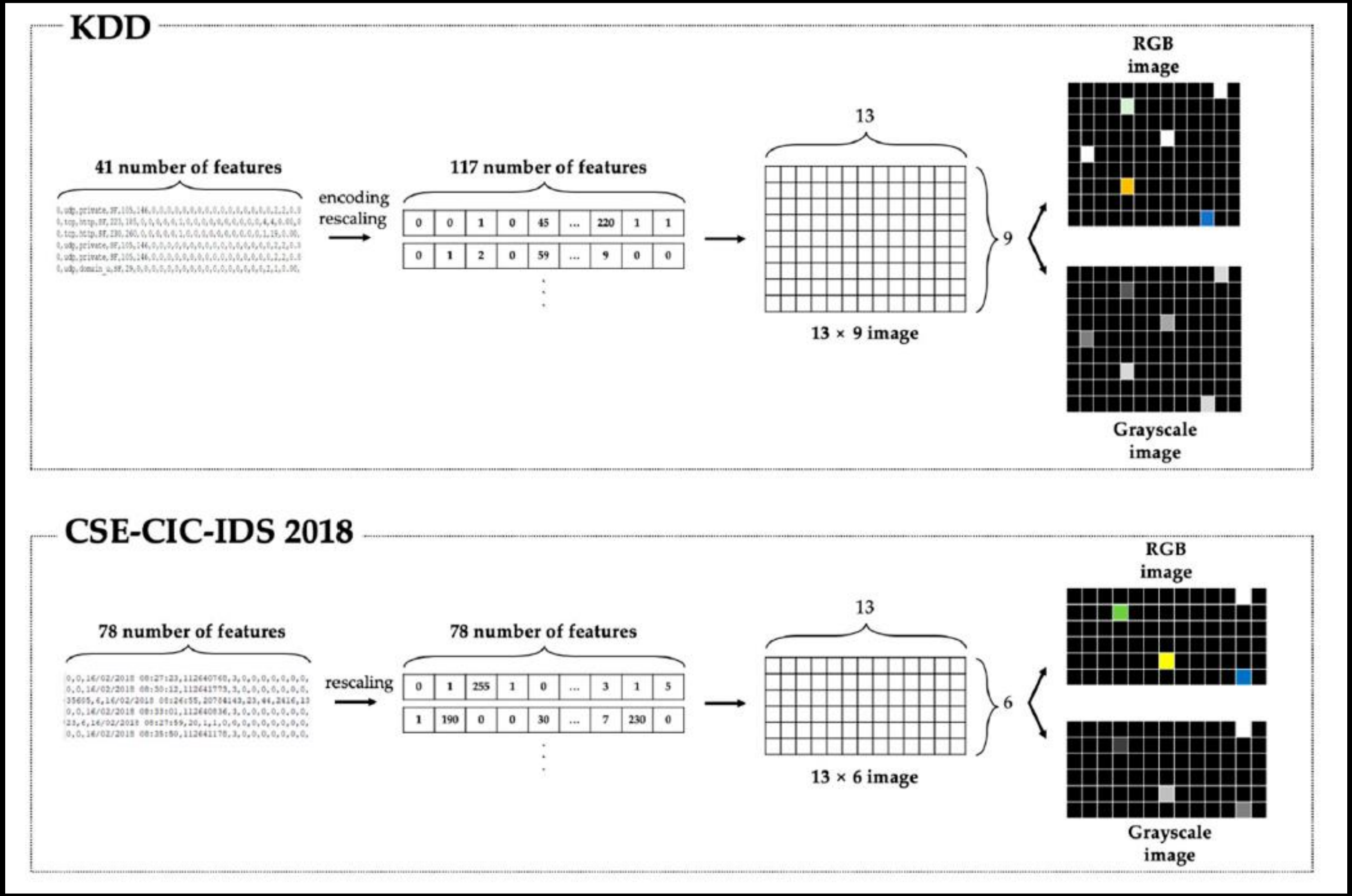


[Wang et al. 2017]



[Wang et al. 2017]

CNN-based network intrusion detection against denial-of-service attacks [Kim et al. 2020]



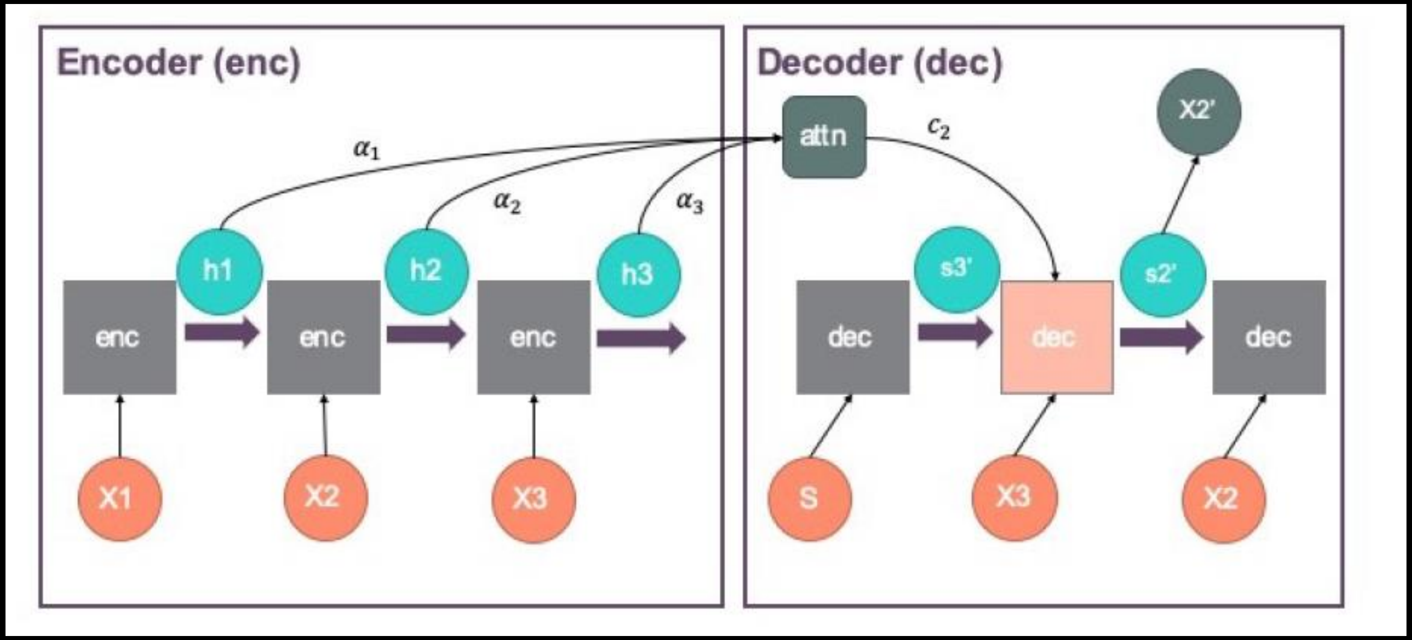
Traffic as a Sequence

Traffic as a sequence

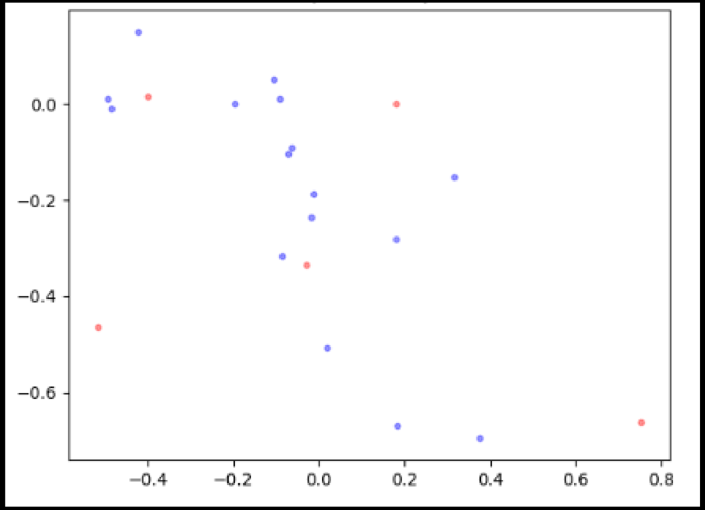
- Flow data can be considered as a multivariate, temporal sequence
- Several researchers have used LSTM / Transformer based architectures
- Unsupervised / self-supervised anomaly detection

Attention-based self-supervised feature learning for security data [Lee et al. 2020]

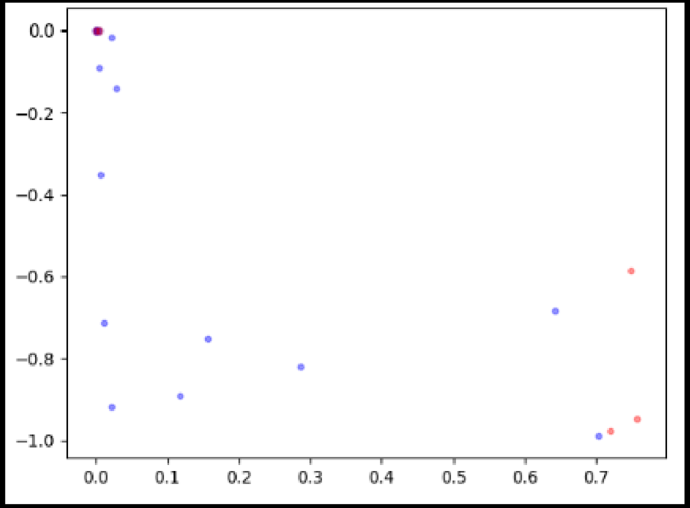
- Flows are extracted for users (IP's) over time windows and clustered
- Sequence to sequence encoder-decoder with attention



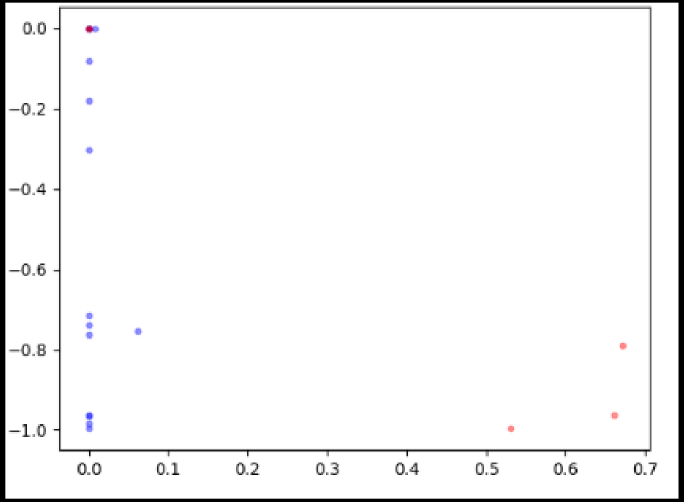
Evidence of representation learning during training



Epoch: 1



Epoch: 15



Epoch: 24

[Lee et al. 2020]

Traffic as a Graph

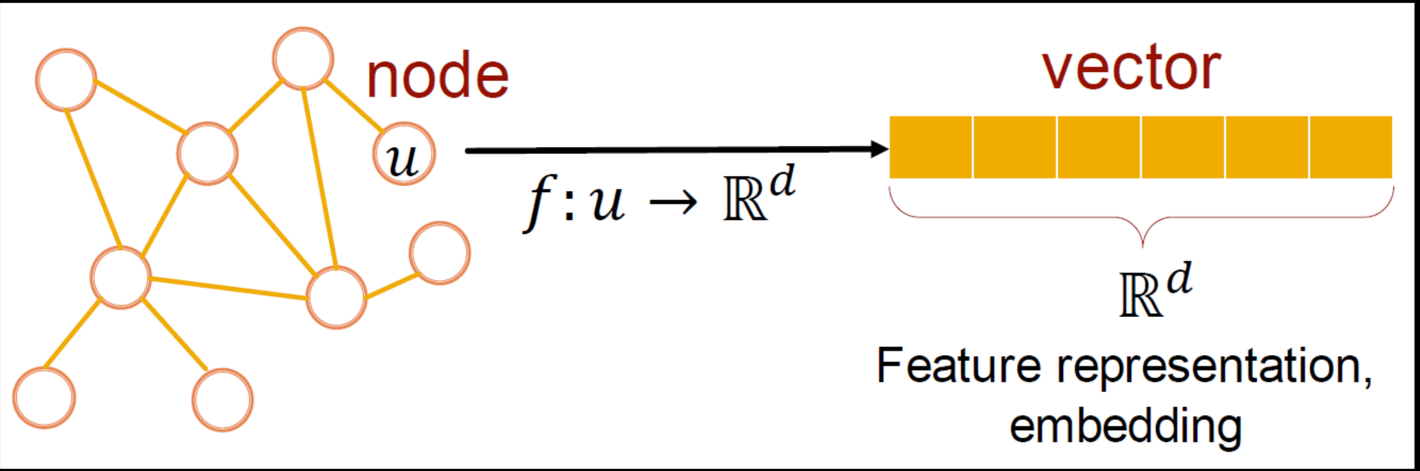
Traffic as a graph

- Network traffic data intrinsically exhibits a graph structure
- With machines/users as nodes and traffic between them as edges
- Lot of work on extracting graph-based features from network traffic data (e.g., graph-based anomaly detection [Akoglu et al. 2015])
    - Node level features
    - Link (edge) level features
    - Graph level features

- Can we learn graph-based features instead?

Graph Neural Networks

- Replace graph feature engineering with learning embeddings
- Learn embeddings for nodes, graphs, edges



Advantages

- Embedding encode semantics
- Similarity in embeddings of two nodes imply their similarity in the graph
- Can be used for any downstream ML task

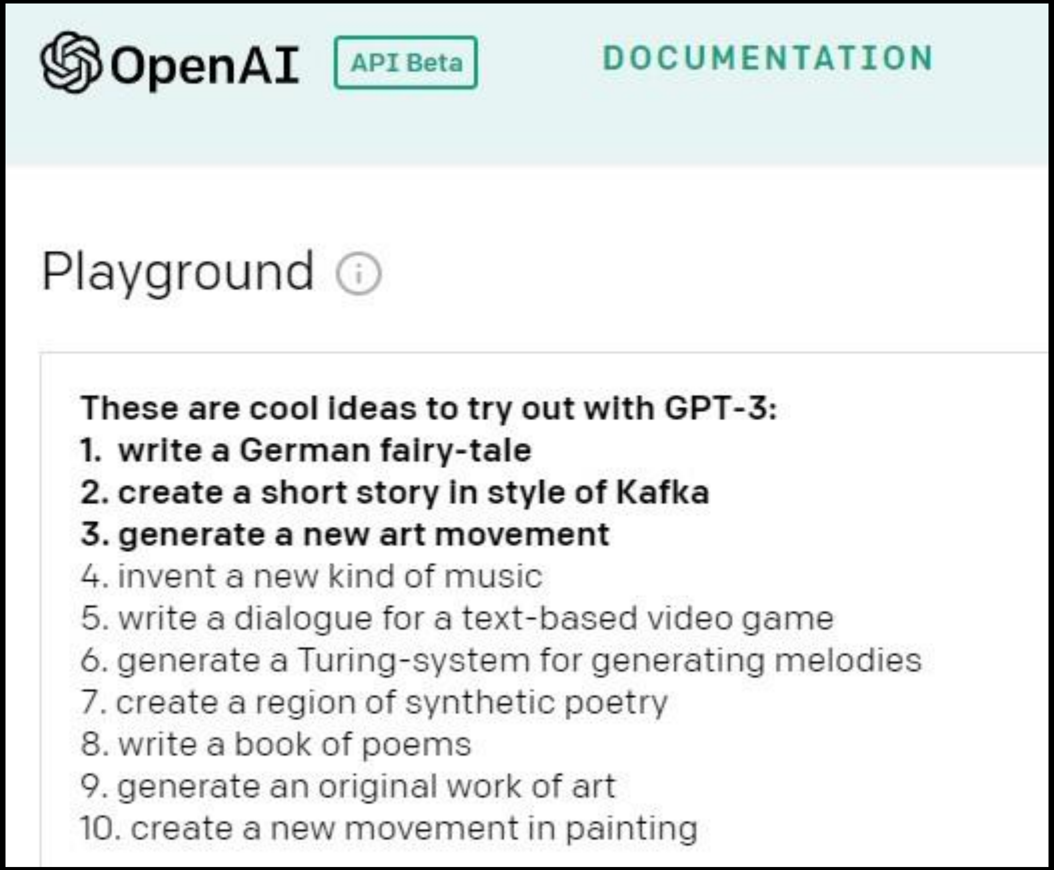
References

- Francesco Bronzino, Paul Schmitt, Sara Ayoubi, Hyojoon Kim, Renata Teixeira, Nick Feamster (2022). Traffic Refinery: Cost-Aware Data Representation for Machine Learning on Network Traffic. In ACM SIGMETRICS, pp. 1–12, Mumbai, India.
- Davis, Jonathan J., and Andrew J. Clark. "Data preprocessing for anomaly based network intrusion detection: A review." computers & security 30.6-7 (2011): 353-375.
- Ring, Markus, et al. "Ip2vec: Learning similarities between ip addresses." 2017 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, 2017.
- Goodman, Eric L., Chase Zimmerman, and Corey Hudson. "Packet2vec: Utilizing word2vec for feature extraction in packet data." arXiv preprint arXiv:2004.14477 (2020).
- Ferriyan, Andrey, et al. "Encrypted Malicious Traffic Detection Based on Word2Vec." Electronics 11.5 (2022): 679.
- Wang, Wei, et al. "Malware traffic classification using convolutional neural network for representation learning." 2017 International conference on information networking (ICOIN). IEEE, 2017.
- Kim, Jiyeon, et al. "CNN-based network intrusion detection against denial-of-service attacks." Electronics 9.6 (2020): 916.
- Lee, I-Ta, Manish Marwah, and Martin Arlitt. "Attention-Based Self-Supervised Feature Learning for Security Data." arXiv preprint arXiv:2003.10639 (2020).
- Akoglu, Leman, Hanghang Tong, and Danai Koutra. "Graph based anomaly detection and description: a survey." Data mining and knowledge discovery 29.3 (2015): 626-688.
- Wu, Kehe, Zuge Chen, and Wei Li. "A novel intrusion detection model for a massive network using convolutional neural networks." Ieee Access 6 (2018): 50850-50859.
- Kim, Jiyeon, et al. "CNN-based network intrusion detection against denial-of-service attacks." Electronics 9.6 (2020): 916

Part 3: Synthetic Network Traffic Generation

Generative Models

https://thispersondoesnotexist.com/image



https://towardsdatascience.com/20-creative-things-to-try-out-with-gpt-3-2aacee3e2abf

Generative methods

- Learn P(x, y) instead of just P(y | x)

- Kernel density estimation
- Probabilistic Graphical models
  - Bayesian networks
- …

- Generative adversarial network (GAN)
- Autoregressive neural model
- Variational autoencoder
- …

Evaluation

- Once you generate synthetic data, how can it be validated?

Compare with real data:

- Marginal distributions
- Conditional distributions
- Pairwise correlations
- Mutual information

Use it for tasks that real data is used

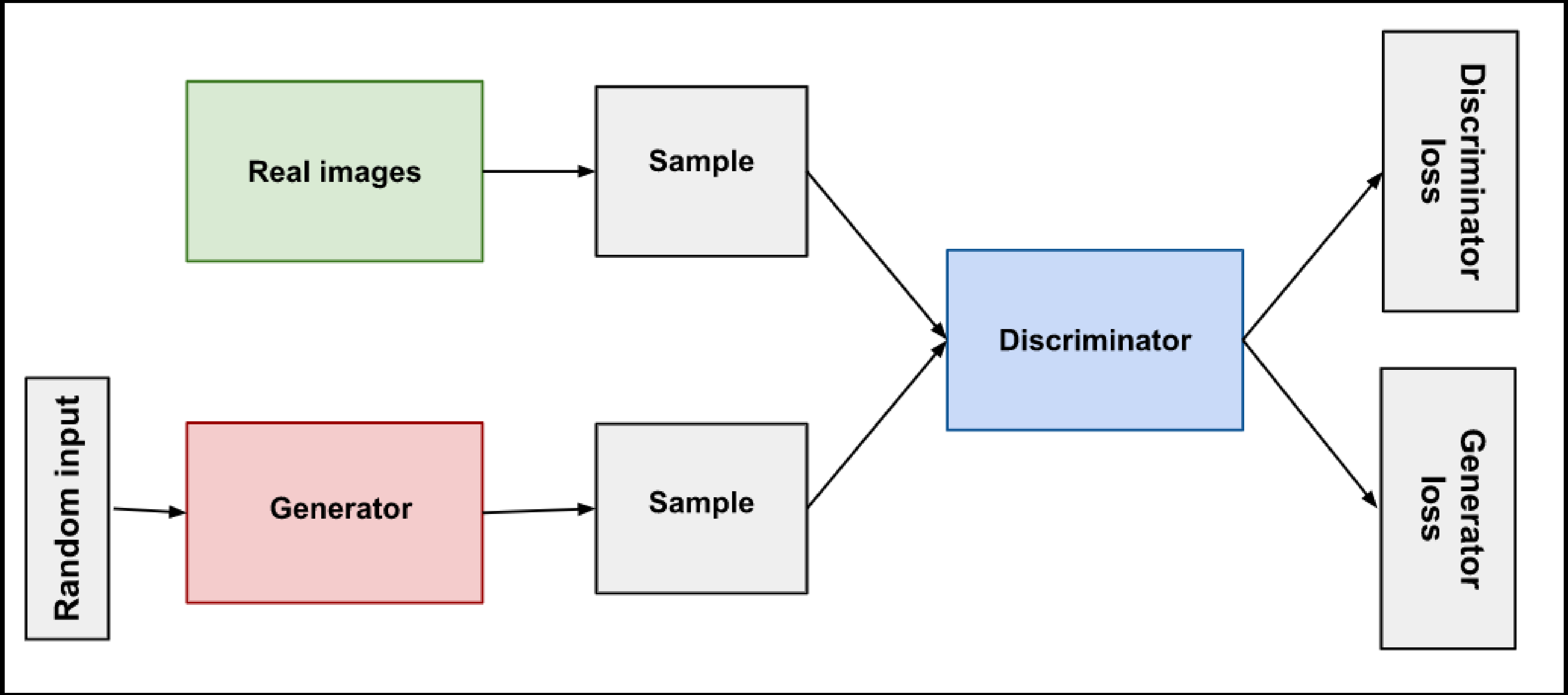
- Train ML models
- Compare performance on real data

How do you learn to generate, say, netflow data?

Terminal File Edit View Search Terminal Help

```
$ nfdump -R /var/cache/nfdump/
Date first seen          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2019-10-01 18:51:06.726     0.013 TCP    193.190.205.209:443   ->     192.168.0.14:42438        3      187     1
2019-10-01 18:51:06.726     0.002 TCP       192.168.0.14:42438 ->  193.190.205.209:443          2      104     1
2019-10-01 18:51:15.843     0.000 UDP      192.168.0.100:5353  ->      224.0.0.251:5353         1      155     1
2019-10-01 18:51:08.044     7.730 UDP       192.168.0.14:5353  ->      224.0.0.251:5353         2      162     1
2019-10-01 18:51:30.381     6.389 TCP       192.168.0.14:42504 ->  193.190.205.209:443         20     6279     1
2019-10-01 18:51:30.392     6.389 TCP    193.190.205.209:443   ->     192.168.0.14:42504       21     7486     1
2019-10-01 18:51:49.061     0.043 TCP      192.168.0.100:6690  ->     192.168.0.14:59660       31     9971     1
2019-10-01 18:51:49.059     0.045 TCP       192.168.0.14:59660 ->    192.168.0.100:6690        17     5497     1
2019-10-01 18:51:45.465     7.238 TCP       192.168.0.14:38968 ->       10.67.1.60:3128         4      240     1
2019-10-01 18:51:49.659     3.001 UDP       192.168.0.14:59044 ->  239.255.255.250:1900         4      800     1
2019-10-01 18:51:32.223    24.580 TCP       192.168.0.14:38112 ->    67.202.110.12:443          3      120     1
2019-10-01 18:51:55.528     3.031 TCP       192.168.0.14:38992 ->       10.67.1.60:3128         3      180     1
2019-10-01 18:51:32.327    24.576 TCP      67.202.110.12:443   ->     192.168.0.14:38112        4      206     1
2019-10-01 18:52:01.399     6.168 TCP    193.190.205.209:443   ->     192.168.0.14:42574       22     7538     1
2019-10-01 18:52:01.391     6.166 TCP       192.168.0.14:42574 ->  193.190.205.209:443         19     6227     1
2019-10-01 18:51:19.919    62.567 UDP       192.168.0.14:45062 ->      109.88.203.3:53          3      186     1
2019-10-01 18:51:19.935    62.568 UDP       109.88.203.3:53    ->     192.168.0.14:45062        3      618     1
2019-10-01 18:51:19.919    62.567 UDP       192.168.0.14:45062 ->    62.197.111.140:53          3      186     1
2019-10-01 18:51:19.919    62.567 UDP       192.168.0.14:45062 ->          8.8.8.8:53          3      186     1
2019-10-01 18:51:19.939    62.564 UDP            8.8.8.8:53    ->     192.168.0.14:45062        3      282     1
2019-10-01 18:51:54.333    28.169 UDP     62.197.111.140:53    ->     192.168.0.14:45062        2      412     1
2019-10-01 18:51:28.128    60.395 TCP       192.168.0.14:47118 ->     54.154.86.41:443         13      556     1
2019-10-01 18:52:28.442     0.000 UDP       192.168.0.14:46582 ->  239.255.255.250:1900         1      154     1
2019-10-01 18:51:28.162    60.361 TCP       54.154.86.41:443   ->     192.168.0.14:47118       12      887     1
2019-10-01 18:52:31.383     8.121 TCP       192.168.0.14:42636 ->  193.190.205.209:443         20     7115     1
```

Generative Adversarial Network (GAN)

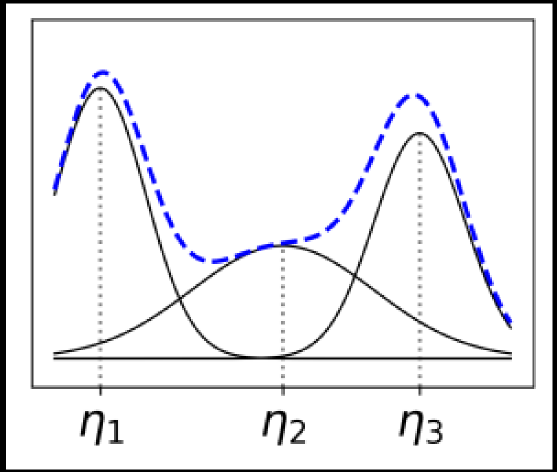


https://developers.google.com/machine-learning/gan/gan_structure

Modeling Tabular Data using Conditional GAN [Xu et al. 2019]

- Tabular data is difficult to model using regular GANs
  - Mixed data types – continuous and discrete
  - Non-Gaussian, multimodal continuous distributions
  - Imbalanced categorical columns – can cause mode collapse

- CTGAN addresses these issues
  - Mode-specific normalization for multimodal distributions
  - The generator is made conditional to solve the imbalance problem



- Does not capture temporal dependence

Flow-based Network Traffic Generation using Generative Adversarial Networks [Ring et al. 2019]

- Synthesize netflow data
- Use Wasserstein GAN
- Derive embeddings for netflow fields

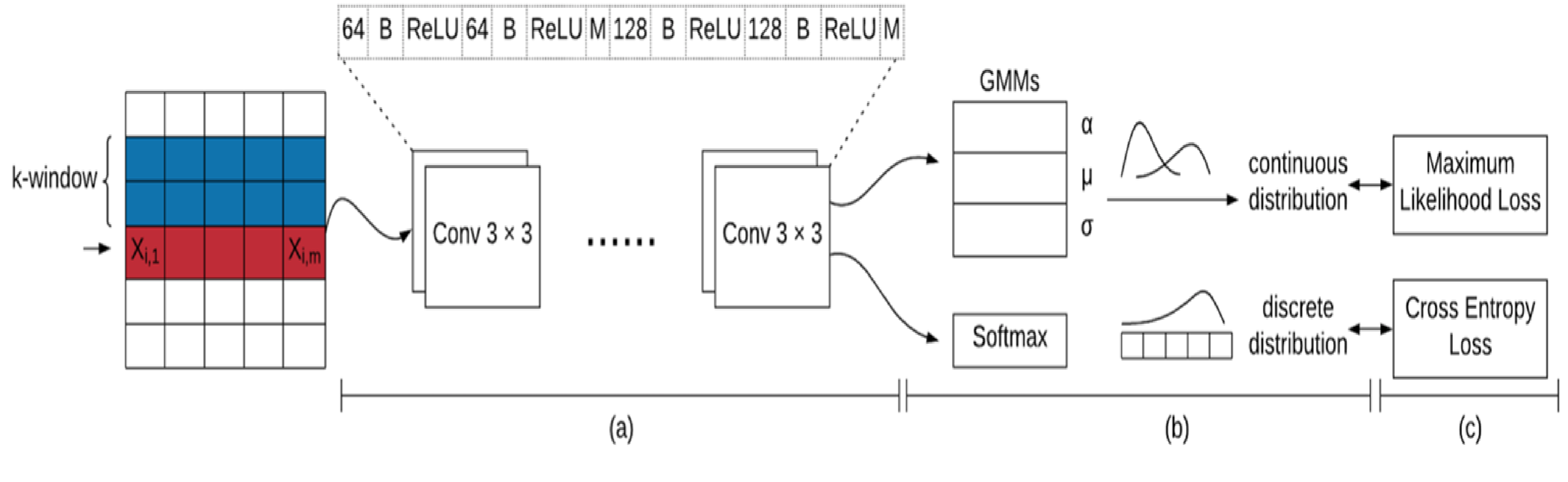
Evaluation

- Distributions
- Domain tests

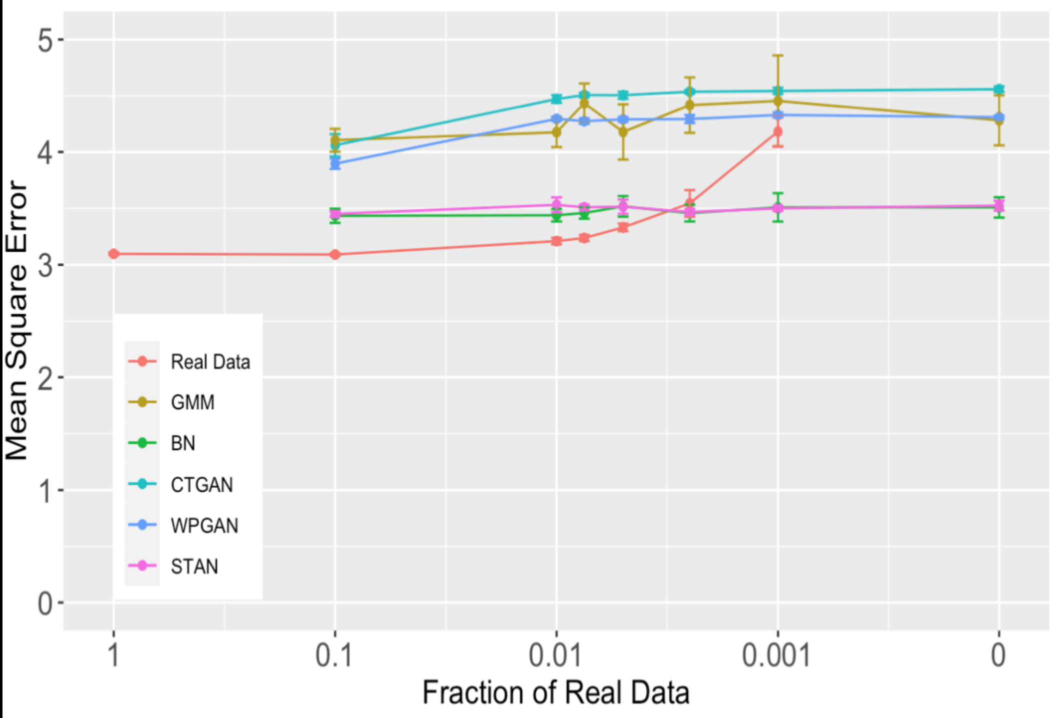
STAN: Synthetic Network Traffic Generation with Generative Neural Models [Xu et al. 2021]

- Uses neural autoregressive model, similar to pixelCNN
- CNN-based architecture to estimate conditional probability density
- Captures temporal dependence as well

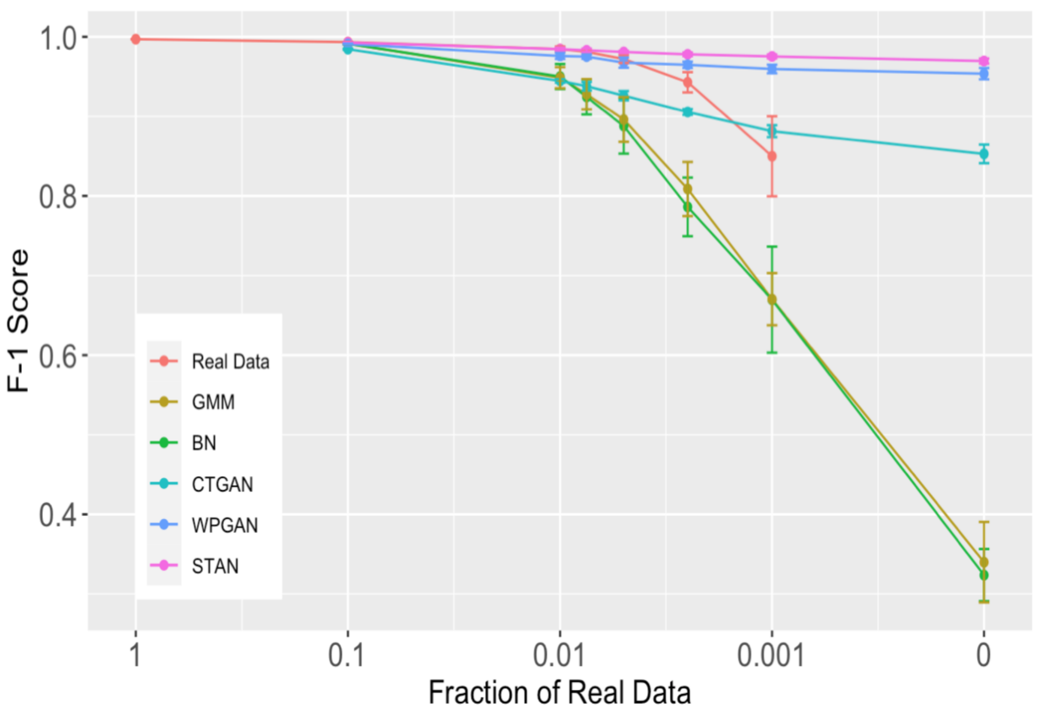
$$\mathbb{P}(\mathbf{x}) = \prod_{i=1}^{n}\prod_{j=1}^{m}\mathbb{P}(x_{i,j}|x_{i-k},...,x_{i-1})$$



STAN: performance on ML tasks



(b) Mean Square Error of *bytes* Value Forecasting Task



(a) F1-score of same-row Protocol prediction Task

[Xu et al. 2021]

References

- Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems 27 (2014).
- Xu, Lei, et al. "Modeling tabular data using conditional gan." Advances in Neural Information Processing Systems 32 (2019).
- Ring, Markus, et al. "Flow-based network traffic generation using generative adversarial networks." Computers & Security 82 (2019): 156-172.
- Xu, Shengzhe, et al. "Stan: Synthetic network traffic generation with generative neural models." International Workshop on Deployable Machine Learning for Security Defense. Springer, Cham, 2021.

Summary

- Unique Challenges
    - Data
        - privacy
        - representative
        - no labels
        - no true benchmarks
    - Moving target
        - non-stationarity
        - adversary
- Feature Learning
    - Network traffic
        - as image
        - as multivariate sequence
        - as graph
- Synthetic data Generation
    - GANs
    - Autoregressive NN
    - Validation