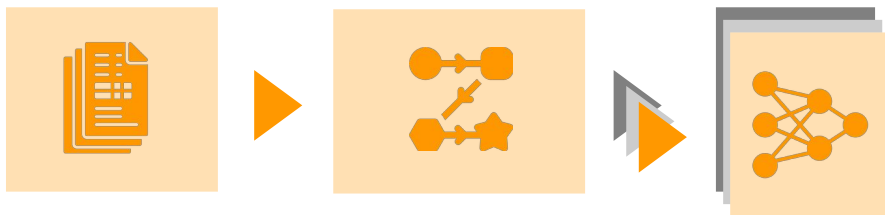


A New Hope for Network Model Generalization

ACM HotNets 2022



Alexander Dietmüller


Siddhant Ray

Romain Jacob

Laurent Vanbever

ETH zürich

What do these systems have in common?



GENET: Automatic Curriculum Generation for Learning Adaptation in Networking

Zhengxu Xia¹, Yajie Zhou², Francis Y. Yan¹, Junchen Jiang¹
¹University of Chicago, ²Boston University, ³Microsoft Research

ABSTRACT

As deep reinforcement learning (RL) showcases its strengths in networking, its pitfalls are also coming to the public's attention. Training on a wide range of network environments leads to sub-optimal performance, whereas training on a narrow distribution of environments results in poor generalization.

This work presents GENET, a new training framework for learning better RL-based network adaptation algorithms. GENET is built on curriculum learning, which has proved effective against similar issues in other RL applications. At a high level, curriculum learning gradually feeds more "difficult" environments to the training rather than choosing them uniformly at random. However, applying curriculum learning in networking is non-trivial since the "difficulty" of a network environment is unknown. Our insight is to leverage traditional rule-based baselines. If the current RL model performs significantly worse on a network environment than the rule-based baselines, then further training it in this environment tends to bring substantial improvement. GENET automatically searches for such environments and iteratively promotes them to

(CC) [24], adaptive bitrate streaming (ABR) [32], load balancing (LB) [1], wireless resource scheduling [16], and cloud scheduling [24]. For a given distribution of training network environments (e.g., network connections with certain bandwidths, patterns, delay, and queue lengths), RL trains a policy to optimize performance over these environments.

However, these RL-based techniques face two challenges that can ultimately impede their wide use in practice:

- **Training in a wide range of environments:** When the training distribution spans a wide variety of network environments (e.g., a large range of possible bandwidths), an RL policy may perform poorly even if tested in the environments drawn from the same distribution as training.
- **Generalization:** RL policies trained on one distribution of synthetic or trace-driven environments may have poor performance and even anomalous behavior when tested in a new distribution of environments.

Our analysis in §2 will reveal that, across three RL use cases in networking, these challenges can cause well-trained RL policies

Video streaming, congestion control, and load balancing
 SIGCOMM'22
 [GENET]

Learning in situ: a randomized experiment in video streaming

Francis Y. Yan Hudson Ayers Chenzhi Zhu¹ Sajjad Fouladi
 James Hong Keyi Zhang Philip Levis Kevin Winstein

¹Stanford University, ²Tsinghua University


Abstract

We describe the results of a randomized controlled trial of video-streaming algorithms for bitrate selection and network prediction. Over the last year, we have streamed 38.6 years of video to 63,508 users across the Internet. Sessions are randomized in blinded fashion among algorithms.

We found that in the real-world setting, it is difficult for sophisticated or machine-learned control schemes to outperform

In the academic literature, many recent ABR algorithms use statistical and machine-learning methods [4, 25, 38–40, 46], which allow algorithms to consider many input signals and try to perform well for a wide variety of clients. An ABR decision can depend on recent throughput, client-side buffer occupancy, delay, the experience of clients on similar ISPs or types of connectivity, etc. Machine learning can find patterns in seas of data and is a natural fit for this problem domain. However, it is a potential lesson that the performance of

Video streaming
 NSDI'20
 [Puffer]



MimicNet: Fast Performance Estimates for Data Center Networks with Machine Learning

Qizhen Zhang, Kelvin K.W. Ng, Charles W. Katzor¹, Shen Yan¹, Joao Sedoc², and Vincent Liu
 University of Pennsylvania, ¹Vassarstammar College, ²Peking University, ³New York University

(zqzhen,keizheng,liu@upenn.edu, katzor@vassar.edu, janshen@nyu.edu, jsedoc@nyu.edu, vliu@cs.cornell.edu)

ABSTRACT

As-scale evaluation of new data center network innovations is becoming increasingly intractable. This is true for testbeds, where few, if any, can afford a dedicated, full-scale replica of a data center. It is also true for simulations, which while originally designed for precisely this purpose, have struggled to cope with the size of today's networks.

This paper presents an approach for quickly obtaining accurate performance estimates for large data center networks. Our system, MimicNet, provides users with the familiar abstraction of a packet-level simulation for a portion of the network while leveraging redundancy and recent advances in machine learning to quickly and accurately approximate portions of the network that are not directly visible. MimicNet can provide over two orders of magnitude speedup compared to regular simulation for a data center with

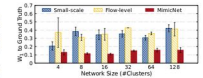


Figure 1: Accuracy for MimicNet's predictions of the FCT distribution for a range of data center sizes. Accuracy is quantified via the Wasserstein distance (W₁) to the distribution observed in the original simulation. Lower is better. Also shown are the accuracy of a flow-level simulator (FlowGrid) and the accuracy of assuming a small (2-cluster) sim-

Network simulation
 SIGCOMM'21
 [MimicNet]

Biases in Data-Driven Networking, and What to Do About Them

Mihovil Bartulovic Junchen Jiang Sivaraman Balakrishnan
 Carnegie Mellon University Microsoft Research/Carnegie Mellon University Carnegie Mellon University

Vyass Sekar Bruno Sinopoli
 Carnegie Mellon University Carnegie Mellon University

ABSTRACT

Recent efforts highlight the promise of data-driven approaches to optimize network decisions. Many such efforts use trace-driven evaluation; i.e., running offline analysis on network traces to estimate the potential benefits of different policies before running them in practice. Unfortunately, such frameworks can have fundamental pitfalls (e.g., skew due to previous policies that were used in the data collection phase and insufficient data for specific subpopulations) that could lead to misleading estimates and ultimately suboptimal decisions. In this paper, we shed light on such pitfalls and identify a promising roadmap to address these pitfalls by leveraging parallelism in causal inference, namely the Doubly Robust estimator.




Figure 1: Trace-driven evaluation predicts the best policies using traces of empirical measurement data.

A common approach is trace-driven evaluation as depicted in Figure 1. Here, we evaluate different policies (e.g., a server selection policy) in an offline fashion by using network traces, before deploying them. Since operators may be reluctant to

Data-driven networking
 HotNets'16
 [Biases]

What do these systems have in common?

They have the same problem setting.



From past traffic ...



... an ML system estimates the state of the network to make a prediction.

MimicNet packet (drop, latency, ECN)

Puffer transmission time

GENET bitrate for next chunk

...

What do these systems have in common?

They have the same problem setting. **But that's about it.**



From past traffic ...



... an ML system estimates the state of the network to make a prediction.

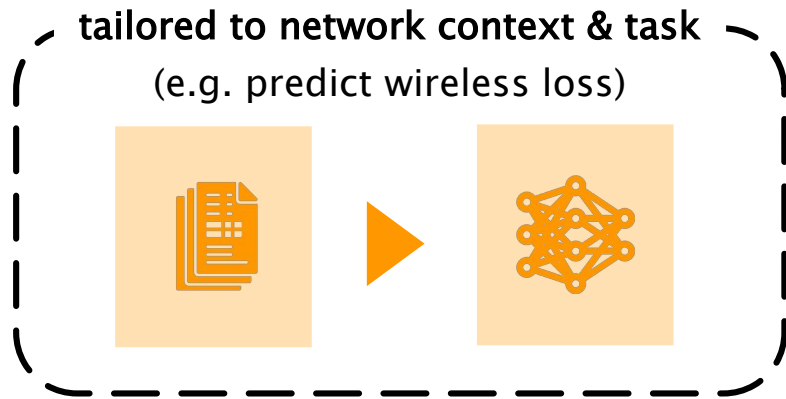
MimicNet packet (drop, latency, ECN)

Puffer transmission time

GENET bitrate for next chunk

...

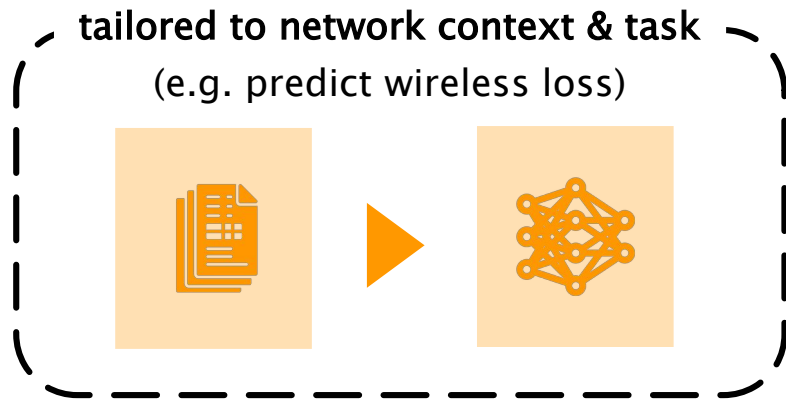
ML systems in networking do not generalize.



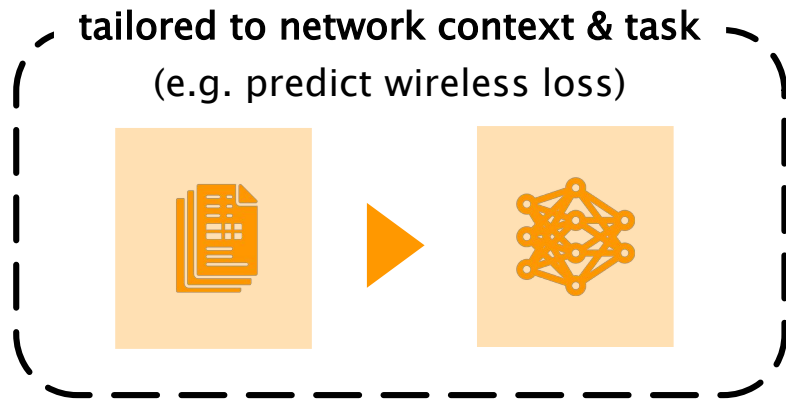
SO
WHAT

SO are the
WHAT consequences for
ML in networking?

ML systems in networking do not generalize. This limits re-usability, forcing us to repeat data collection, model design, and training.



ML systems in networking do not generalize. This limits re-usability, forcing us to repeat data collection, model design, and training.



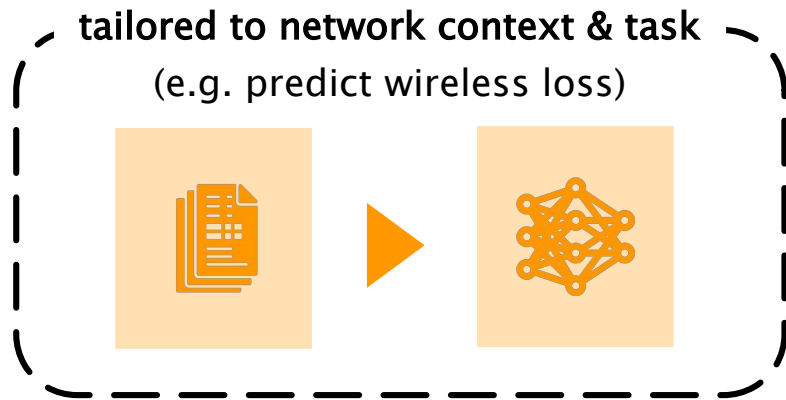
Same task (*predict loss*) with data from

a context in situ

✓

[Puffer]

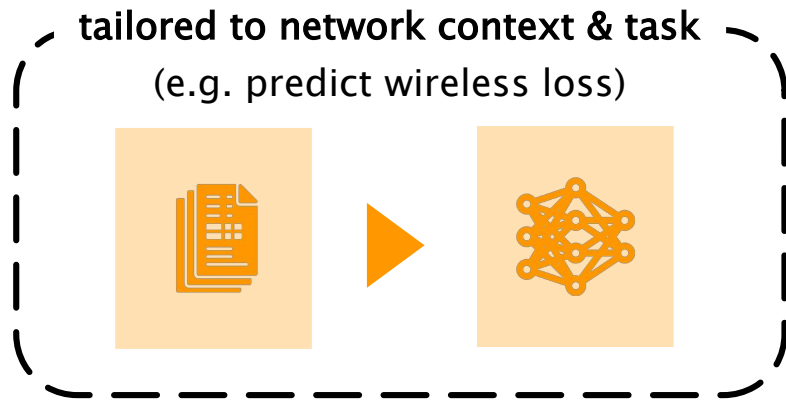
ML systems in networking do not generalize. This limits re-usability, forcing us to repeat data collection, model design, and training.



Same task (*predict loss*) with data from

- | | | |
|---------------------------------------|-------|----------|
| a context in situ | ✓ | [Puffer] |
| a similar context (<i>wireless</i>) | ✓ / ✗ | [GENET] |

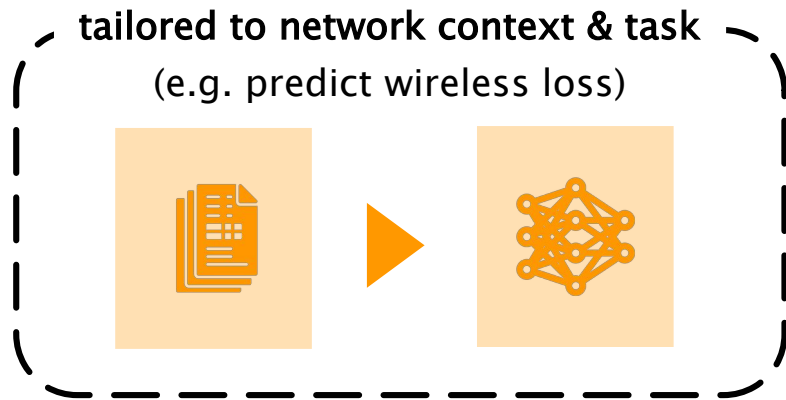
ML systems in networking do not generalize. This limits re-usability, forcing us to repeat data collection, model design, and training.



Same task (*predict loss*) with data from

a context in situ	✓	[Puffer]
a similar context (<i>wireless</i>)	✓ / ✗	[GENET]
a different context (<i>wired</i>)	✗	[Biases]

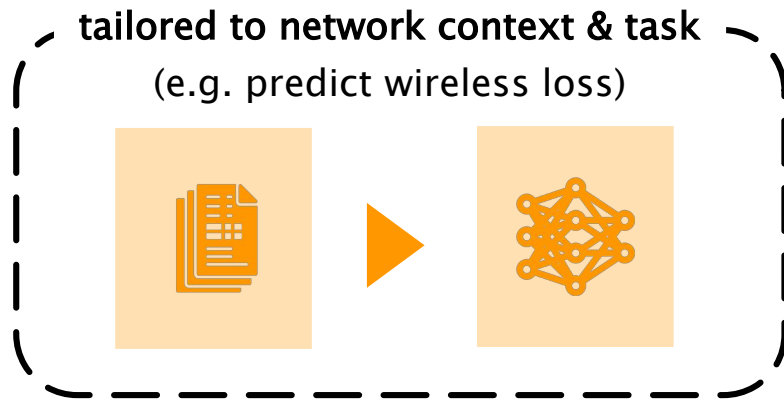
ML systems in networking do not generalize. This limits re-usability, forcing us to repeat data collection, model design, and training.



Same task (*predict loss*) with data from

a context in situ	✓	[Puffer]
a similar context (<i>wireless</i>)	✓ / ✗	[GENET]
a different context (<i>wired</i>)	✗	[Biases]
multiple contexts (<i>both</i>)	✗	

ML systems in networking do not generalize. This limits re-usability, forcing us to repeat data collection, model design, and training.



Same task (*predict loss*) with data from

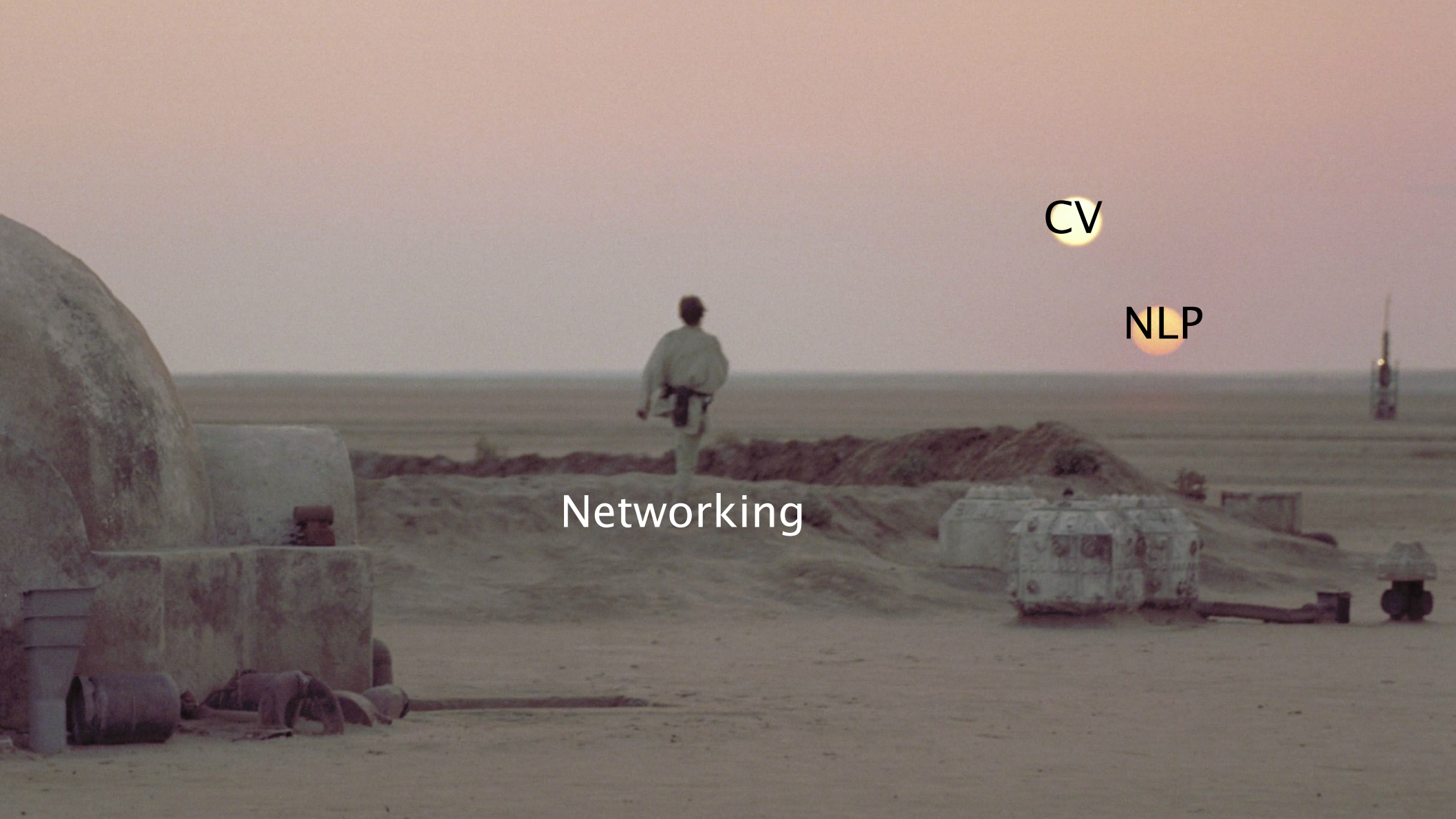
a context in situ	✓	[Puffer]
a similar context (<i>wireless</i>)	✓ / ✗	[GENET]
a different context (<i>wired</i>)	✗	[Biases]
multiple contexts (<i>both</i>)	✗	

Different task (*e.g. predict delay*)

✗ (requires a completely new model and data)

- Is there no way to get
- ◆ optimal performance
 - ◆ for multiple contexts and different tasks
 - ◆ without starting from scratch every time ?

A New Hope for Network Model Generalization



CV

NLP

Networking

In NLP and CV, Transformer-based architectures
generalize by learning to infer sequence context.

In NLP and CV, Transformer-based architectures
generalize by learning to infer sequence context.

Dall-E 2

input: (text)

output: (generated image)

In NLP and CV, Transformer-based architectures
generalize by learning to infer sequence context.

Dall-E 2

input: (text)

Hand me that stick!

Stick to that hand.

output: (generated image)

In NLP and CV, Transformer-based architectures
generalize by learning to infer sequence context.

Dall-E 2

input: (text)

Hand me that stick!

Stick to that hand.

output: (generated image)



In NLP and CV, Transformer-based architectures
generalize by learning to infer sequence context.

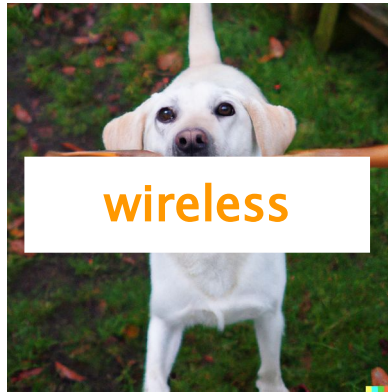
Dall-E 2

input: (text)

Hand me that stick!

Stick to that hand.

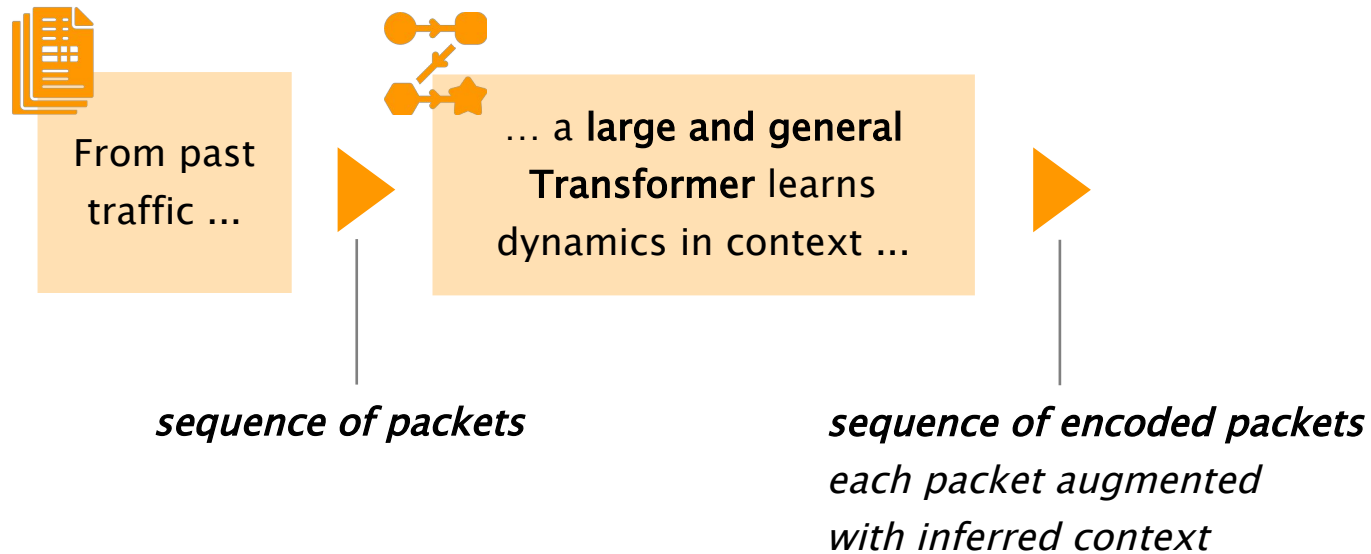
output: (generated image)



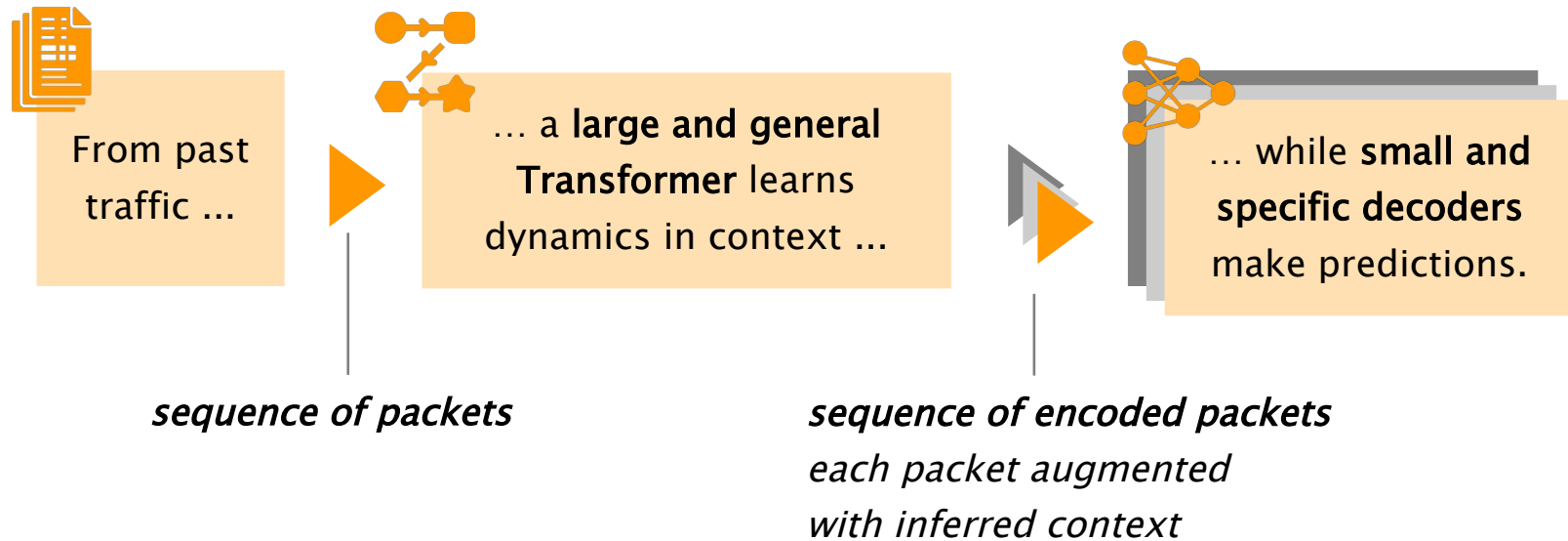
- Maybe we can get
- ◆ optimal performance
 - ◆ for multiple contexts and different tasks
 - ◆ without starting from scratch every time ?

A general pre-trained Transformer encoder
can be combined with specific fine-tuned decoders.

A general pre-trained Transformer encoder
can be combined with specific fine-tuned decoders.

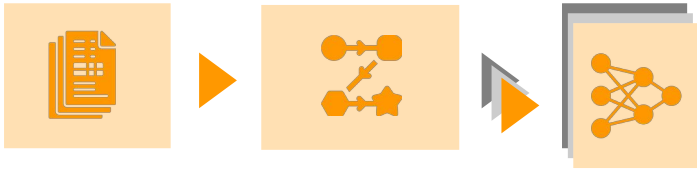


A general pre-trained Transformer encoder
can be combined with specific fine-tuned decoders.



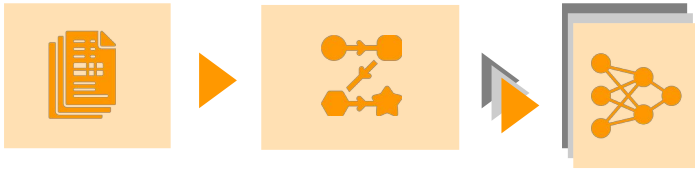
- There is a way to get
- ◆ optimal performance
 - ◆ for multiple contexts and different tasks
 - ◆ without starting from scratch every time !

We cannot just copy an NLP Transformer:
a Network Traffic Transformer (NTT) must handle network challenges!



We cannot just copy an NLP Transformer:

a Network Traffic Transformer (NTT) must handle network challenges!



Challenge #1

Avoid packet features tailored to a specific task.

Challenge #2

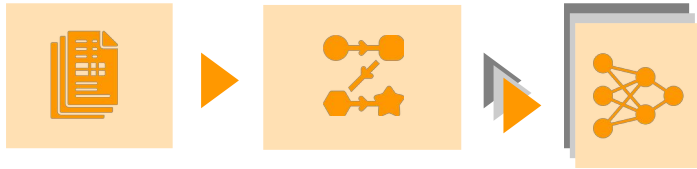
Process long sequences without losing detail.

Challenge #3

Learn contextual dynamics during pre-training.

We cannot just copy an NLP Transformer:

a Network Traffic Transformer (NTT) must handle network challenges!



Challenge #1

Avoid packet features tailored to a specific task.
→ learning features

Challenge #2

Process long sequences without losing detail.
→ aggregate past packets hierarchically

Challenge #3

Learn contextual dynamics during pre-training.
→ pre-train to predict end-to-end delay



In simulation, we observe first evidence
that networking could benefit from pre-trained models as well.

We pretrain, ...



context

30 senders and
a single shared
bottleneck

task

delay prediction

In simulation, we observe first evidence that networking could benefit from pre-trained models as well.

We pretrain, ...



context

30 senders and
a single shared
bottleneck

task

delay prediction

... fine-tune, ...



with different contexts

indep. bottlenecks with
unobserved cross-traffic

In simulation, we observe first evidence that networking could benefit from pre-trained models as well.

We pretrain, ...



context

30 senders and
a single shared
bottleneck

task

delay prediction

... fine-tune, ...



with different contexts

indep. bottlenecks with
unobserved cross-traffic



with another task

predict message
completion time

In simulation, we observe first evidence that networking could benefit from pre-trained models as well.

We pretrain, ...



context

30 senders and
a single shared
bottleneck

task

delay prediction

... fine-tune, ...



with different contexts

indep. bottlenecks with
unobserved cross-traffic



with another task

predict message
completion time

... and find that we:

- ◆ get equal or better performance
 - ◆ with less training time
- compared to starting from scratch.

In simulation, we observe first evidence that networking could benefit from pre-trained models as well.

We pretrain, ...



context

30 senders and a single shared bottleneck

task

delay prediction

... fine-tune, ...



with different contexts

indep. bottlenecks with unobserved cross-traffic



with another task

predict message completion time

... and find that we:

- ◆ get equal or better performance
- ◆ with less training time

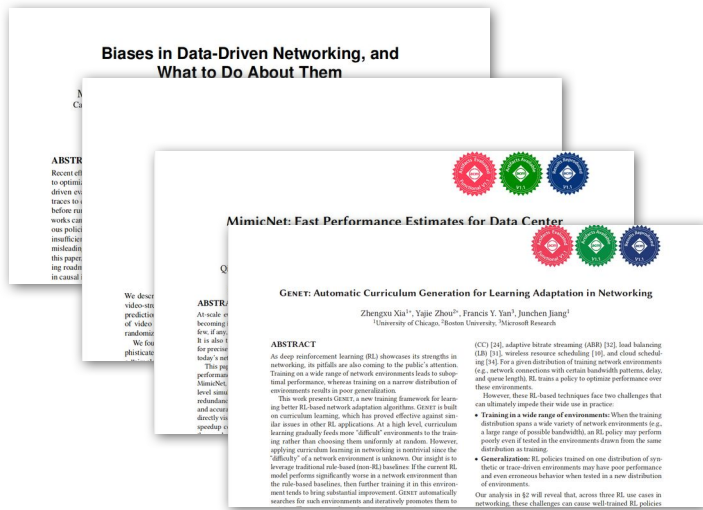
compared to starting from scratch.



NTT *may* generalize.

What next?

Our simulation results are promising, and it is time to use and evaluate NTT-based models in the real-world.



Re-create existing models based on NTT, collecting new data where needed.

Our simulation results are promising, and it is time to **use and evaluate NTT-based models in the real-world.**



Re-create existing models based on NTT, collecting new data where needed.

Create new models based-on NTT.

Real-world applications will reveal all limits, but there are clear steps to **refine the NTT design**.

Real-world applications will reveal all limits, but there are clear steps to **refine the NTT design**.

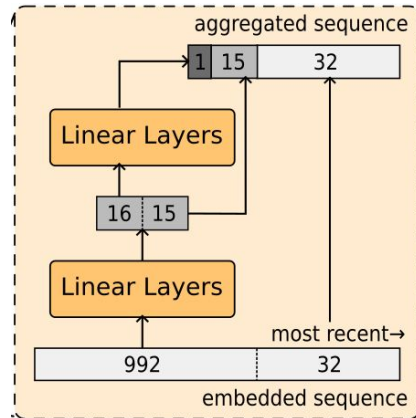
```
▶ Frame 56: 122 bytes on wire (976 bits)
▶ Ethernet II, Src: RivetNet_db:8e:93 (9
▶ Internet Protocol Version 4, Src: 192.
▶ Transmission Control Protocol, Src Por
▶ Secure Sockets Layer
```

How can we represent any combination of headers?

Real-world applications will reveal all limits, but there are clear steps to **refine the NTT design**.

- ▶ Frame 56: 122 bytes on wire (976 bits)
- ▶ Ethernet II, Src: RivetNet_db:8e:93 (9
- ▶ Internet Protocol Version 4, Src: 192.
- ▶ Transmission Control Protocol, Src Por
- ▶ Secure Sockets Layer

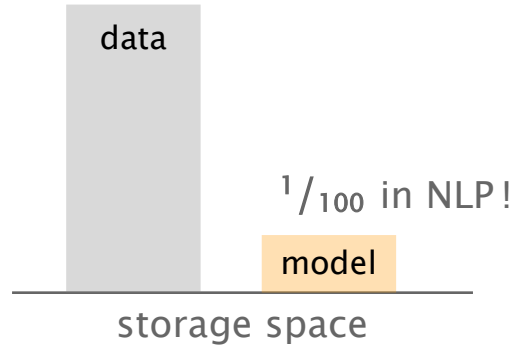
How can we represent any combination of headers?



Which aggregation levels cover all significant network interactions?

Transformer models like NTT extract and compress information,
facilitating sharing and collaboration.

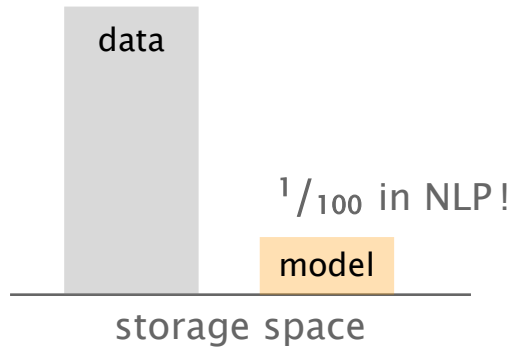
Transformer models like NTT extract and compress information, **facilitating sharing and collaboration.**



train from large traces every time

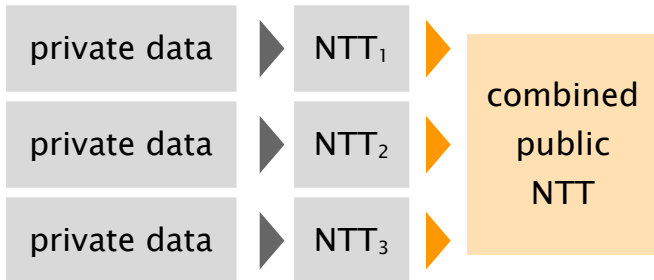
download a pre-trained model

Transformer models like NTT extract and compress information, **facilitating sharing and collaboration.**



train from large traces every time

download a pre-trained model



share private data



combine insights via federated learning

A New Hope for Network Model Generalization

ACM HotNets 2022

Pretraining

Transformers

Networking

Alexander Dietmüller

Siddhant Ray

Romain Jacob

Laurent Vanbever

ETH zürich