Demystifying Deep Learning in Networking

Ying Zheng, Ziyu Liu, Xinyu You, Yuedong Xu   Junchen Jiang

FUDAN UNIVERSITY 1905 復旦大學

THE UNIVERSITY OF CHICAGO

Why **Deep Neural Nets** in Networking?

Application	Example gains
Cloud job scheduling	**32.4%** less job slowdown
Adaptive-bitrate streaming	**12-25%** better QoE
Routing	**23-50%** less congestion
Cellular traffic scheduling	**14.7%** higher utilization

Many more papers are coming!

Great promises of DNNs: Two-digit improvements in multiple applications

So what's wrong with DNNs in networking?

Traditional System State → White-box logic → Action

Deep learning System State → DNN → Action

DNN is a "black box"

Problems
- **Hard to confer causality**
- **Hard to trust**
- **Agnostic to domain knowledge**

Black-Box Problem #1: Hard to confer causality

Sequence of incoming jobs (length) | Picked job (length)

5, 1, 3, 10, … → DNN → 1

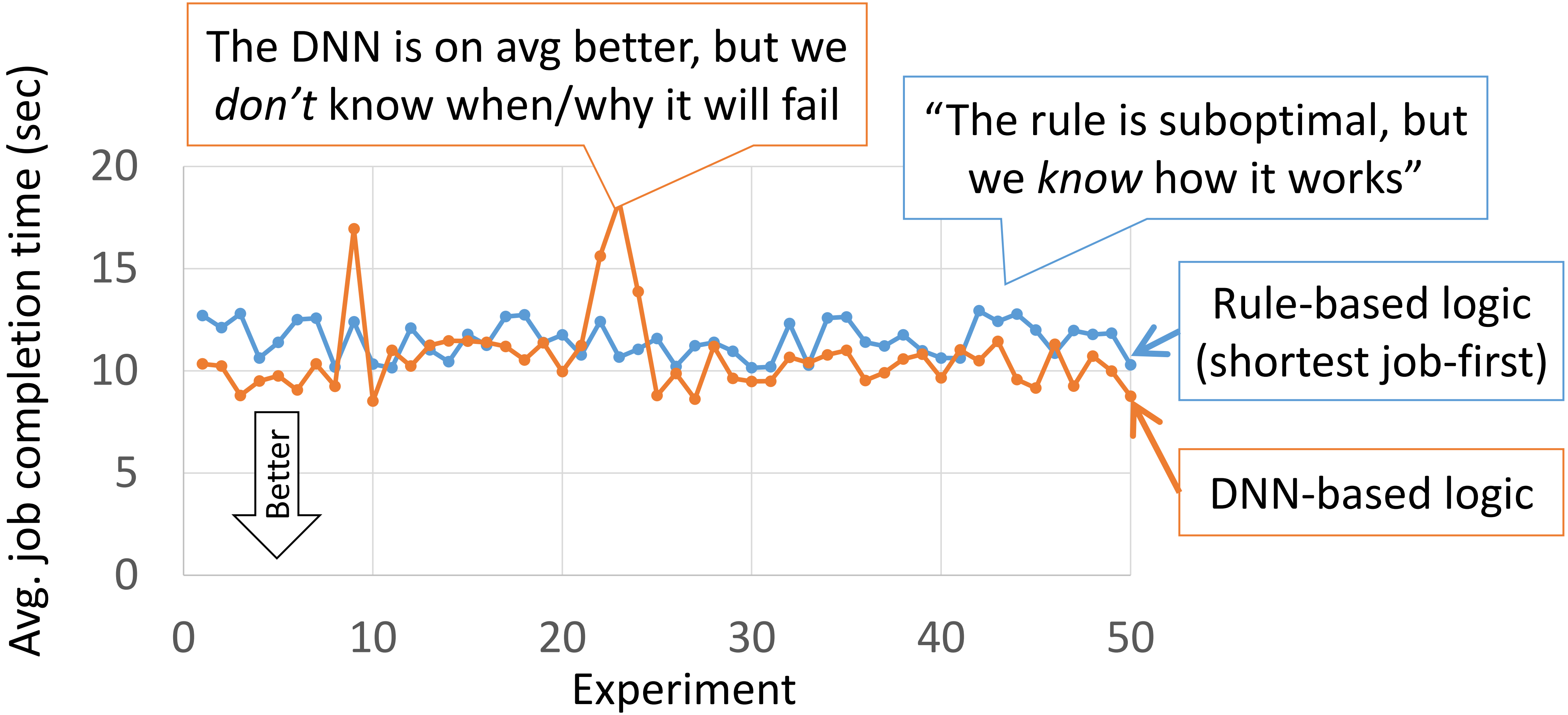
9, 2, 7, 15, … → DNN → 2

So probably DNN is looking for shortest job

But … 20, 11, 13, 14 → DNN → Wait (not 11!)

Correlation ≠ Causality

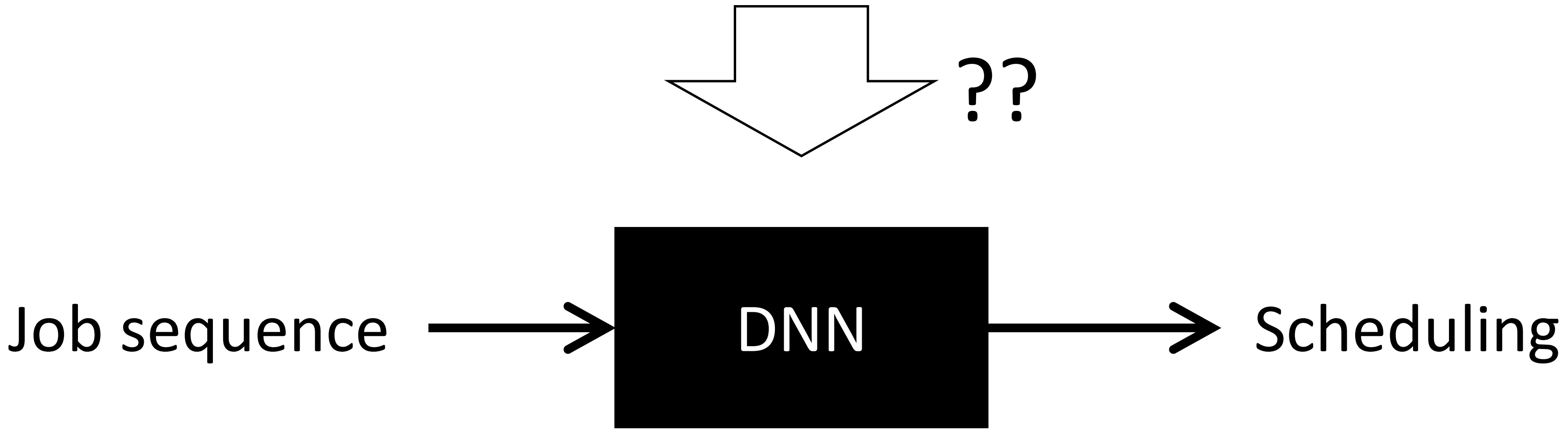
Problem #2: Hard to trust



System engineers may favor certainty over performance

Problem #3: Agnostic to domain-specific knowledge

Empirical requirement:
Resource utilization should be below 80%



Hard to imbue a domain-specific rule in a black box.

Roadmap to white-box DNNs

- How is a decision made?
- When will it fail?
- Can domain-specific knowledge be integrated?

Outline

- Why try to interpret deep learning in networking
- An attempt to explain DNN in networking: A case study
- Examples of worst-case performance of DNN
- Improving performance by domain-specific knowledge

First attempt: use saliency map to understand DNNs

- Calculate gradient vector

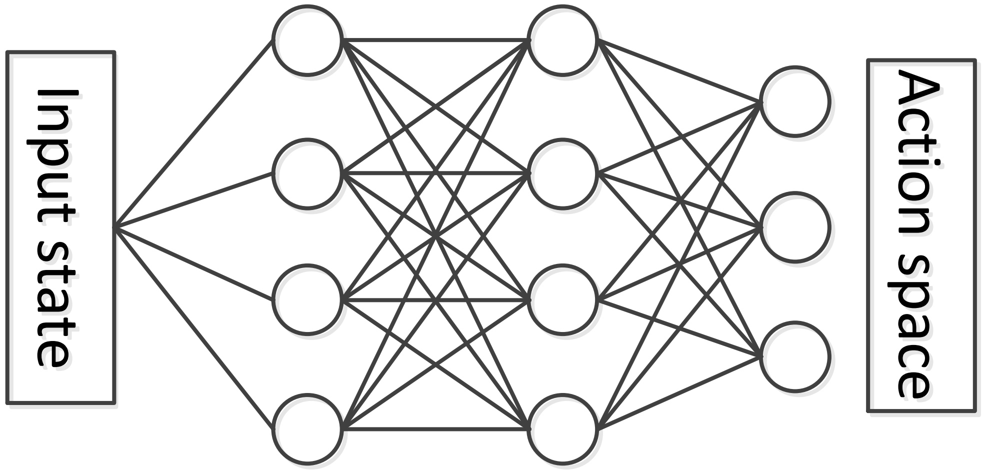
$$w_i = \frac{\partial y_i}{\partial x}$$

- x: input vector
- y: output vector

the j^{th} element of w_i shows how much a small change on the j^{th} feature of x will change how likely i^{th} action is picked.

DeepRM Input/Output: A Case Study

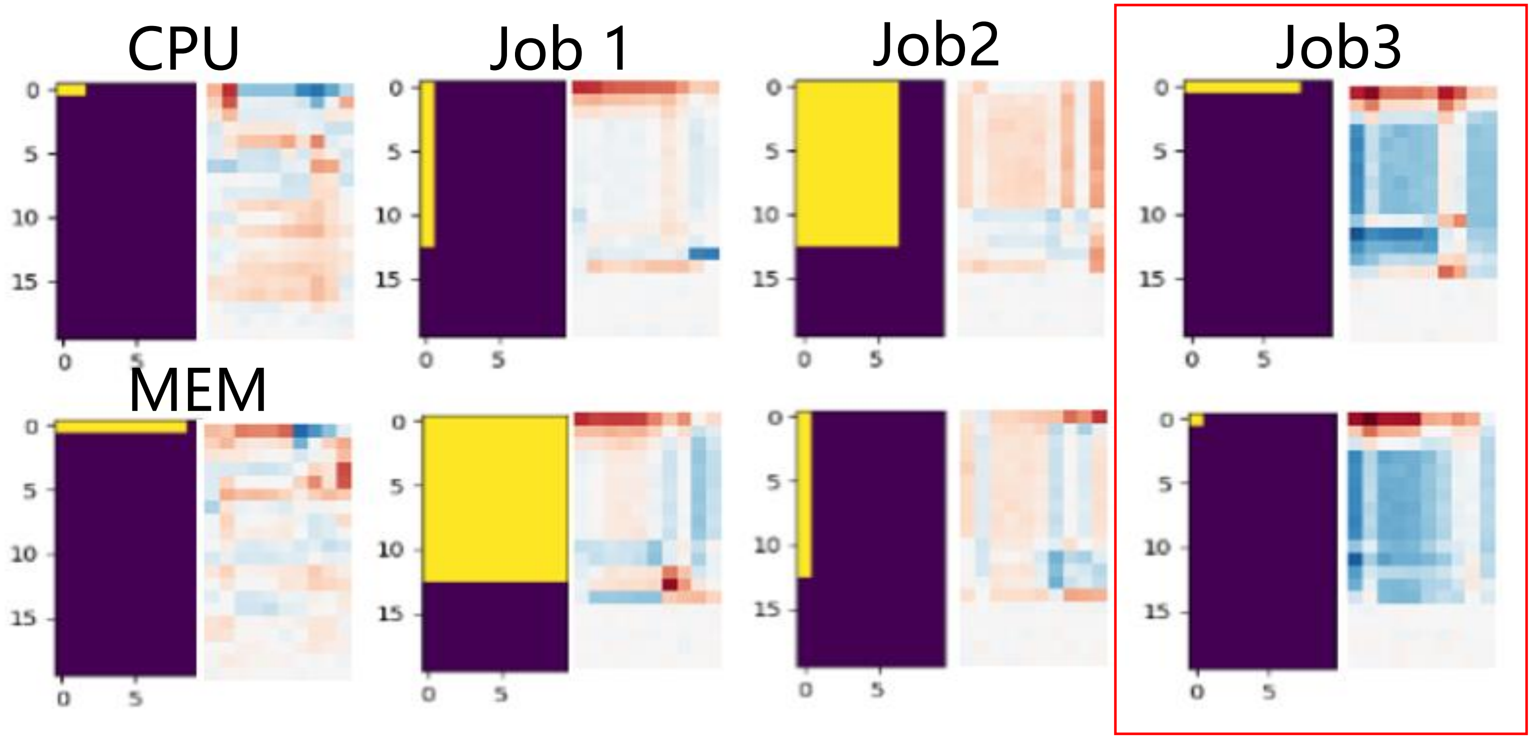
Deep Neural Network(DNN)



- Input: Remaining resource, Resource requirements
- Output: Act probability over possible actions

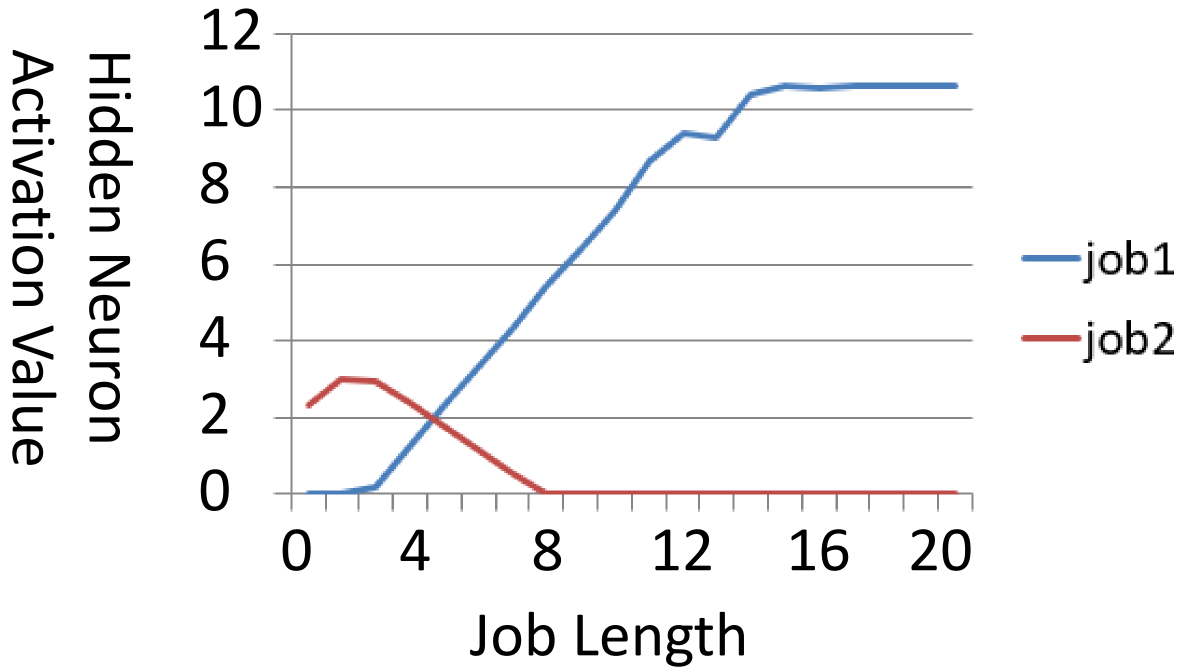
What features make DeepRM a decision?

- Saliency map



- CPU/RAM occupancy has small impact on the output
- Request time of jobs(outstanding district)

Relationship between intermediate output and decision



- High-level feathers have connections to intermediate layer and affect final output

Outline

- Why try to interpret deep learning in networking
- Reverse-engineer DNN in networking: A case study
- Examples of worst-case performance of DNN
- Improving performance by domain-specific knowledge

When DNN fails in a different workload than what it's trained on

Comparison objective	**Testing distribution**	**DNN**	**Rule-based algorithm**
Average job slowdown	Training distribution	3.73	4.51
		3.25	4.87
		3.93	5.37
		3.03	4.71
	Other distribution	10.57	9.69
		11.44	10.23
		12.02	10.75
		11.48	10.22

When DNN fails *even* in the same workload to what it's trained on

Comparison objective	Test data	DeepRM	Rule-based algorithm
Average job slowdown	Normal sequence	1.84	2.01
	Adversarial sequence	1.80	1.46

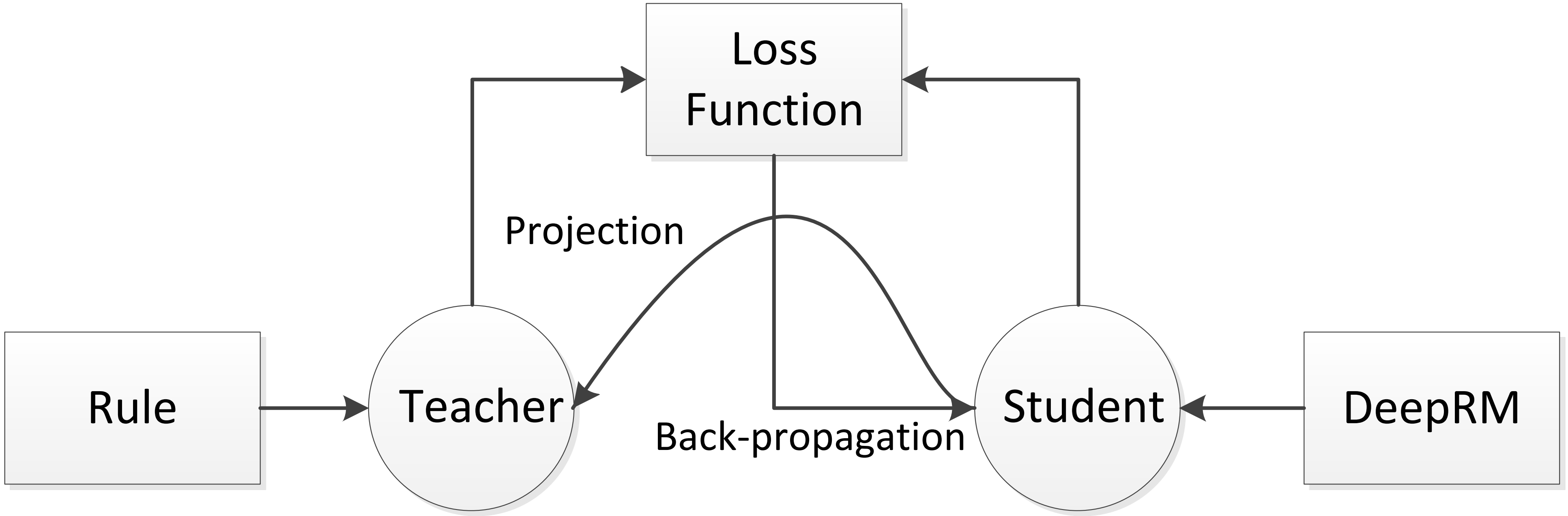
Outline

- Why try to interpret deep learning in networking
- Reverse-engineer DNN in networking: A case study
- Examples of worst-case performance of DNN
- Improving performance by domain-specific knowledge

Domain-specific knowledge might provide a cure

- Better transferability
- Better robustness
- Better training efficiency

A standard way to incorporate domain knowledge



Some takeaways, a lot more need to be done

- DNNs in networking achieve superior performance
- DNNs' limitations compel us to try to at least understand it.
- Our preliminary findings:

a) One can explain to some extent how networking DNNs work

b) DNNs may fail: fits only distribution, vulnerable to noises

Thanks for your listening!