Parity Models
Erasure-Coded Resilience for Prediction Serving Systems

Jack Kosaian Rashmi Vinayak Shivaram Venkataraman

Carnegie Mellon University

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Rashmi Vinayak

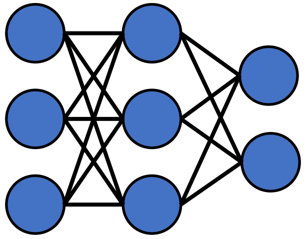
Carnegie Mellon University

Shivaram Venkataraman

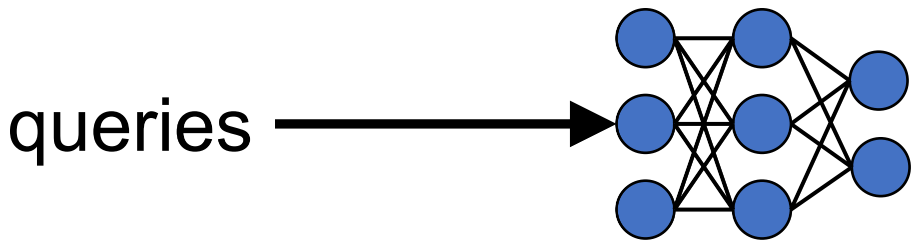
WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Inference: using a trained ML model

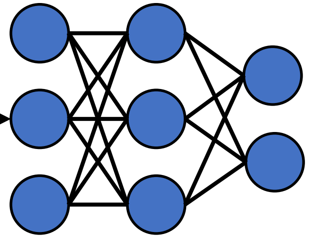
Inference: using a trained ML model



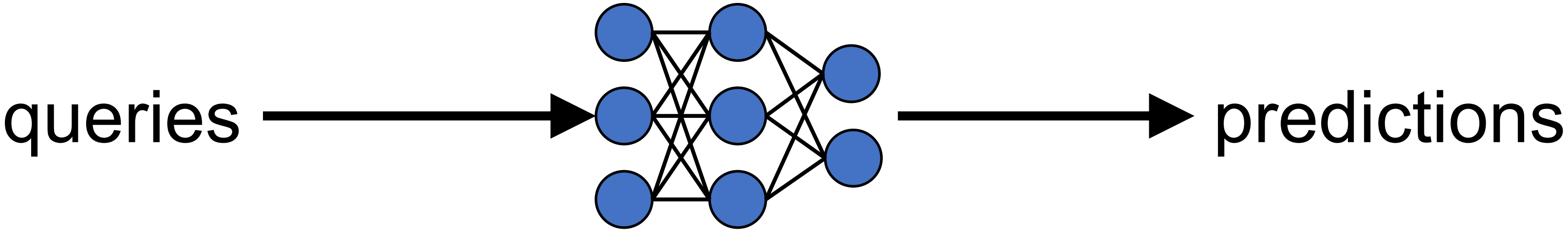
Inference: using a trained ML model



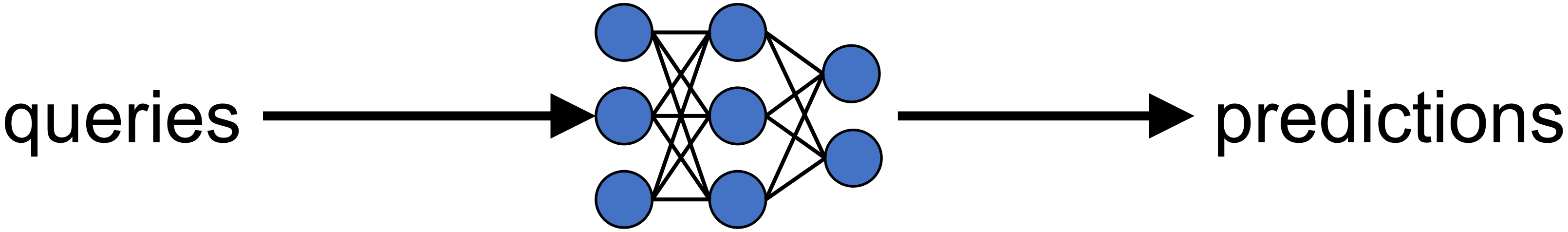
Inference: using a trained ML model

queries →  → predictions

Inference: using a trained ML model

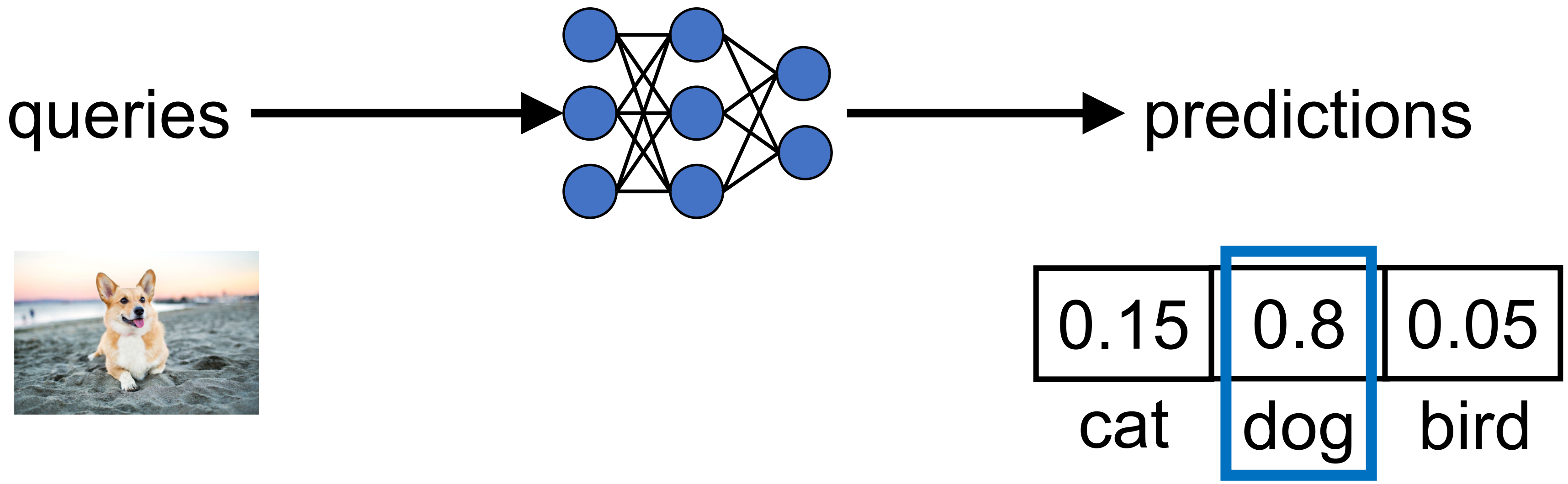


Inference: using a trained ML model



0.15	0.8	0.05
cat	dog	bird

Inference: using a trained ML model



Inference used in latency-sensitive apps

Inference used in latency-sensitive apps

search

translation

ranking

Inference used in latency-sensitive apps

search

translation

ranking

Inference must operate with low, predictable latency

Prediction serving systems: inference in clusters

Prediction serving systems: inference in clusters

Open Source

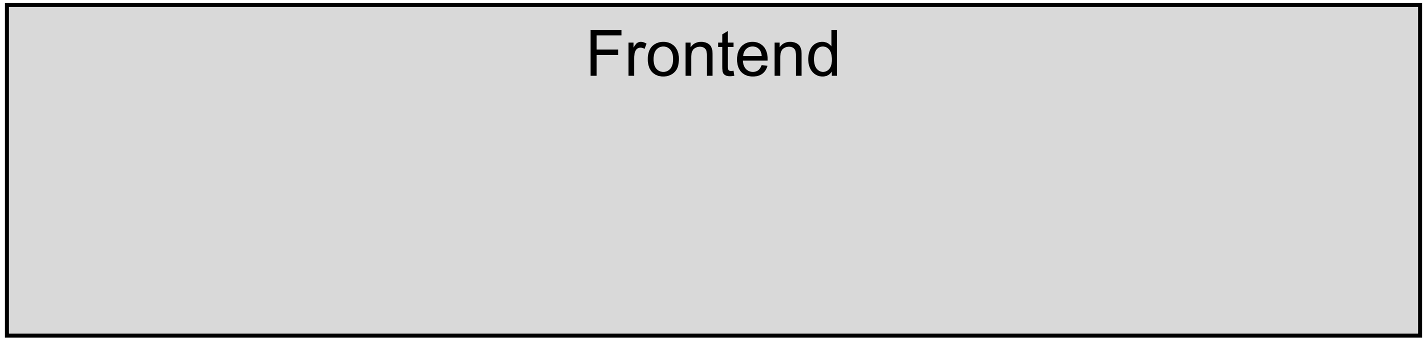
Clipper

TensorFlow Serving

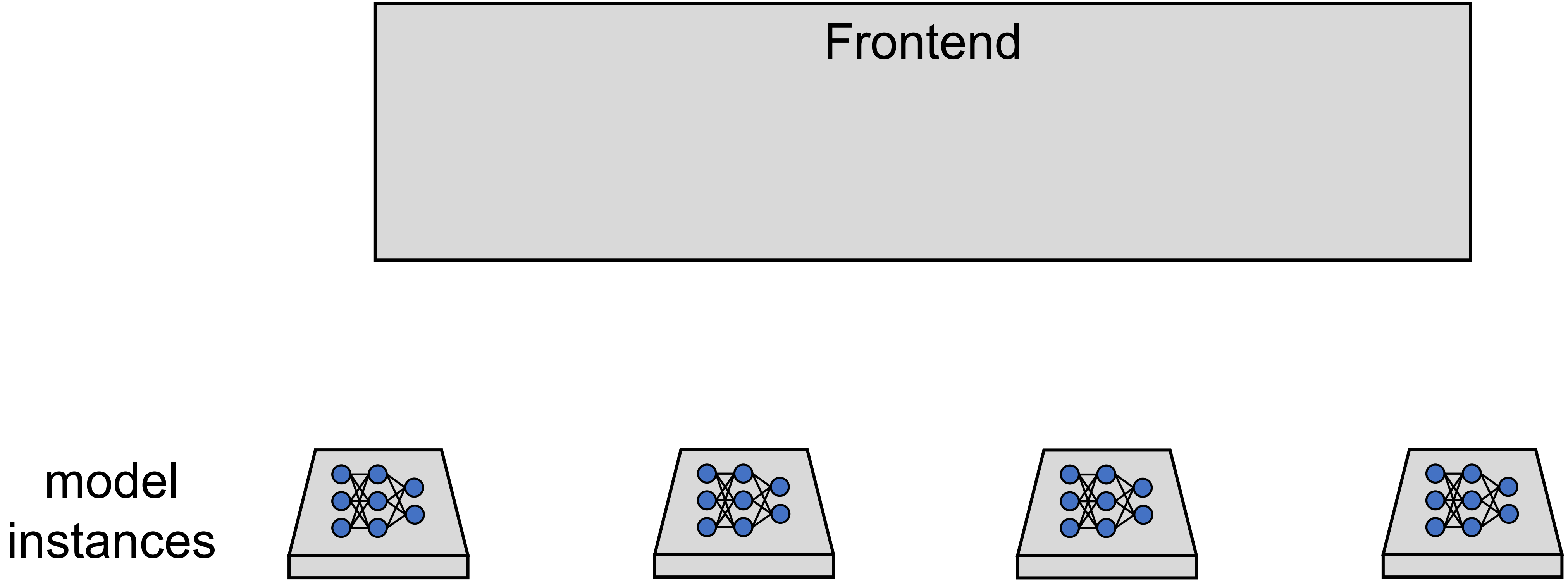
Cloud Services

Prediction serving systems: inference in clusters

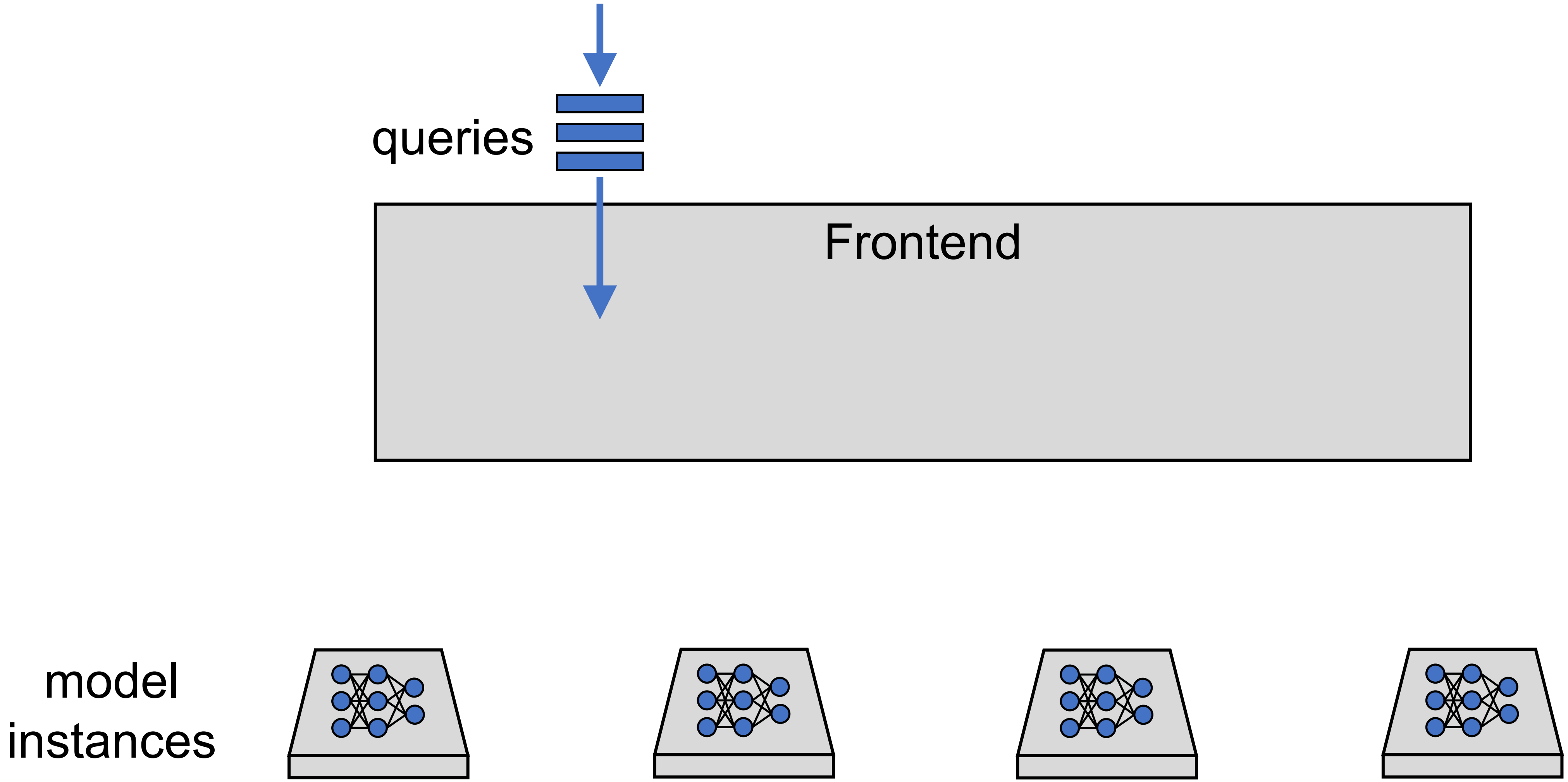
Prediction serving systems: inference in clusters



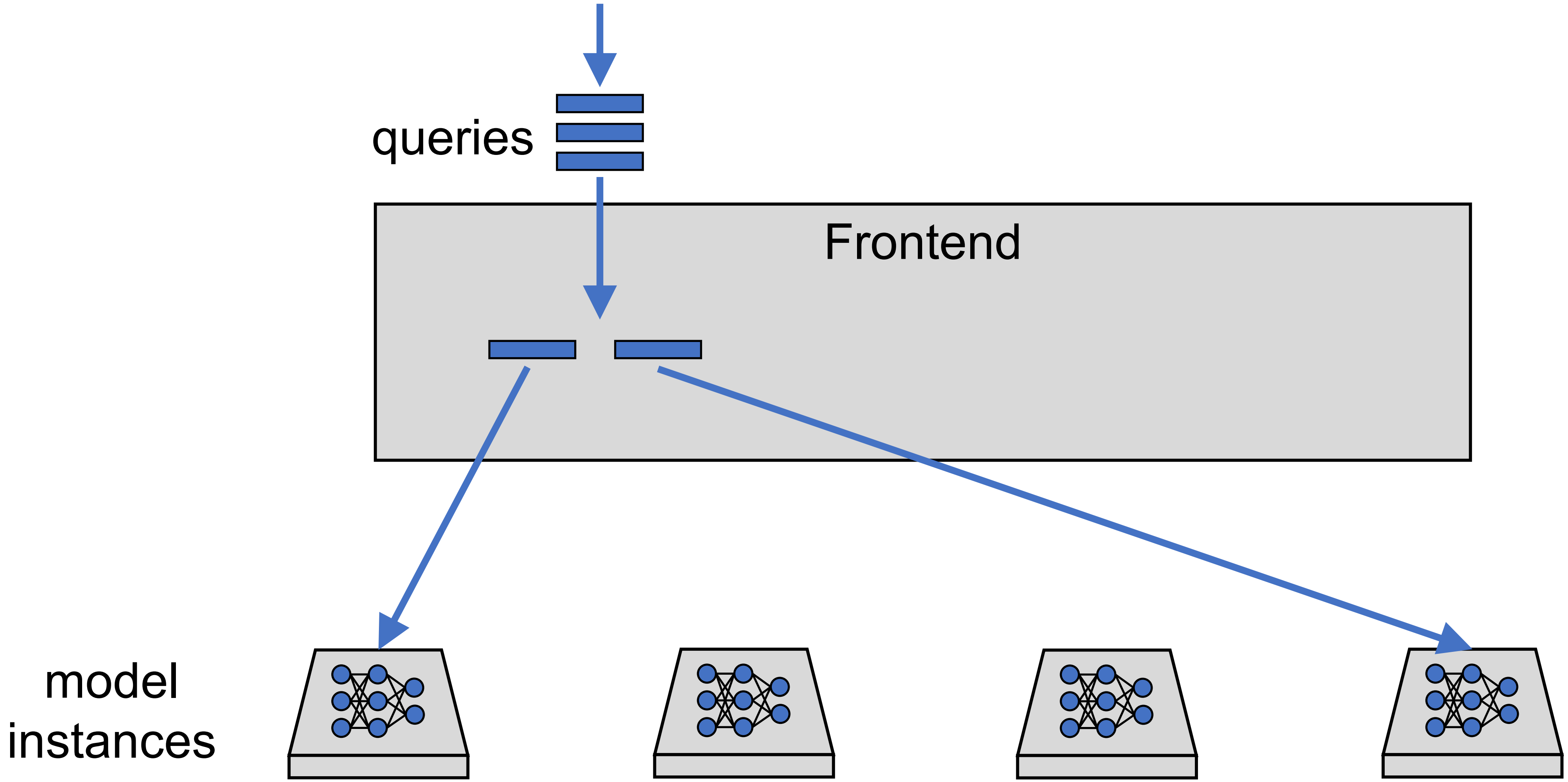
Prediction serving systems: inference in clusters



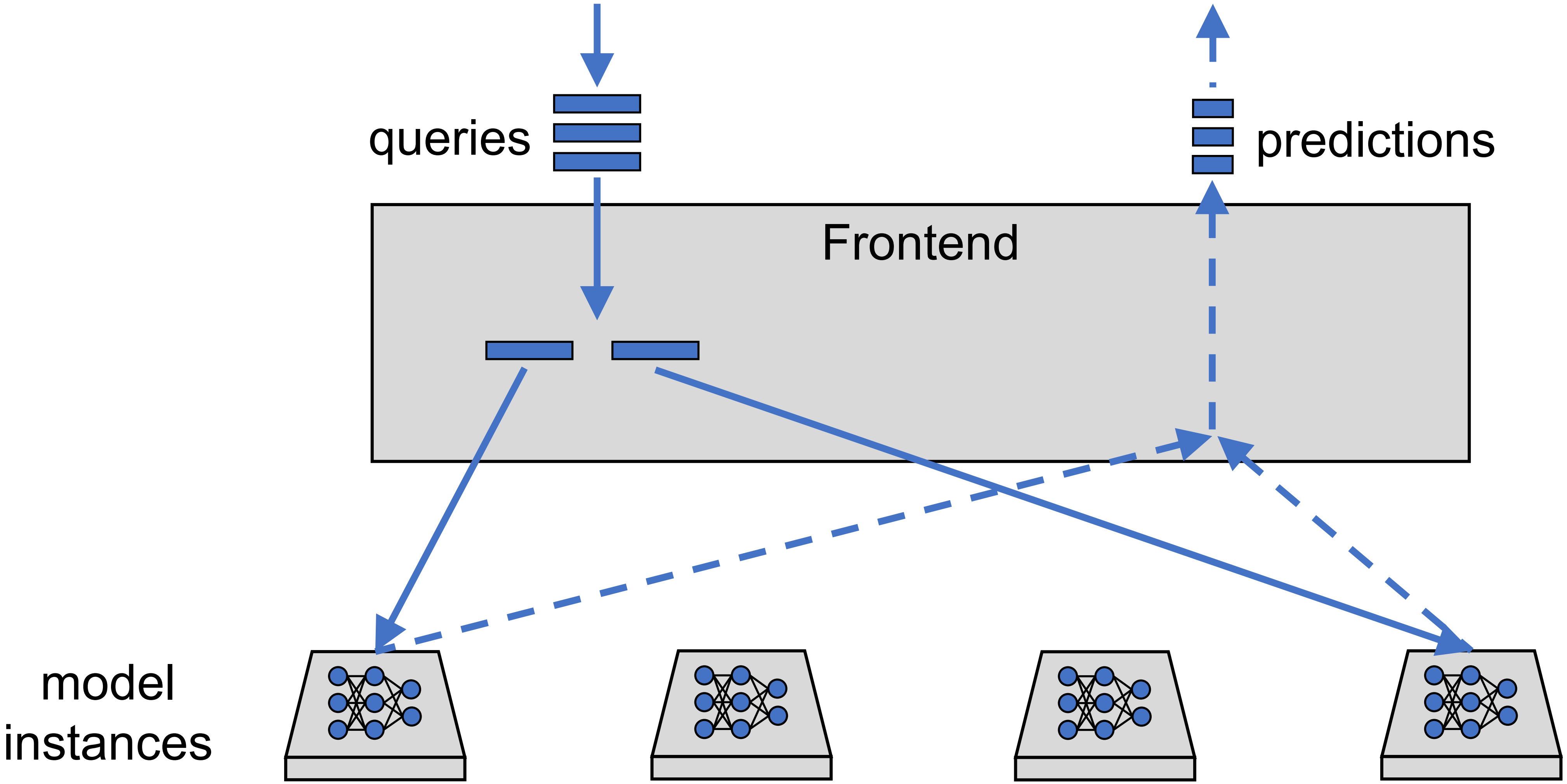
Prediction serving systems: inference in clusters



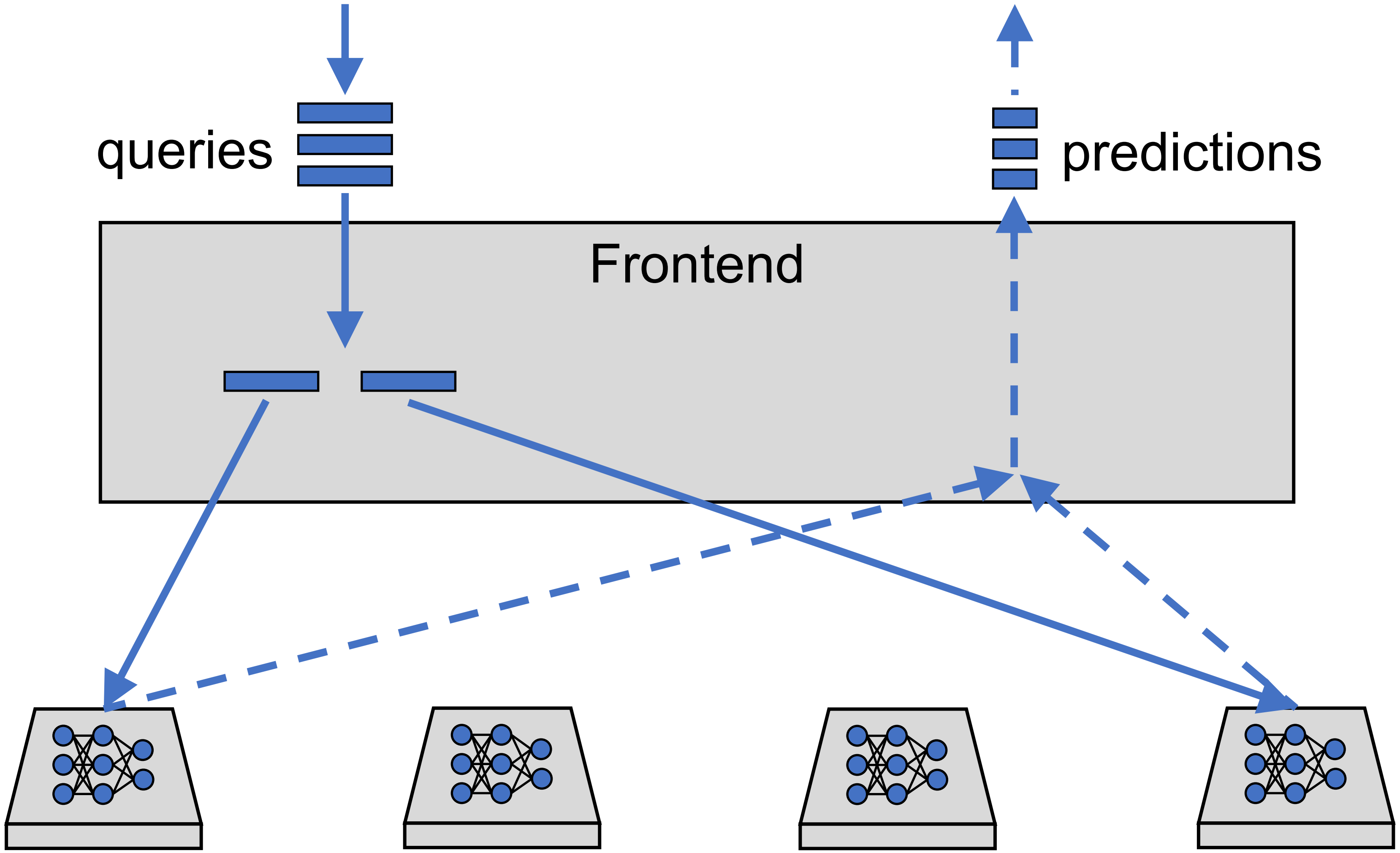
Prediction serving systems: inference in clusters



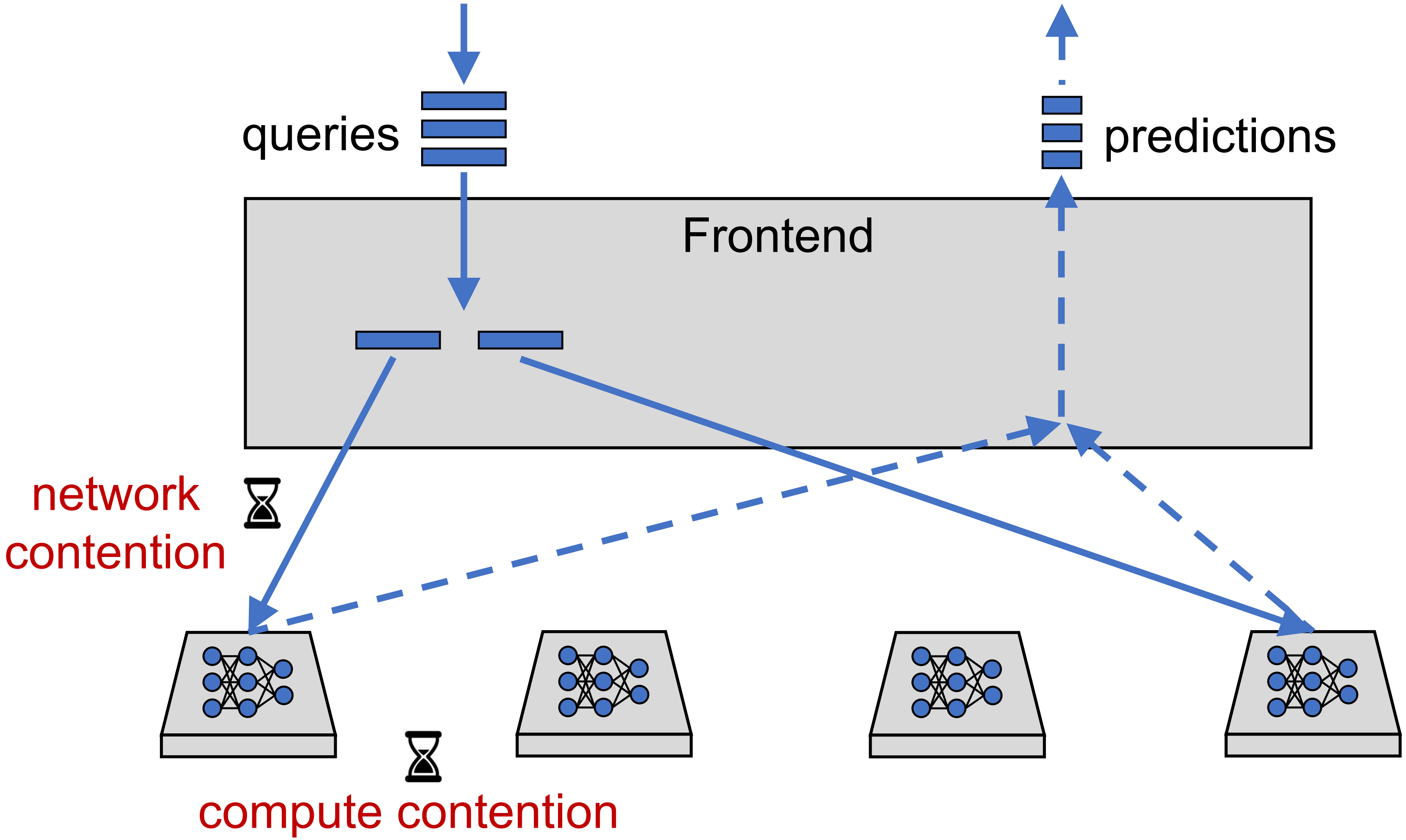
Prediction serving systems: inference in clusters



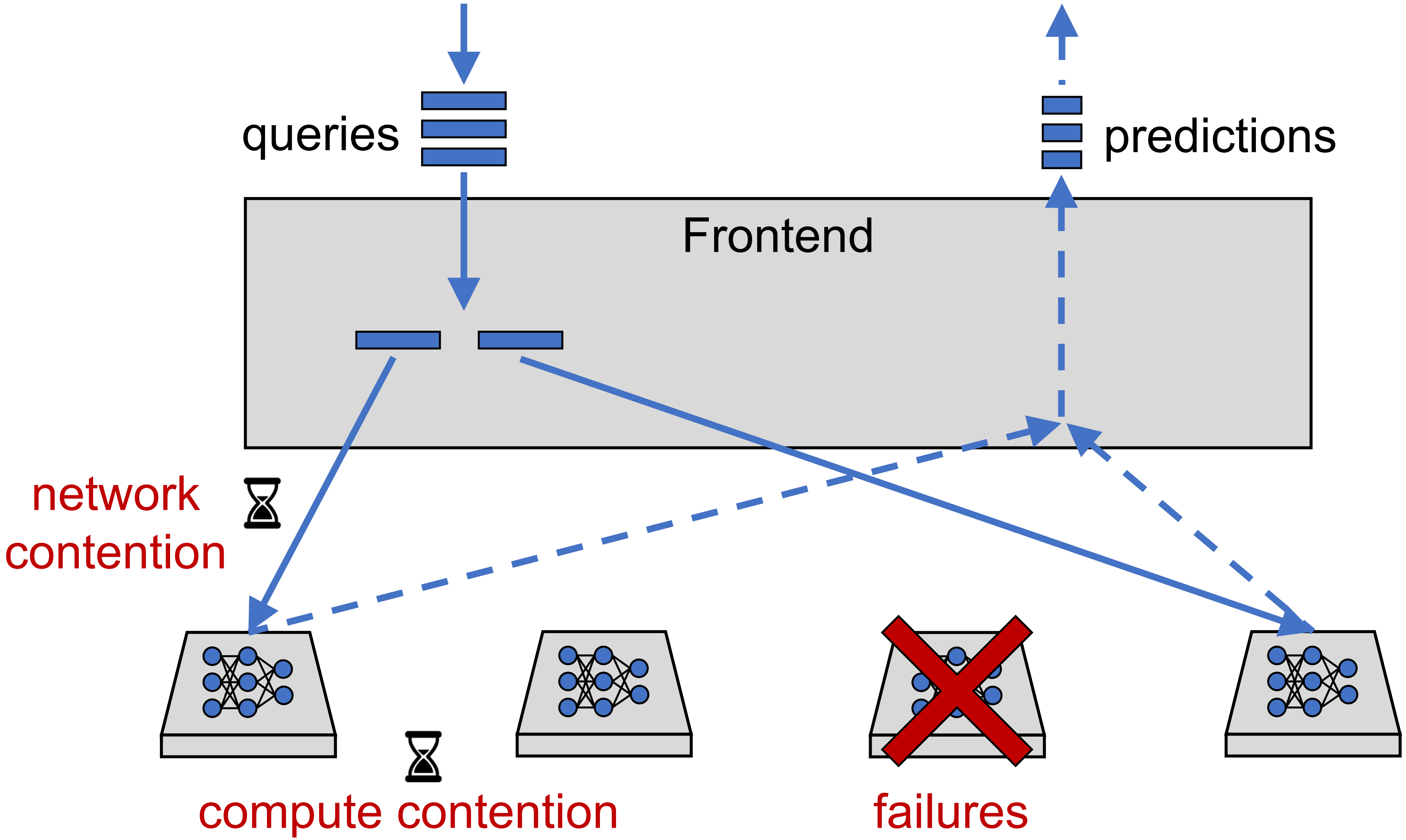
Slowdowns and failures in inference



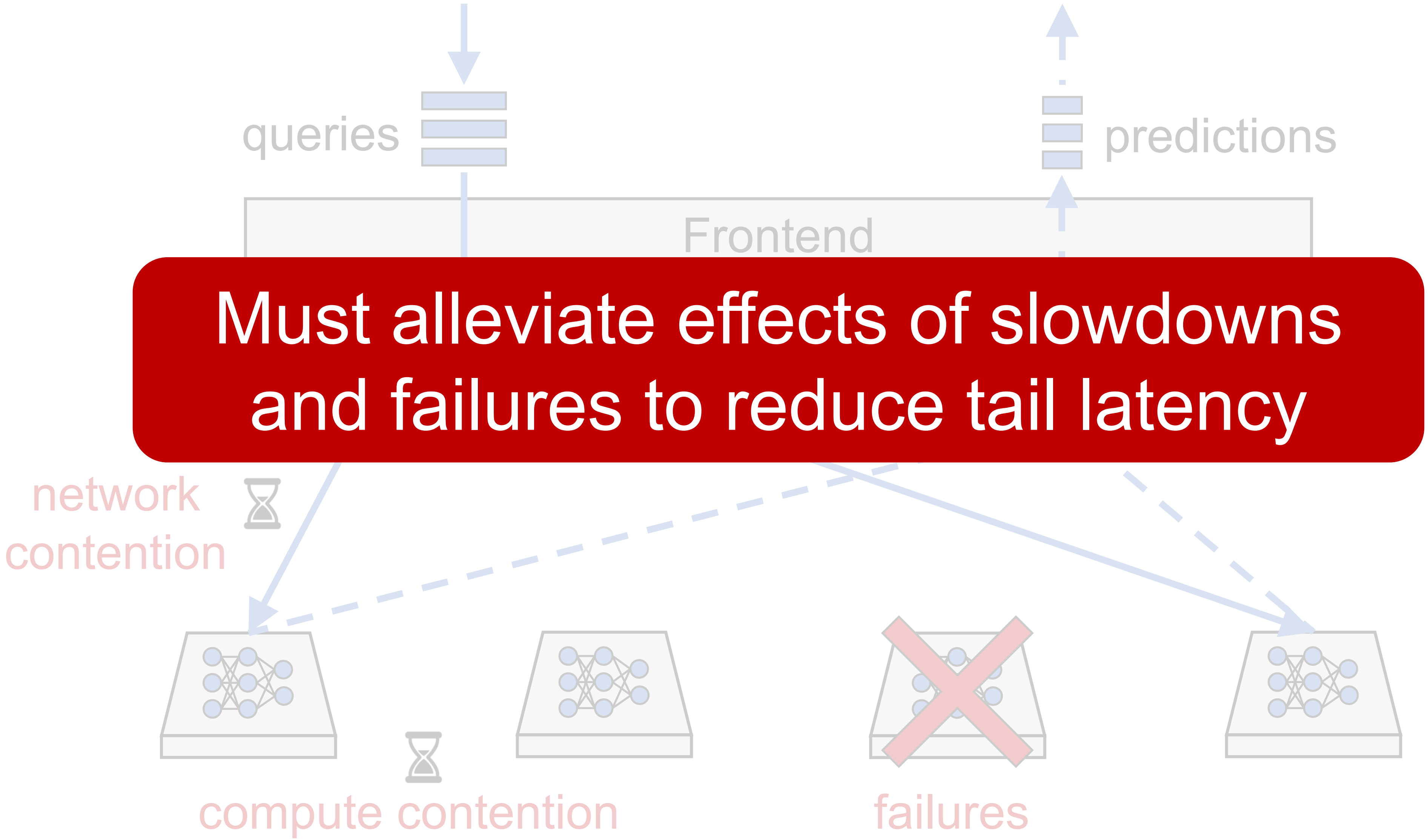
Slowdowns and failures in inference



Slowdowns and failures in inference



Slowdowns and failures in inference



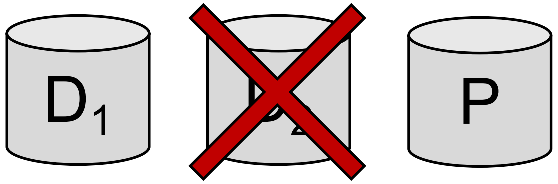
Must alleviate effects of slowdowns and failures to reduce tail latency

Erasure codes widely deployed in systems

Erasure codes widely deployed in systems

Storage systems

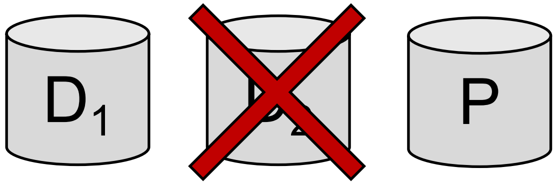
resource-efficient resilience



Erasure codes widely deployed in systems

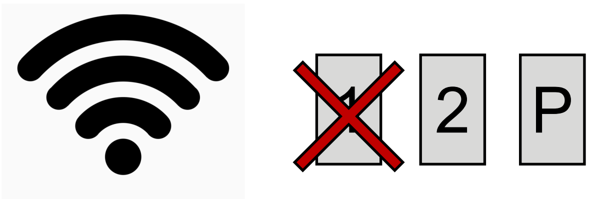
Storage systems

resource-efficient resilience



Communication systems

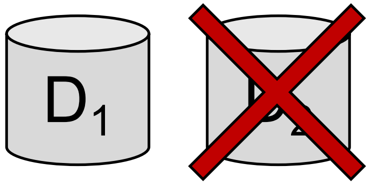
low-latency packet loss recovery



Erasure codes widely deployed in systems

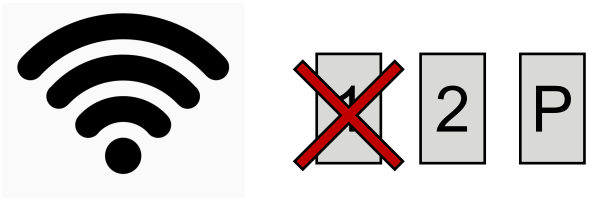
Storage systems

resource-efficient resilience



Communication systems

low-latency packet loss recovery



Erasure codes for systems that compute over data (e.g., serving systems)?

Erasure codes for resilient ML inference

This work: overcome fundamental challenges, use erasure codes for reducing tail latency in machine learning inference

Erasure codes for resilient ML inference

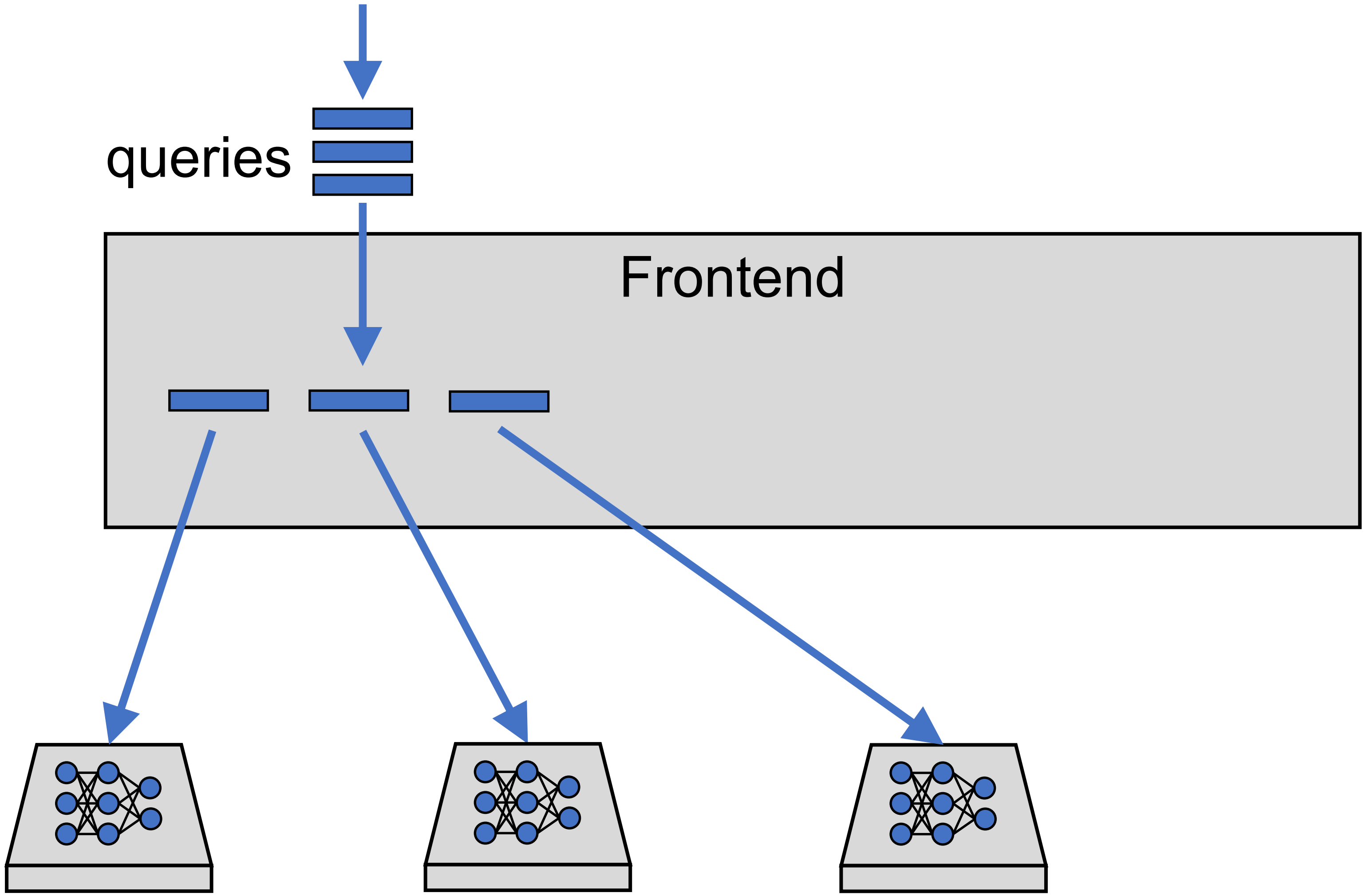
This work: overcome fundamental challenges, use erasure codes for reducing tail latency in machine learning inference

Bring benefits of erasure codes to inference

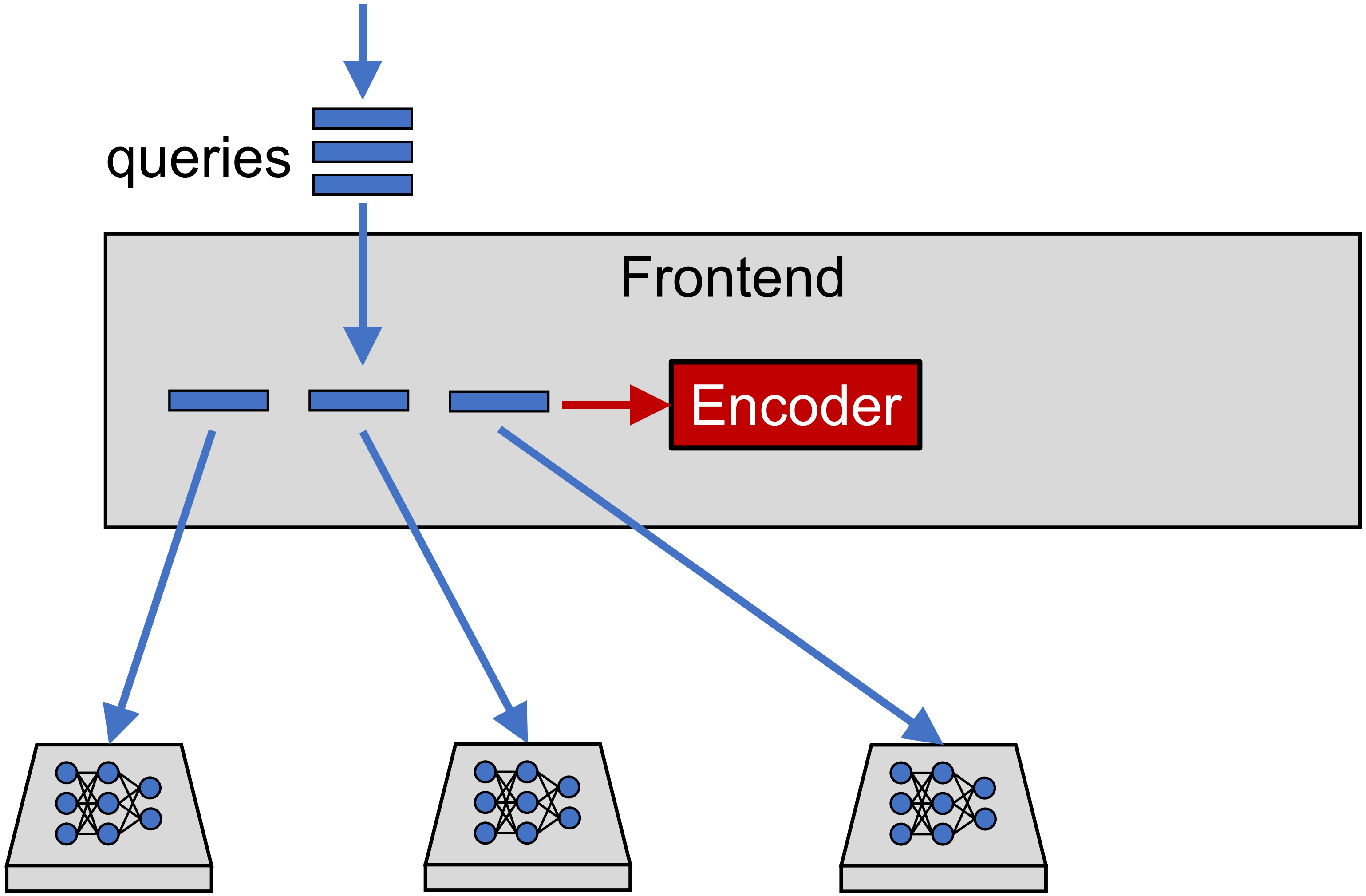
- low recovery latency
- more resource-efficient than replication

End goal: erasure-coded prediction serving

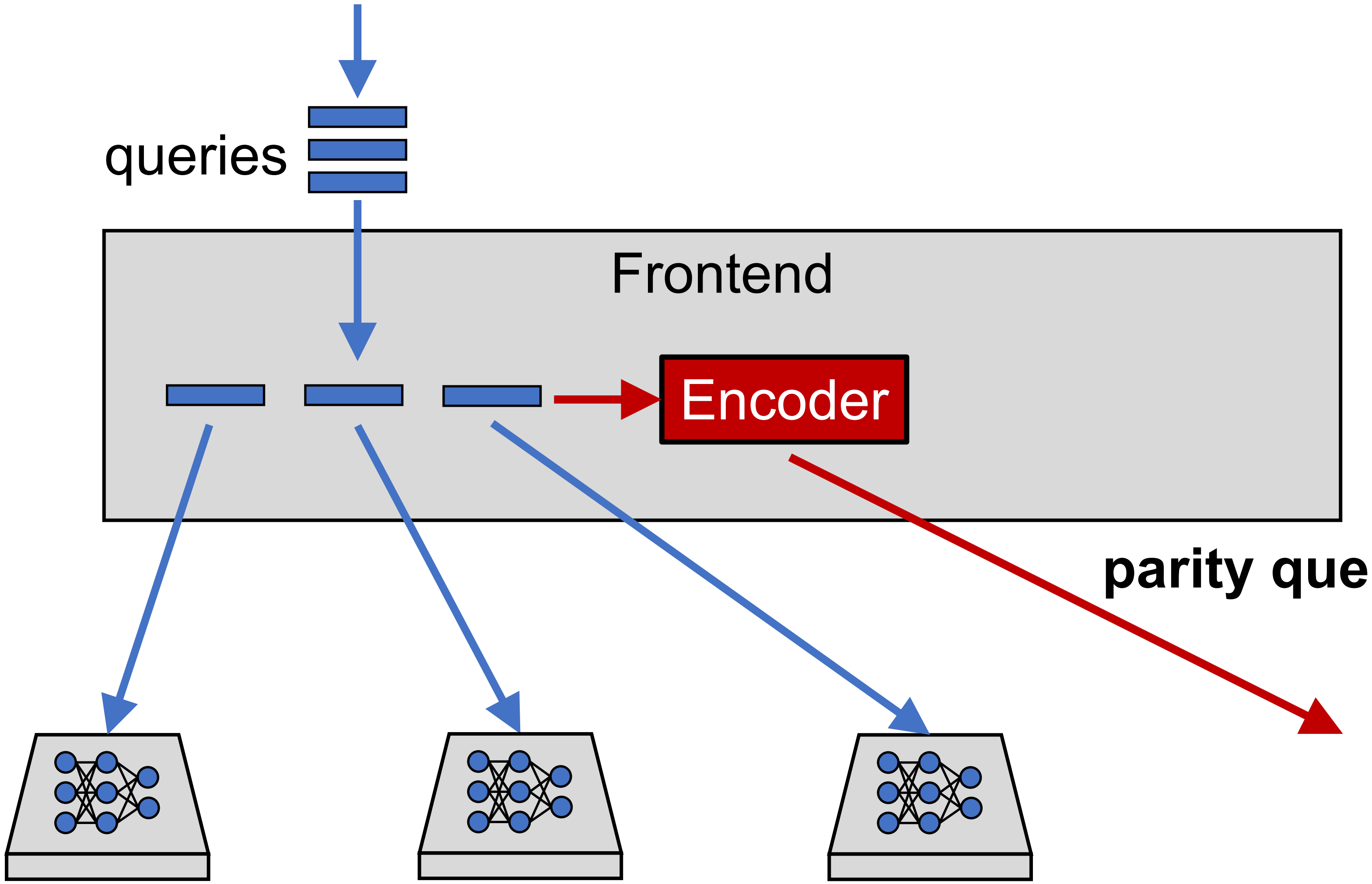
End goal: erasure-coded prediction serving



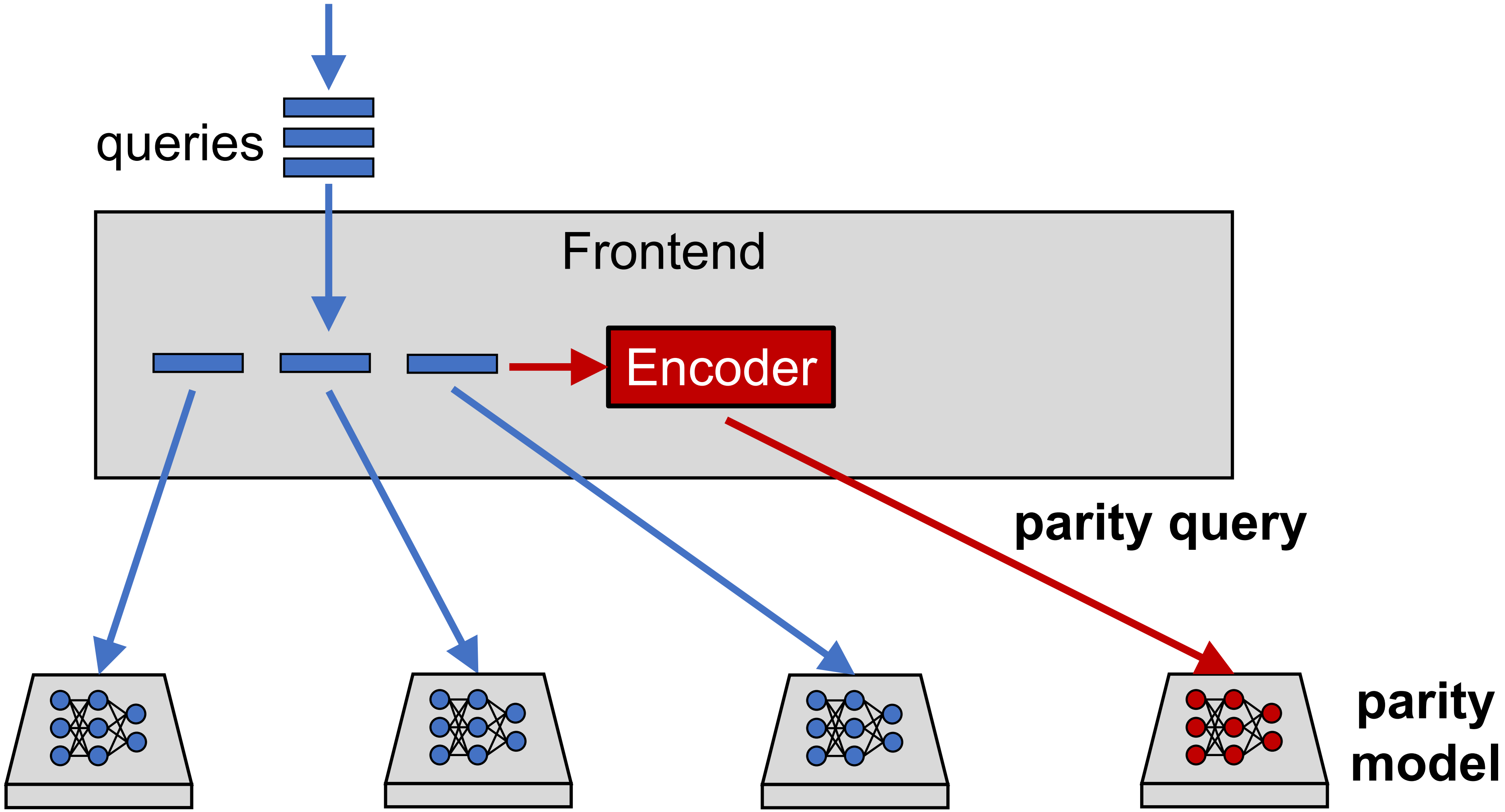
End goal: erasure-coded prediction serving



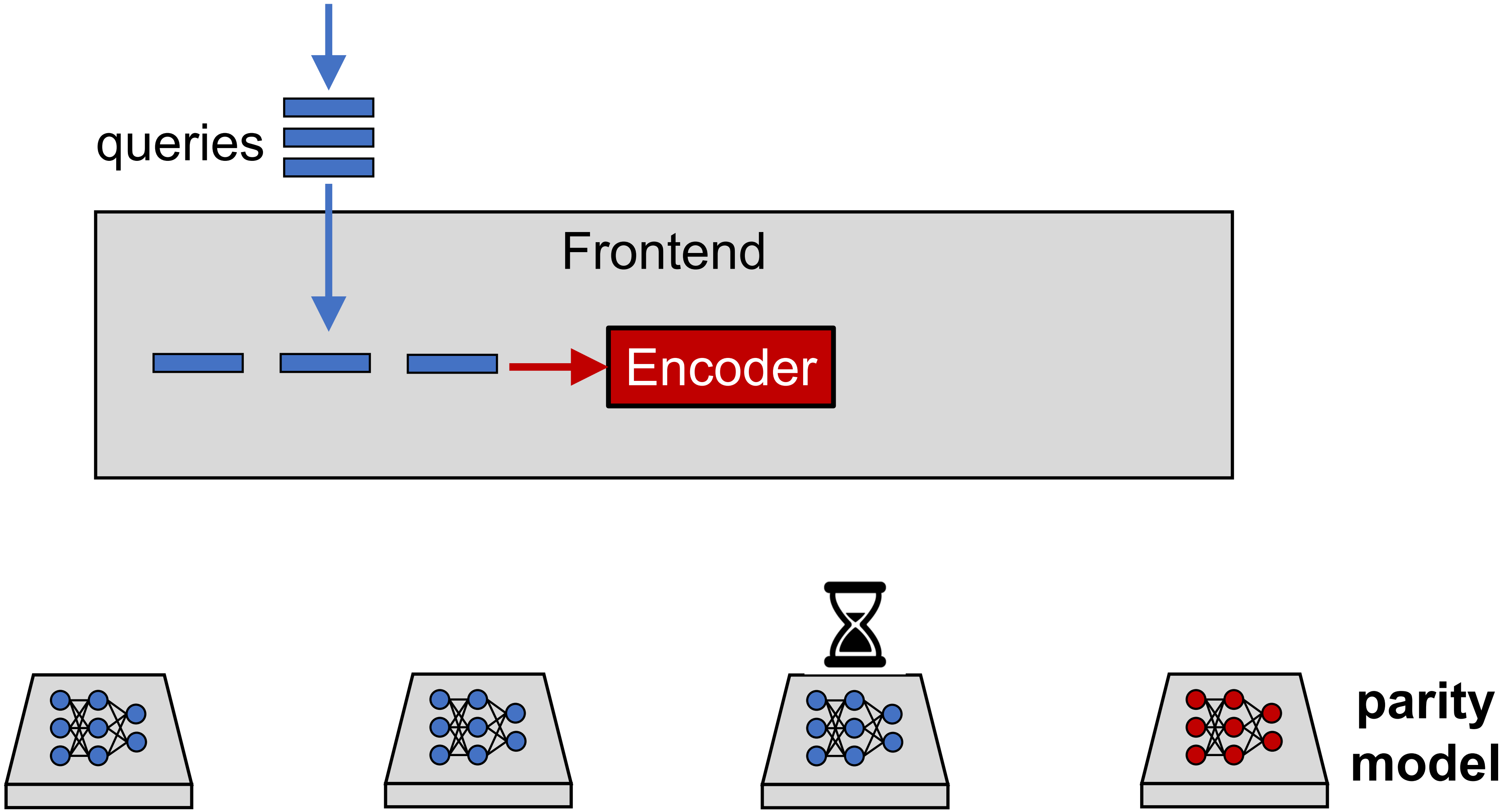
End goal: erasure-coded prediction serving



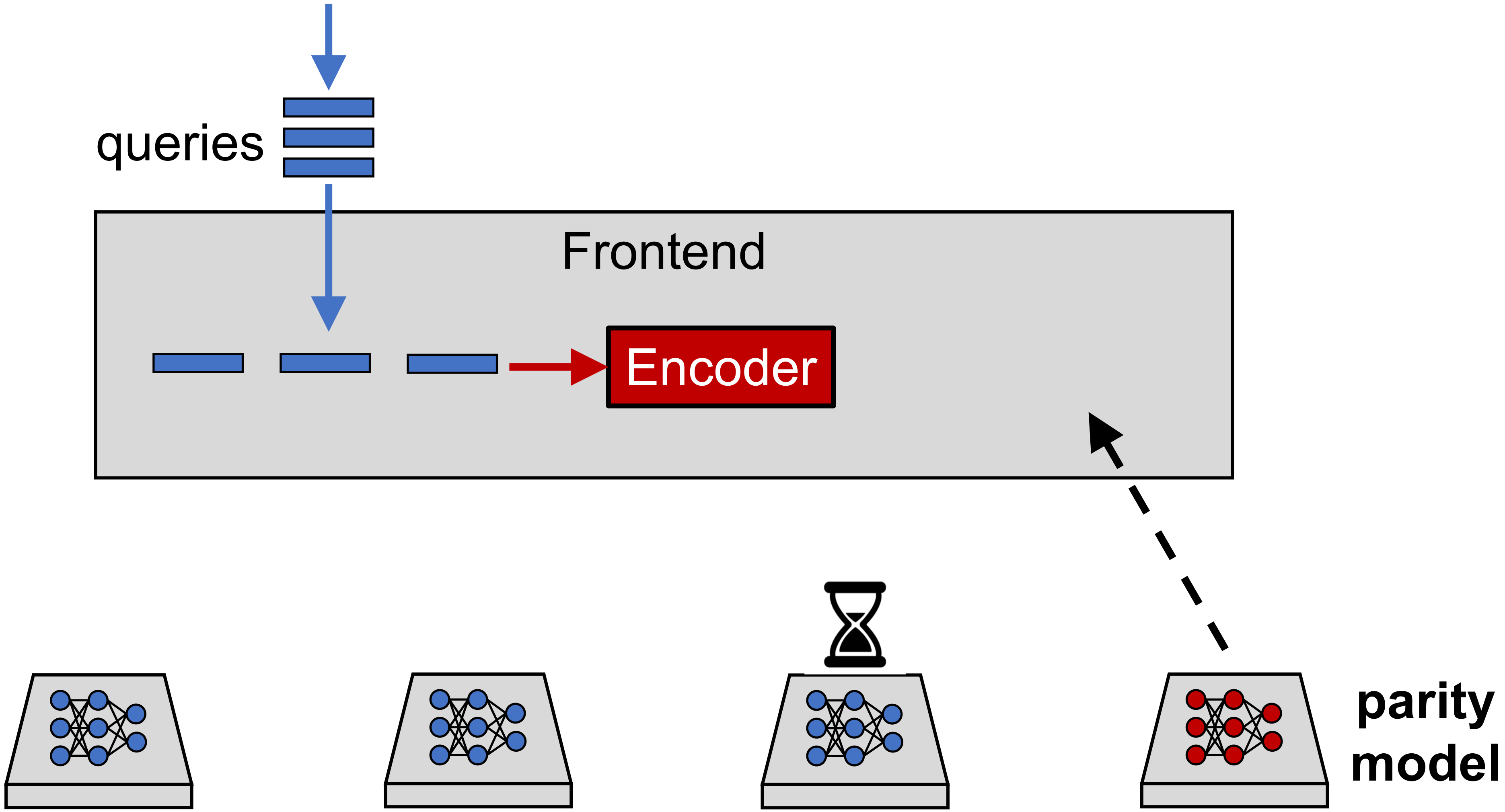
End goal: erasure-coded prediction serving



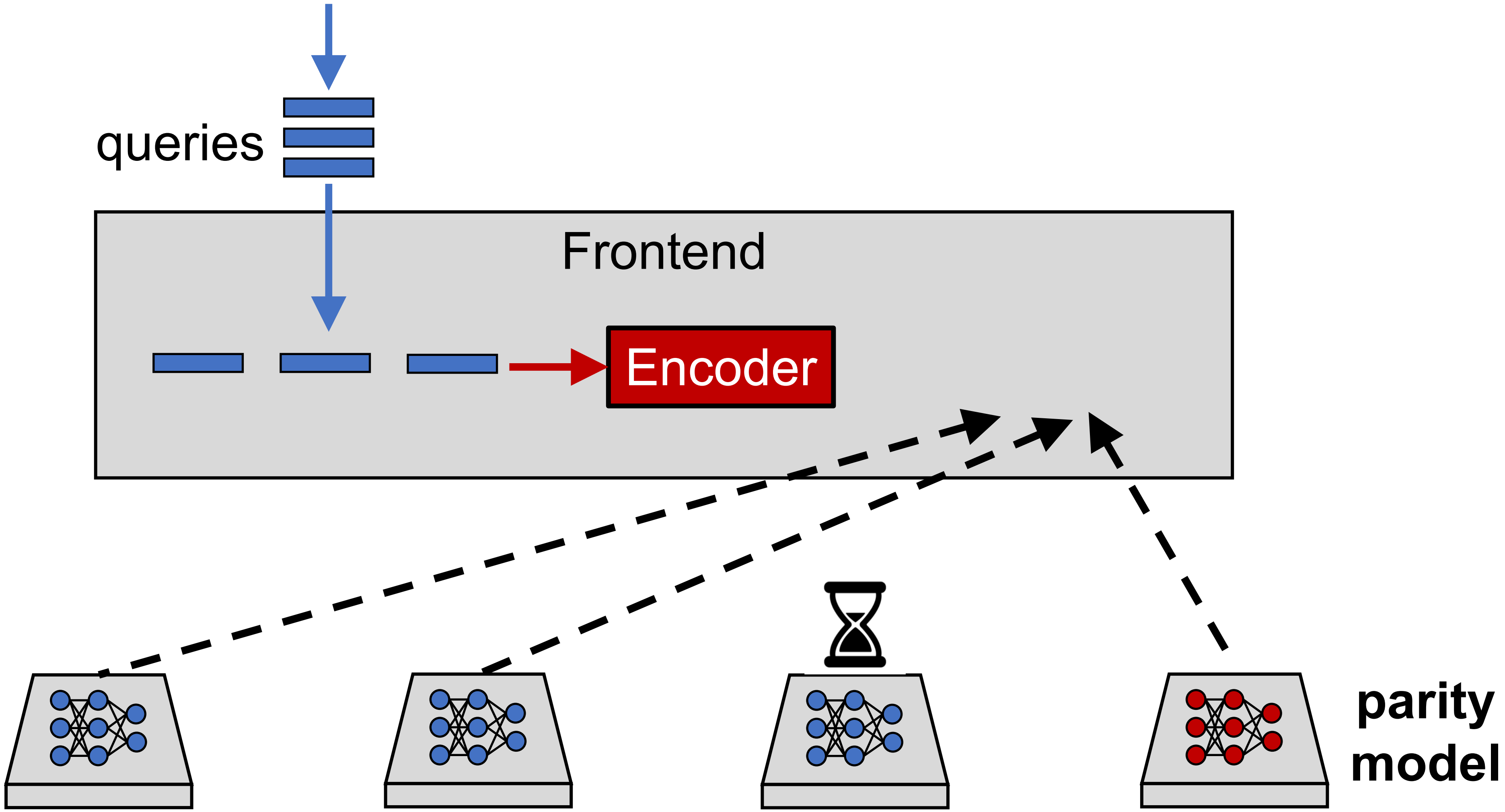
End goal: erasure-coded prediction serving



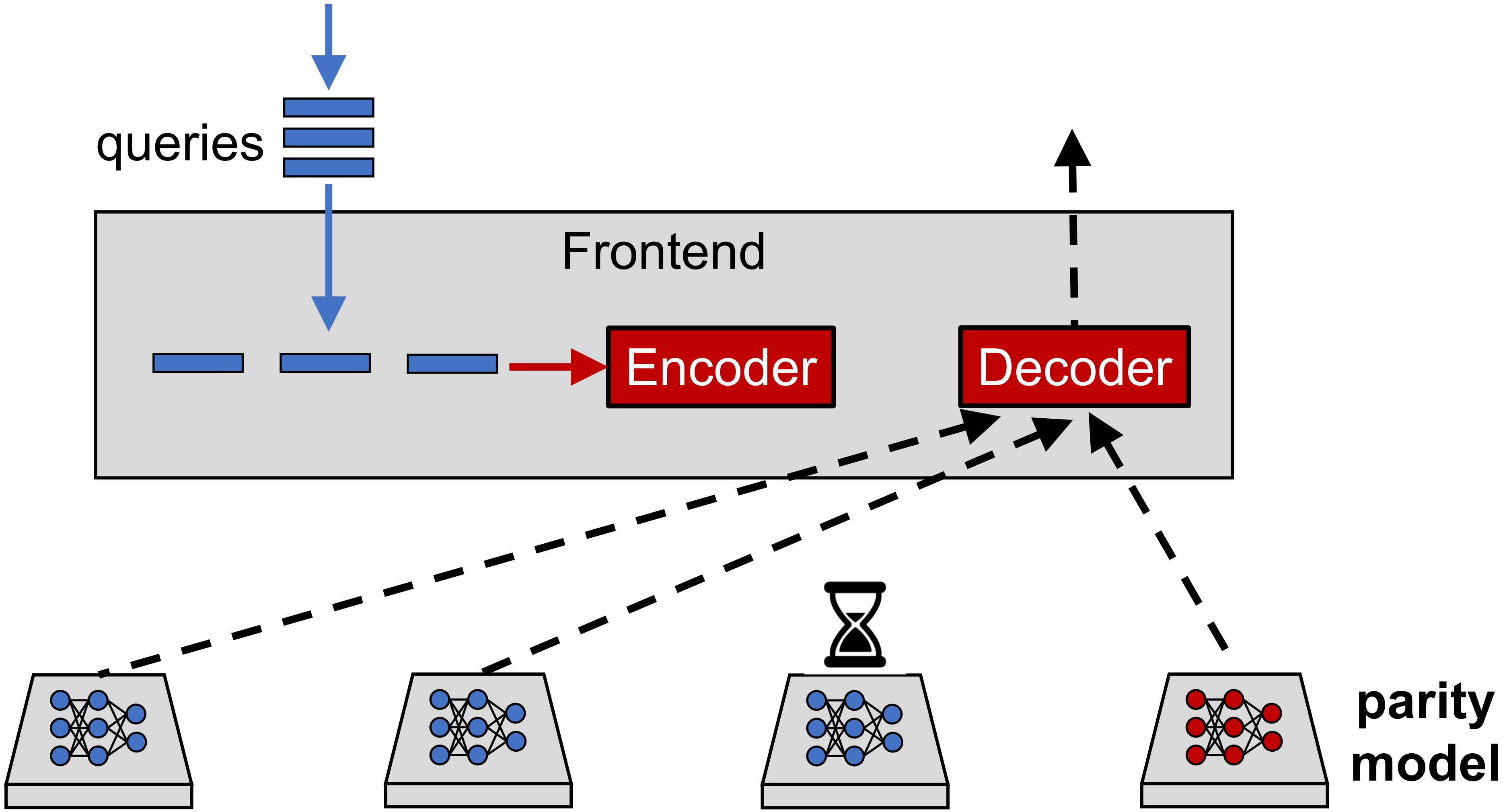
End goal: erasure-coded prediction serving



End goal: erasure-coded prediction serving



End goal: erasure-coded prediction serving

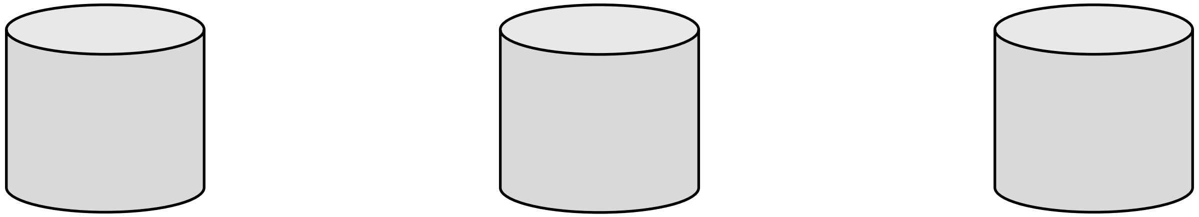


What does it mean to use erasure codes for ML inference?

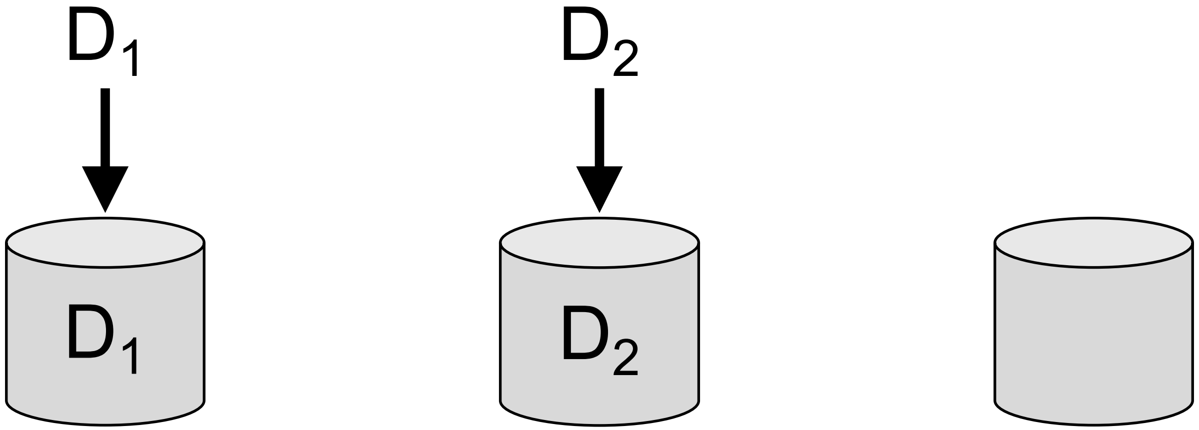
Why is this hard?

Quick recap of erasure codes

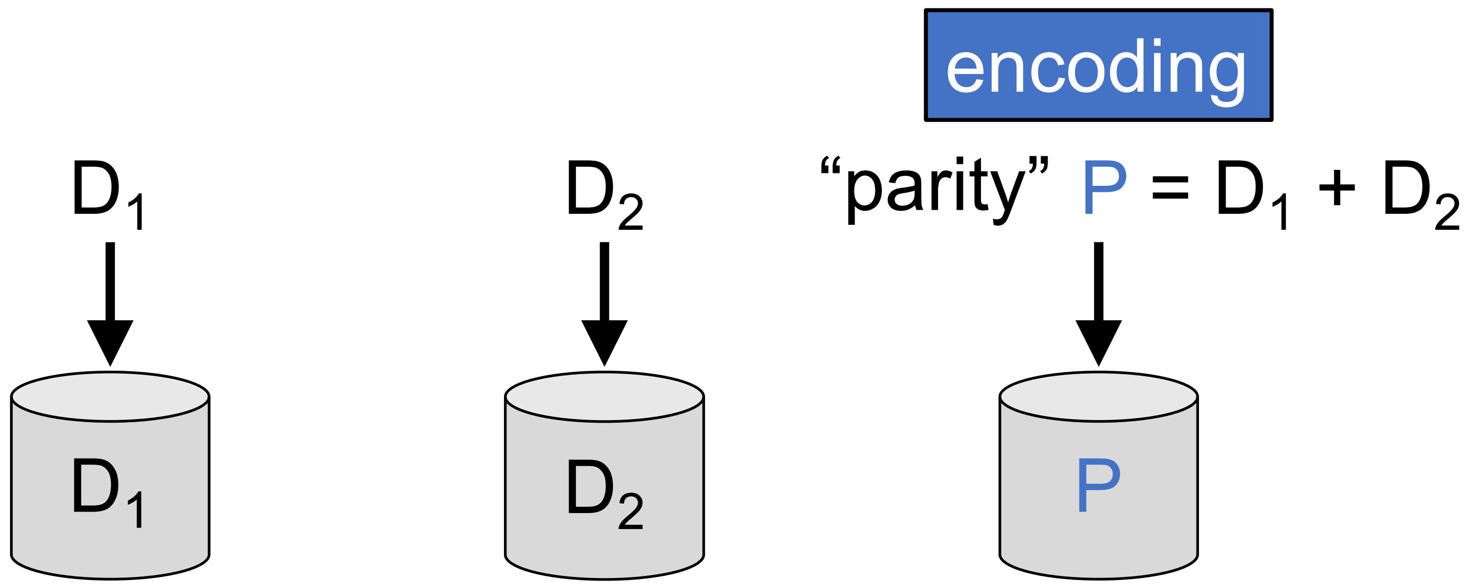
Quick recap of erasure codes



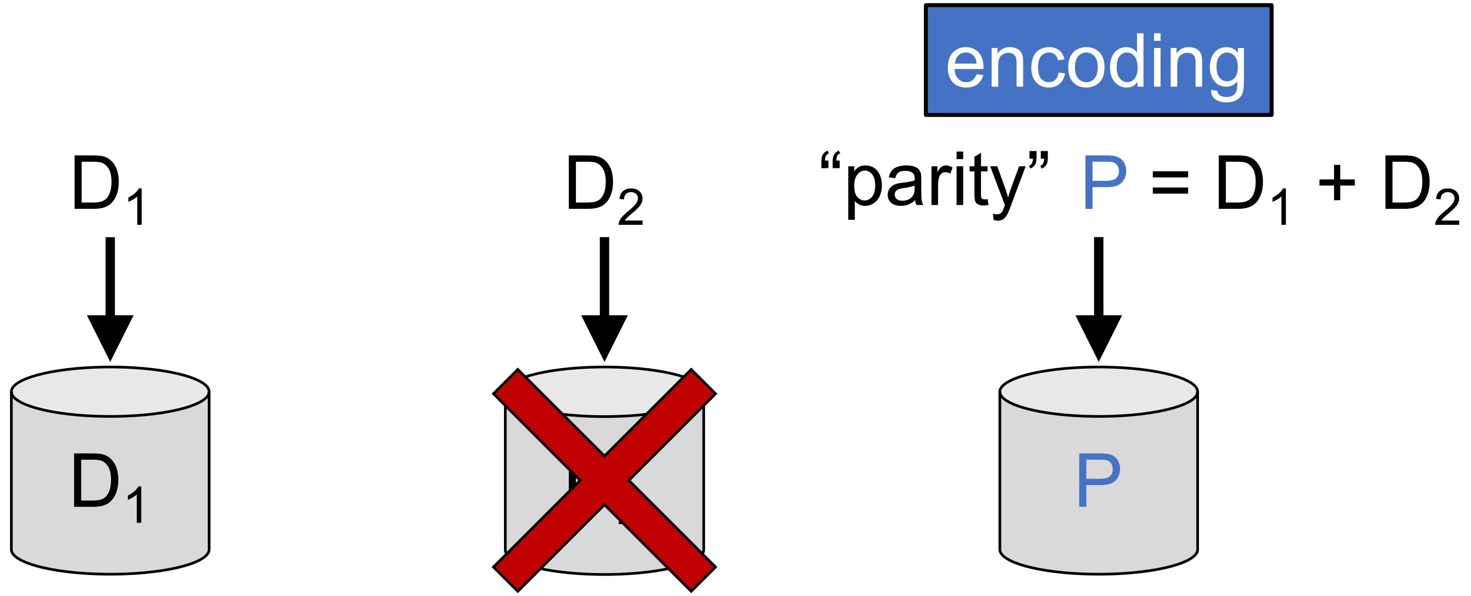
Quick recap of erasure codes



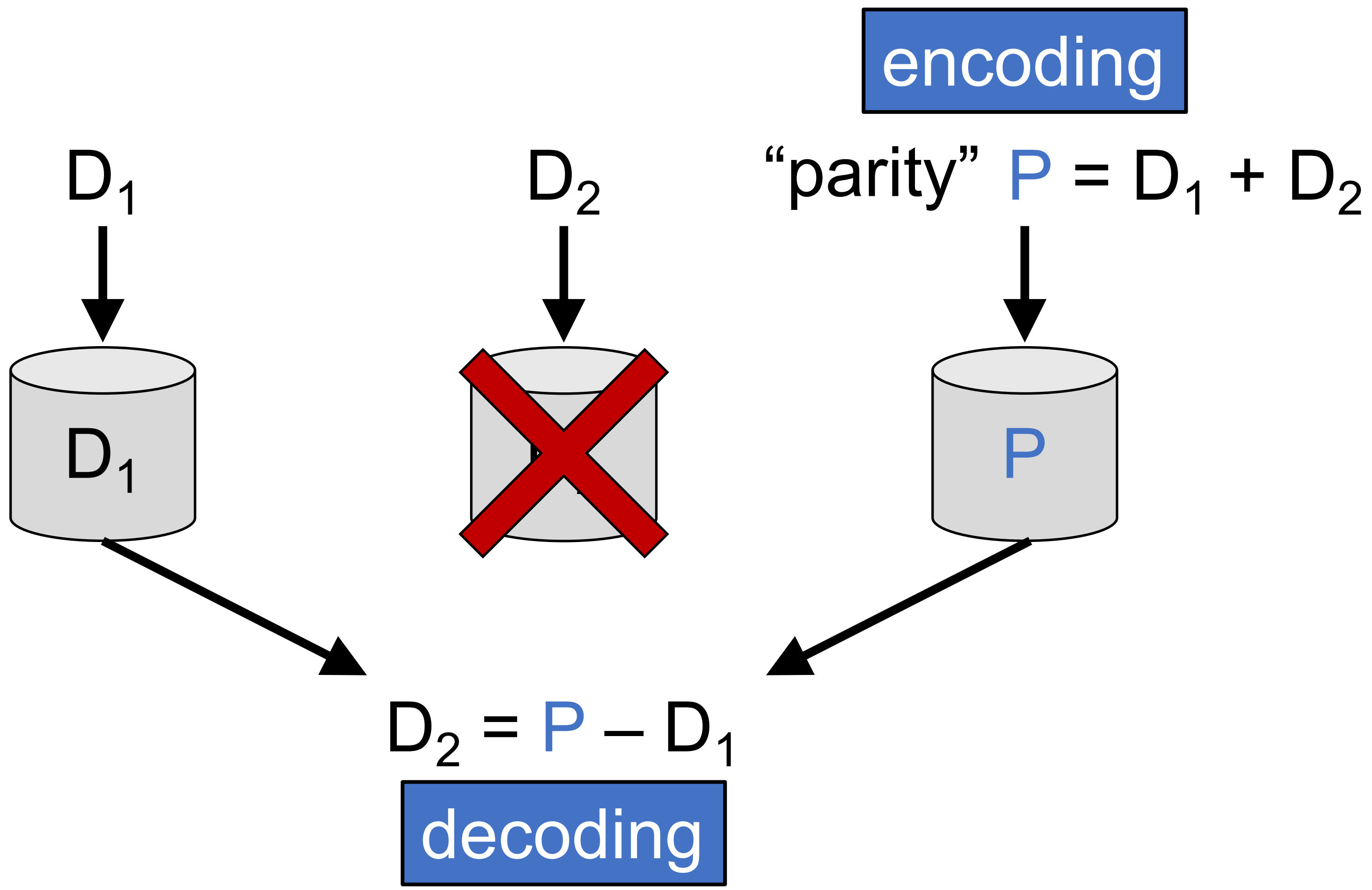
Quick recap of erasure codes



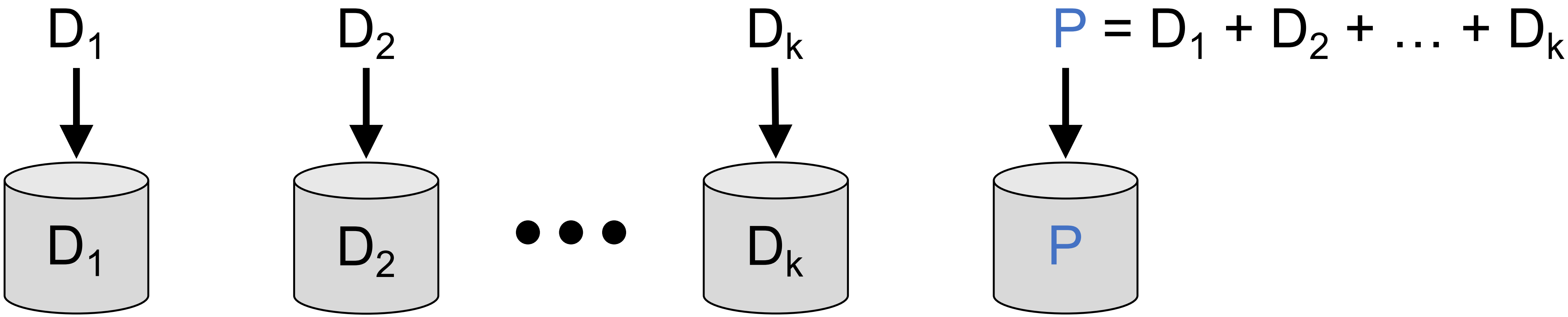
Quick recap of erasure codes



Quick recap of erasure codes

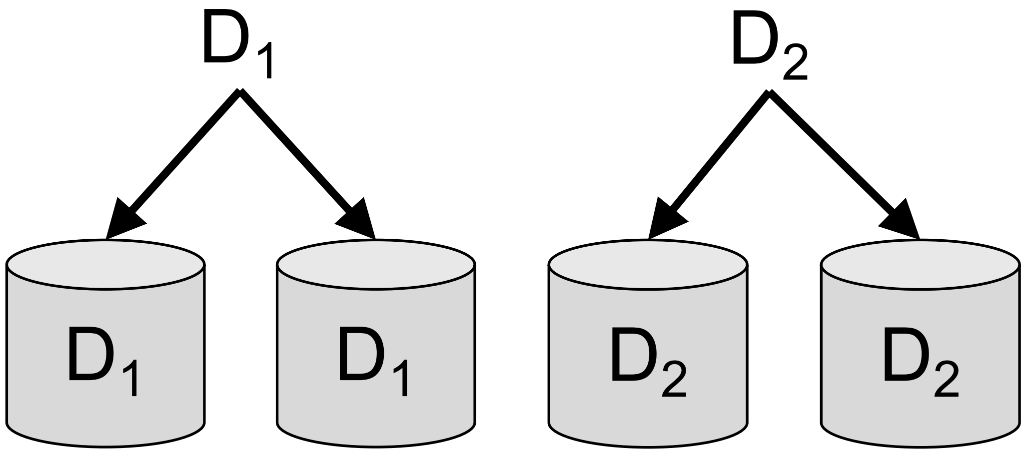


Quick recap of erasure codes: parameter k

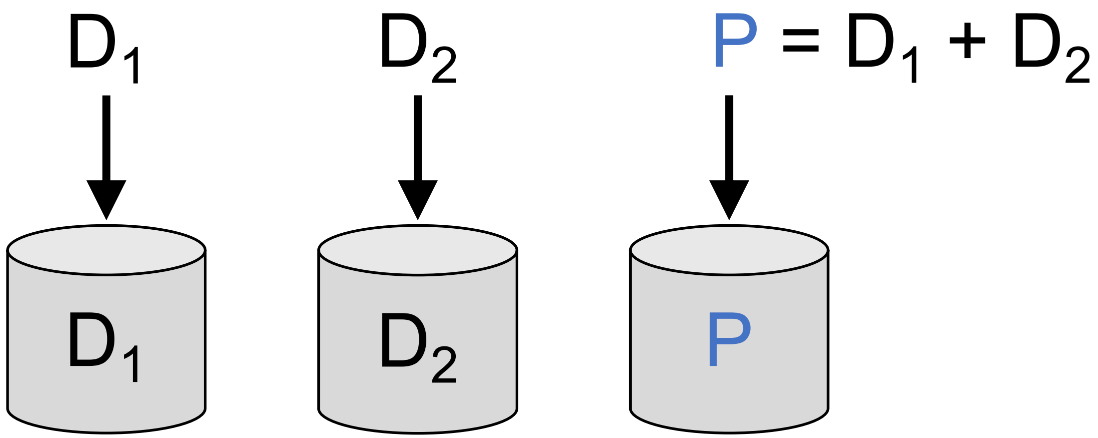


Quick recap of erasure codes: benefits

Replication



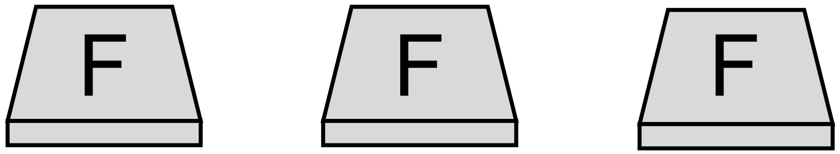
Erasure Coding



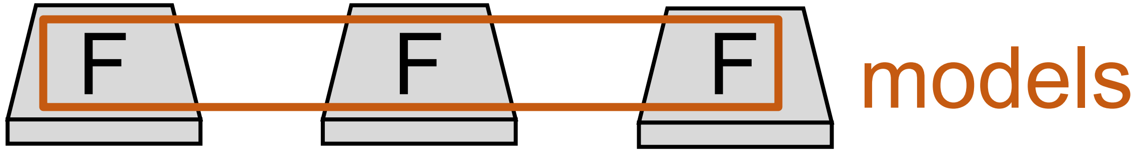
- same resilience
- lower resource-overhead

Using erasure codes for inference

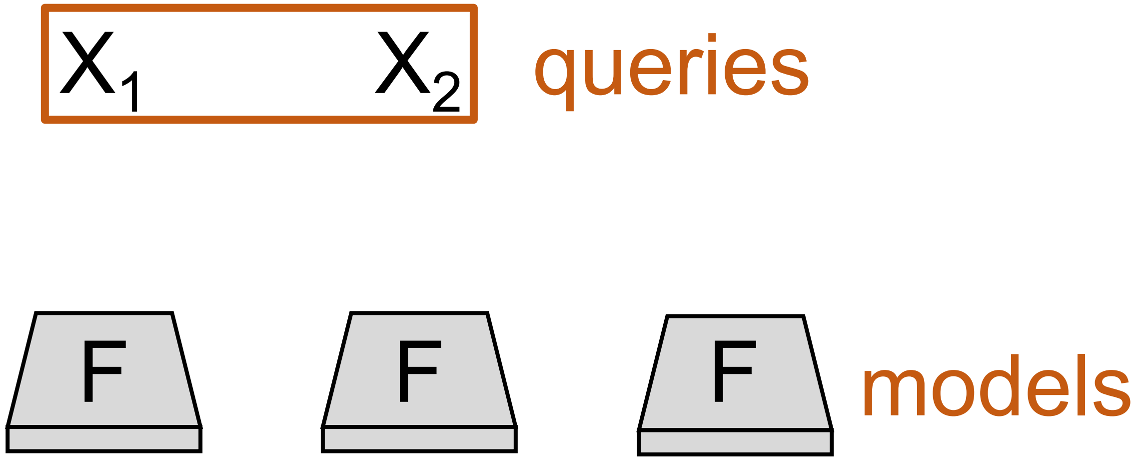
Using erasure codes for inference



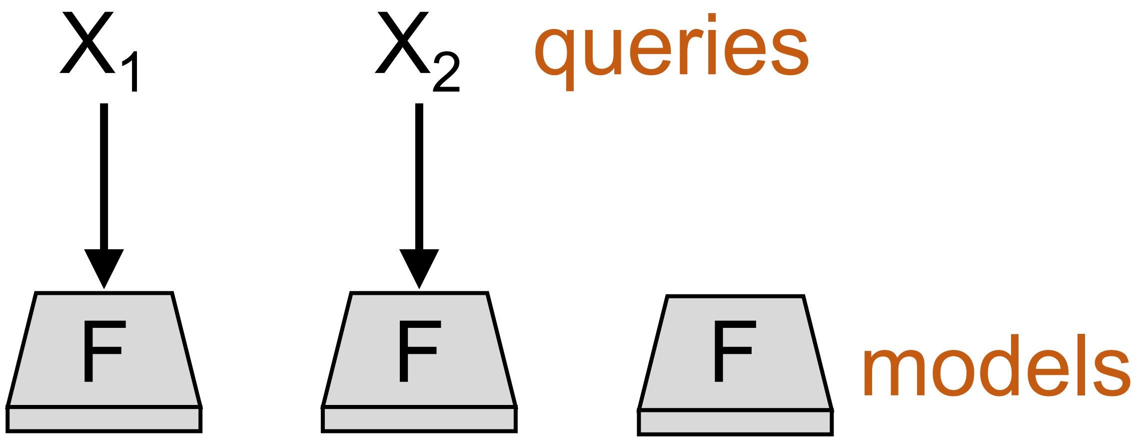
Using erasure codes for inference



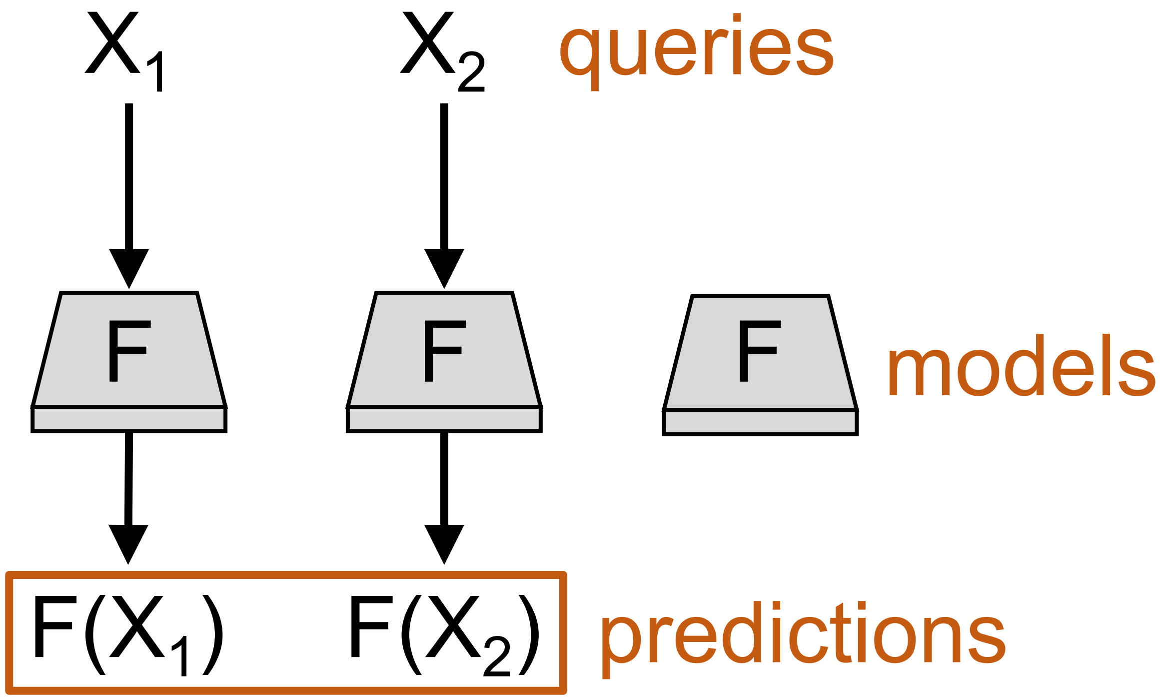
Using erasure codes for inference



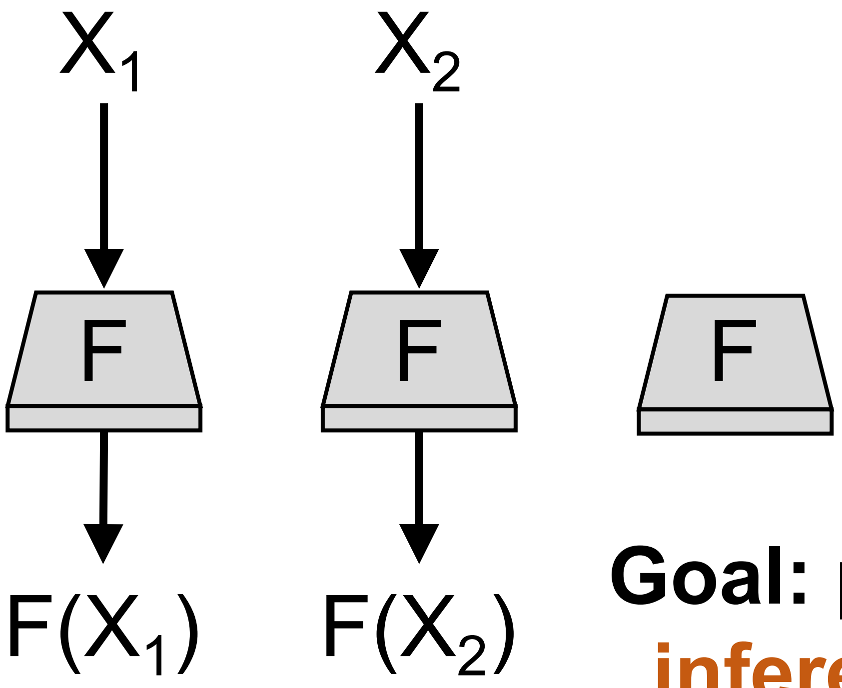
Using erasure codes for inference



Using erasure codes for inference

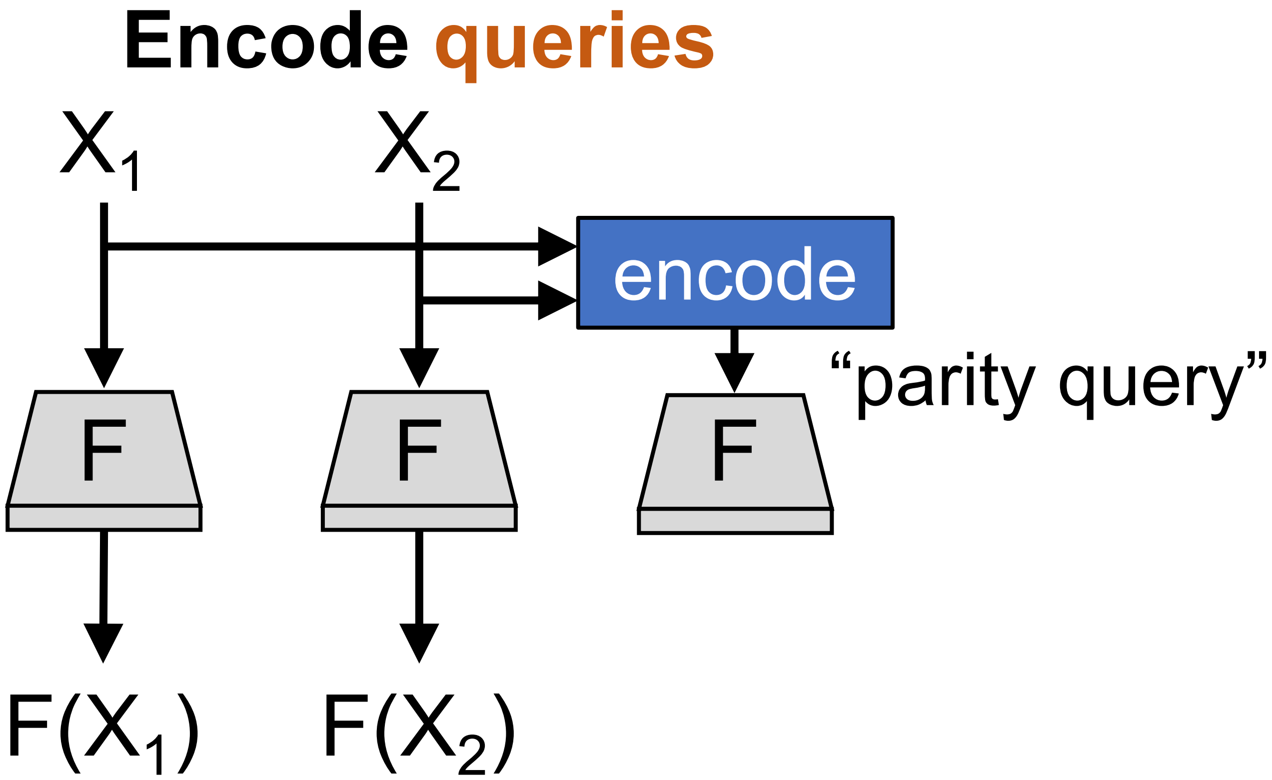


Using erasure codes for inference

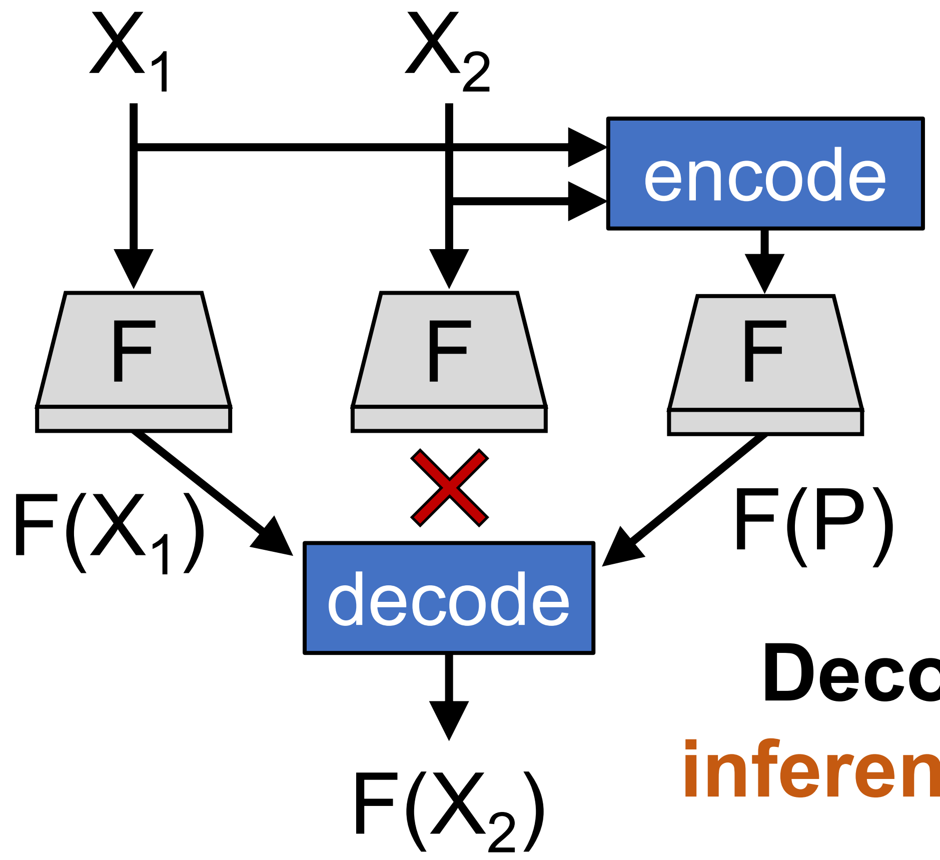


Goal: preserve results of inference over queries

Using erasure codes for inference

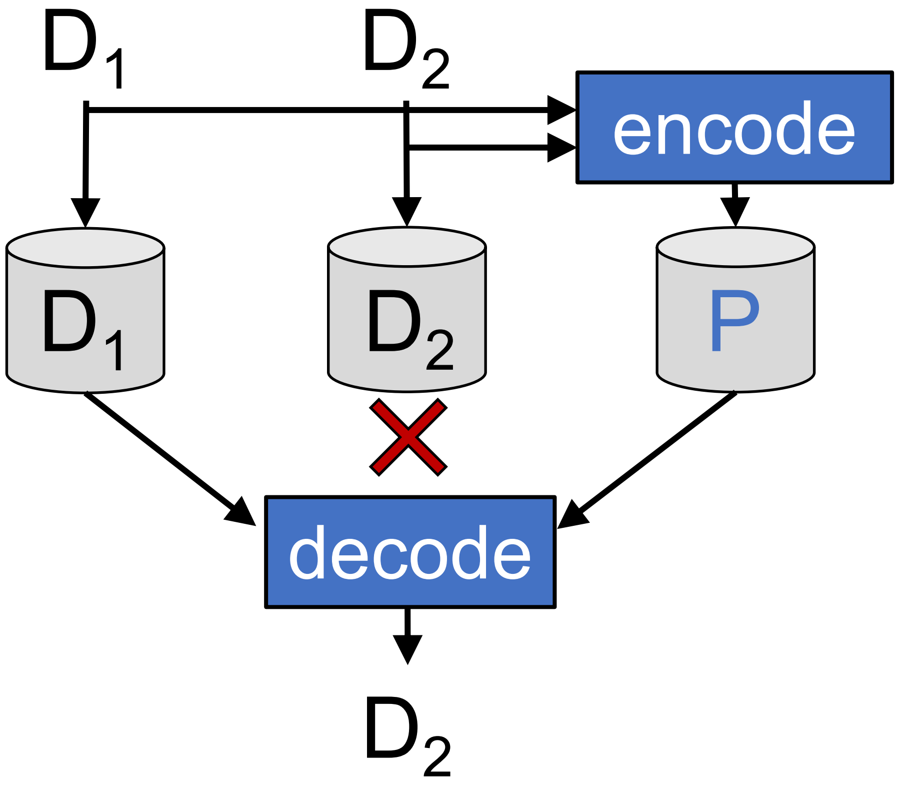


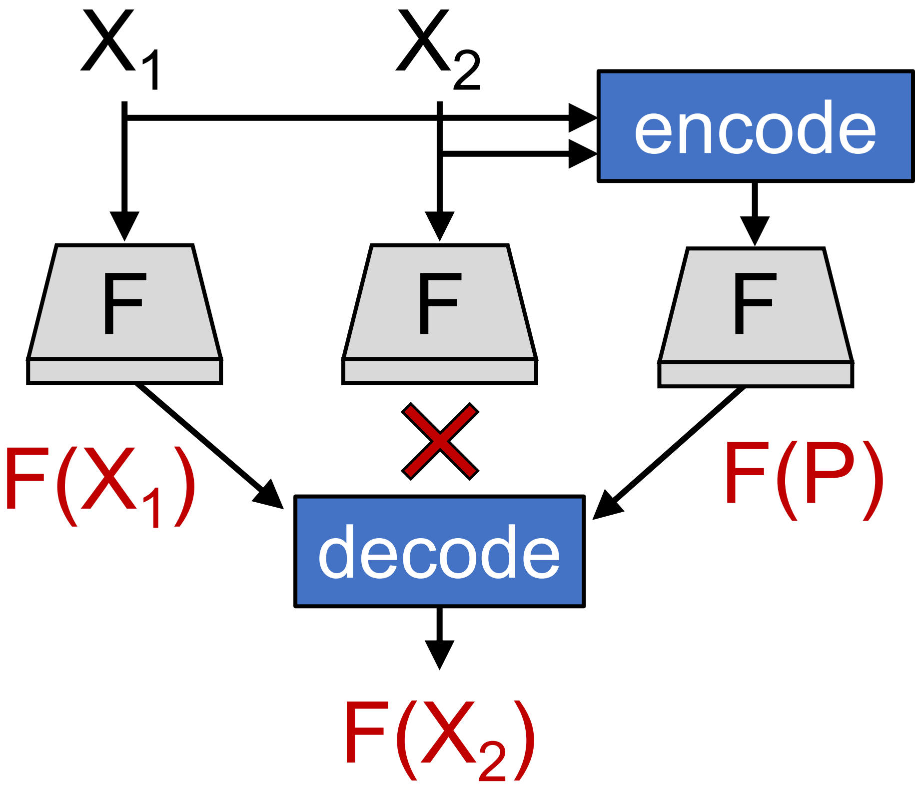
Using erasure codes for inference



Decode results of
inference over queries

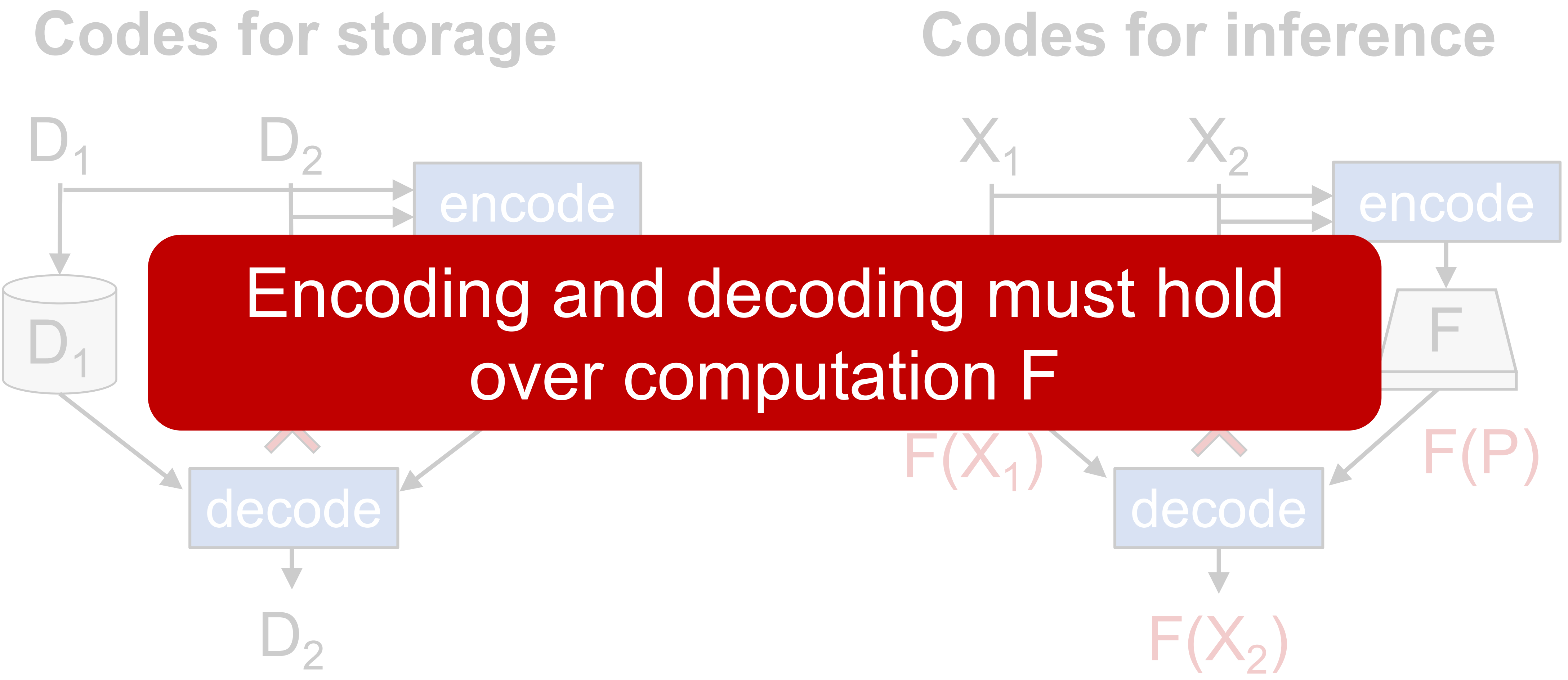
Traditional coding vs. codes for inference





Need to handle **computation over inputs**

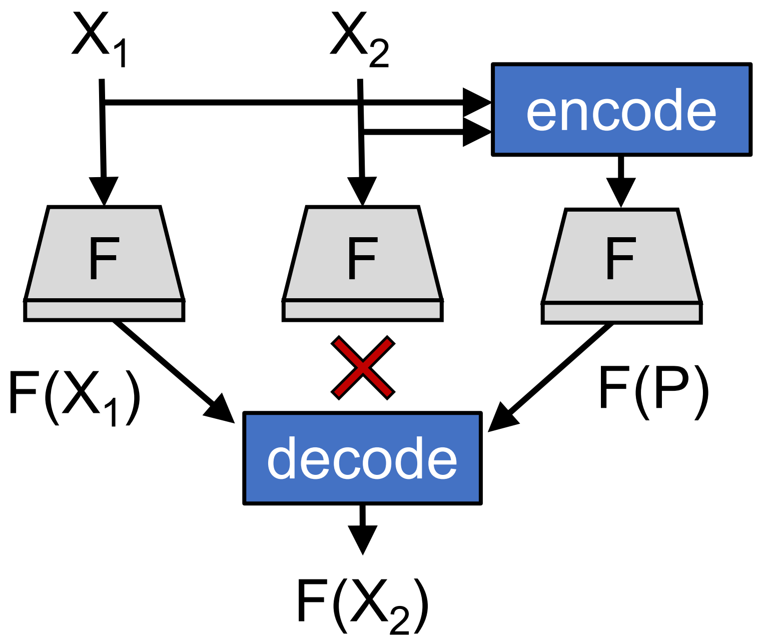
Traditional coding vs. codes for inference



Encoding and decoding must hold over computation F

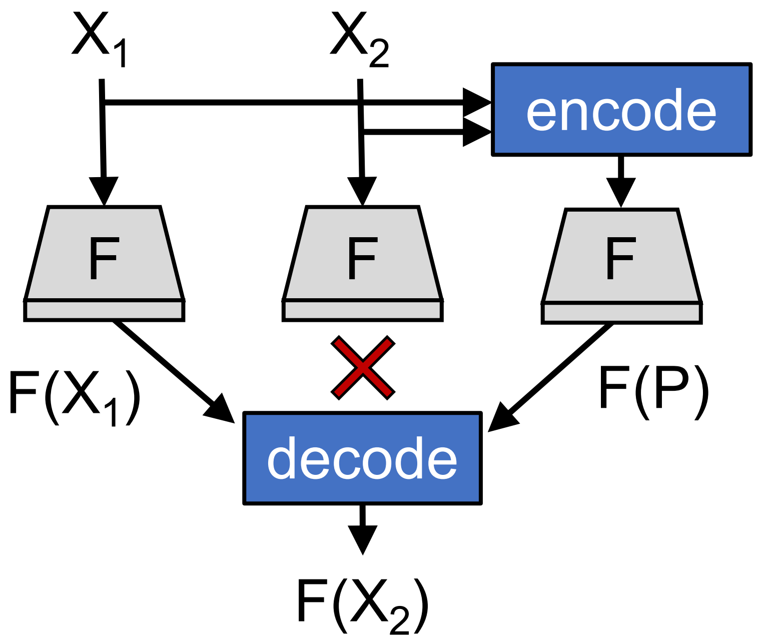
Need to handle **computation over inputs**

Designing erasure codes for inference is hard



Designing erasure codes for inference is hard

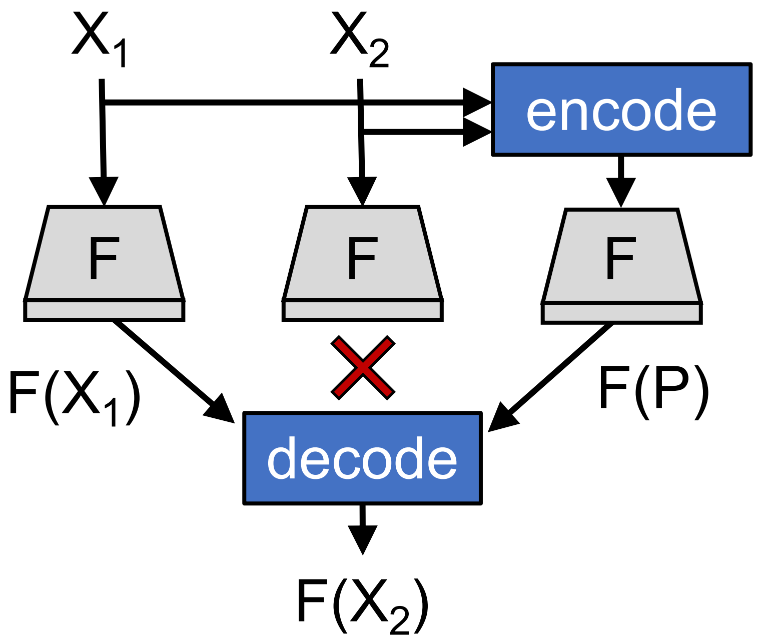
Theoretical framework: “coded-computation”



Designing erasure codes for inference is hard

Theoretical framework: “coded-computation”

Currently: handcraft erasure code

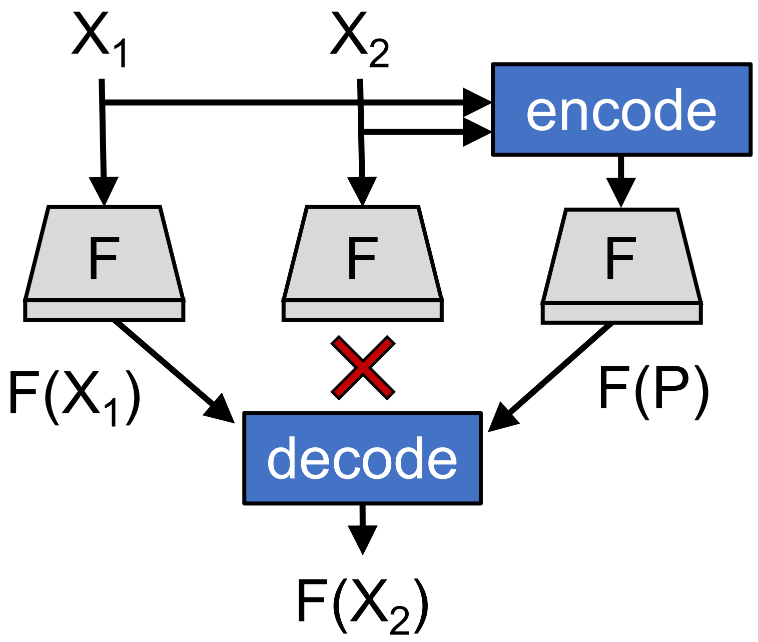


Designing erasure codes for inference is hard

Theoretical framework: “coded-computation”

Currently: handcraft erasure code

- Straight-forward for **linear** F

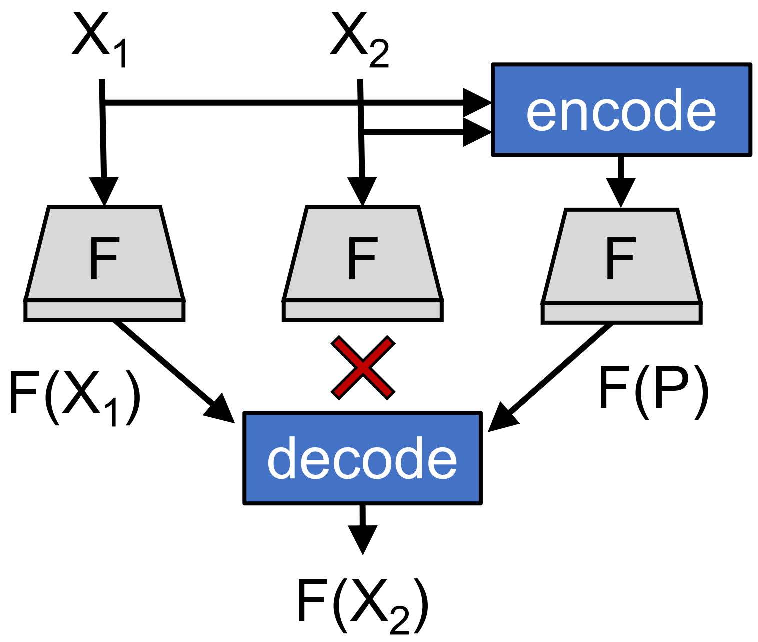


Designing erasure codes for inference is hard

Theoretical framework: “coded-computation”

Currently: handcraft erasure code

- Straight-forward for **linear** F
- Far more challenging for **non-linear** F
  - Apply to only restricted functions (polynomials)
  - Require 2x resource-overhead



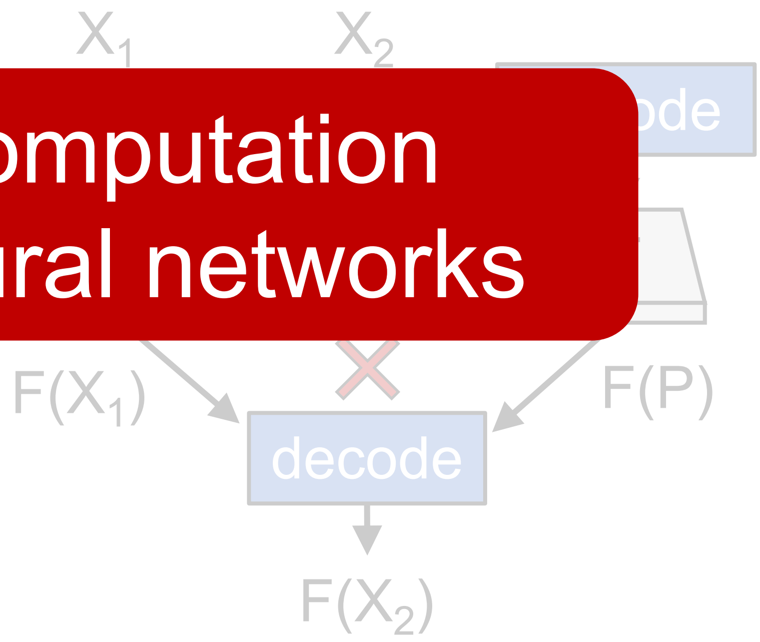
Designing erasure codes for inference is hard

Theoretical framework: “coded-computation”

Currently: handcraft erasure code

Current handcrafted coded-computation approaches cannot support neural networks

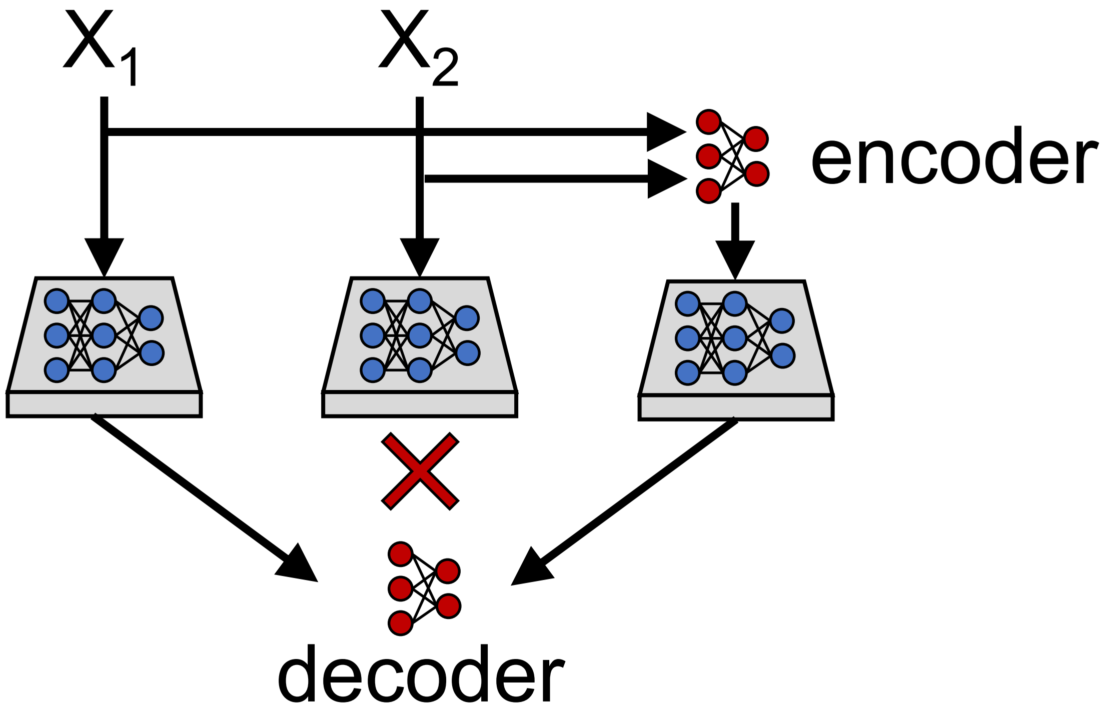
- Apply to only restricted functions (polynomials)
- Require 2x resource-overhead



This work:
overcome challenges of handcrafting erasure codes for coded-computation by taking a **learning-based** approach to erasure-coded resilience

Learning an erasure code?

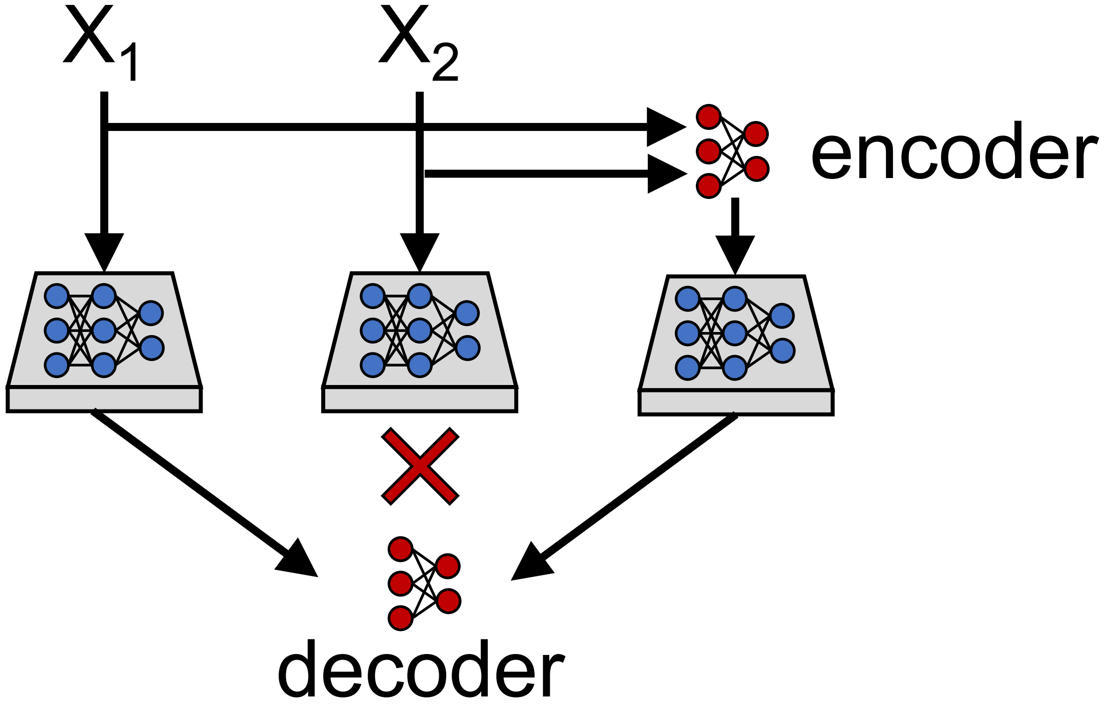
Design encoder and decoder as neural networks



Learning an erasure code?

Design encoder and decoder as neural networks

Accurate

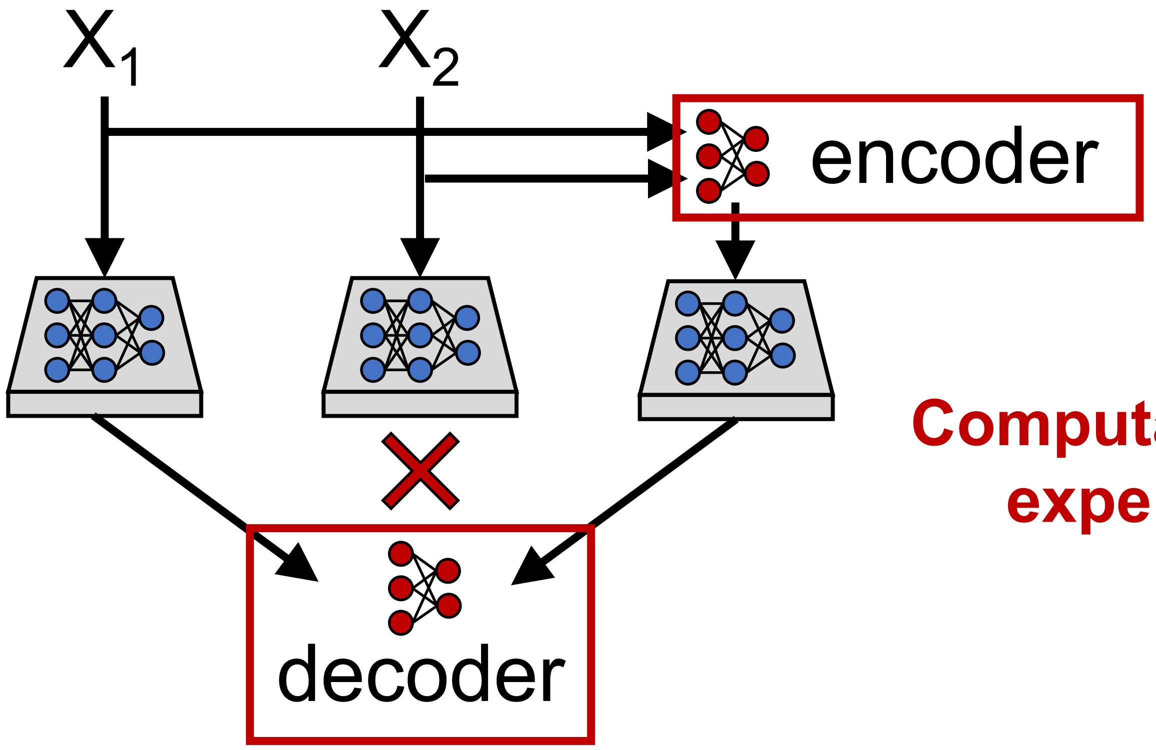


Learning an erasure code?

Design encoder and decoder as neural networks

Accurate

Expensive encoder/decoder



Computationally expensive

Learn computation over parities

Learn computation over parities

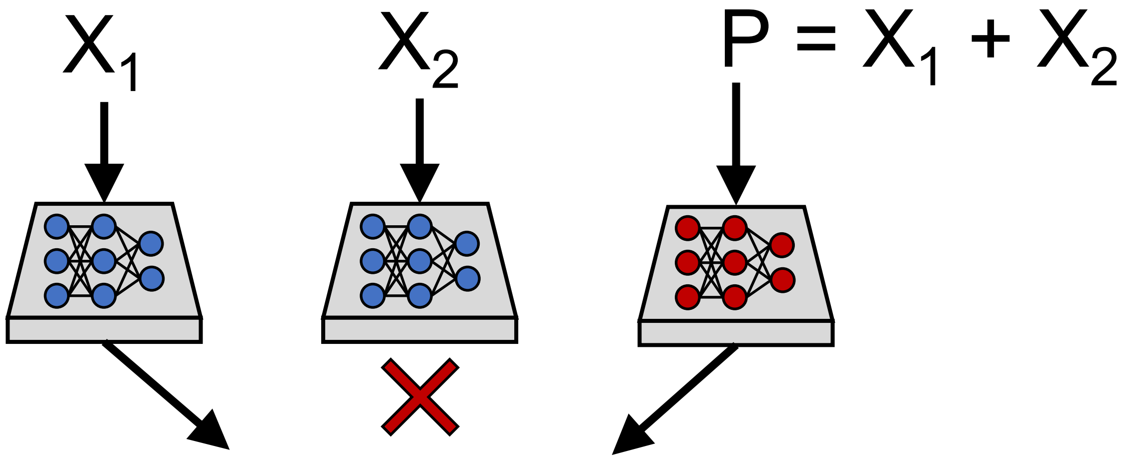
Use simple, fast encoders and decoders

Learn computation over parities: **“parity model”**

Learn computation over parities

Use simple, fast encoders and decoders

Learn computation over parities: **“parity model”**

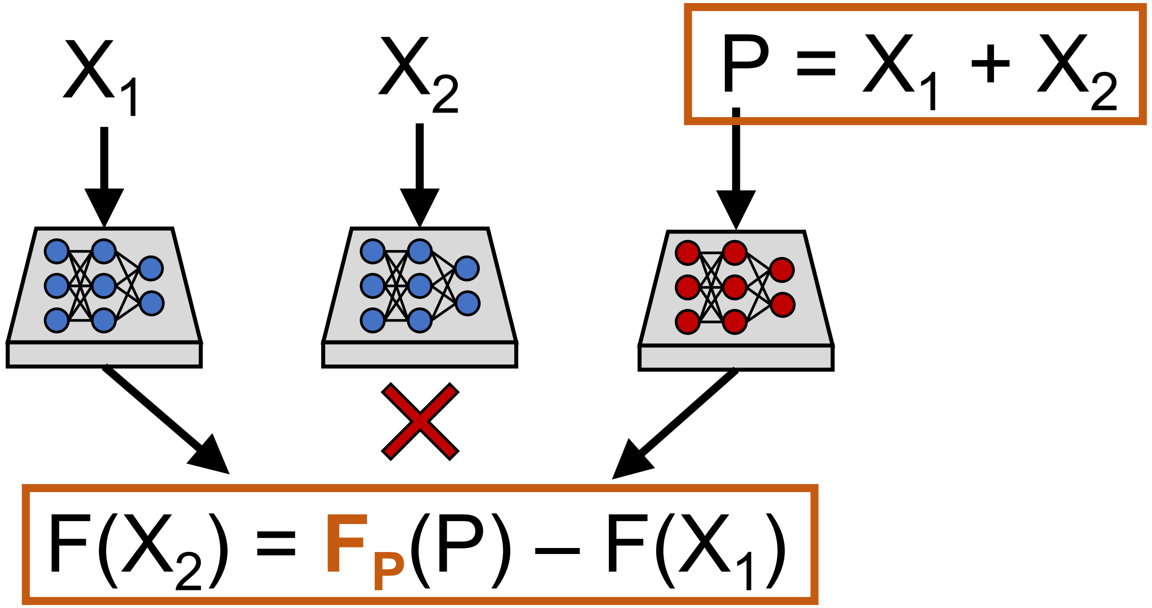


$$F(X_2) = F_P(P) - F(X_1)$$

Learn computation over parities

Use simple, fast encoders and decoders

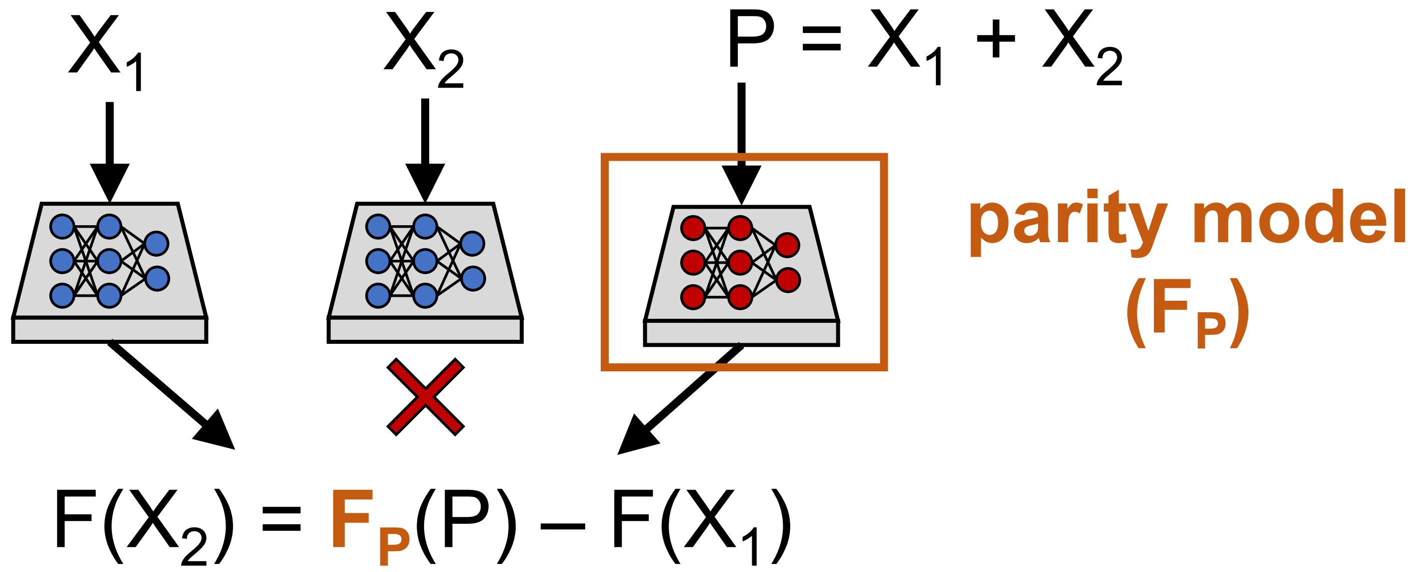
Learn computation over parities: **"parity model"**



Learn computation over parities

Use simple, fast encoders and decoders

Learn computation over parities: **"parity model"**



Learn computation over parities

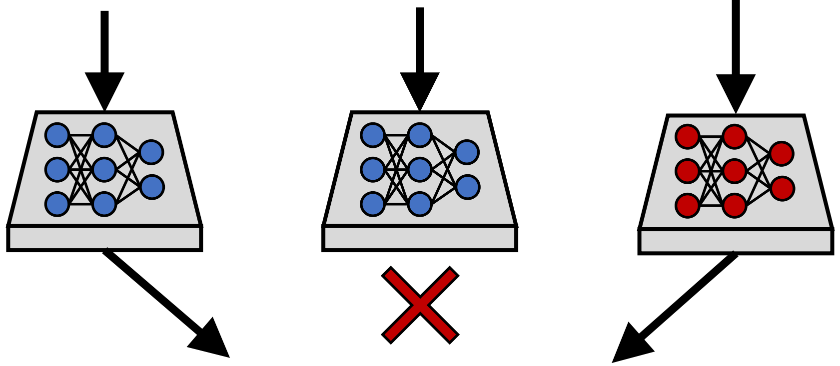
Use simple, fast encoders and decoders

Learn computation over parities: **“parity model”**

Accurate

Efficient encoder/decoder

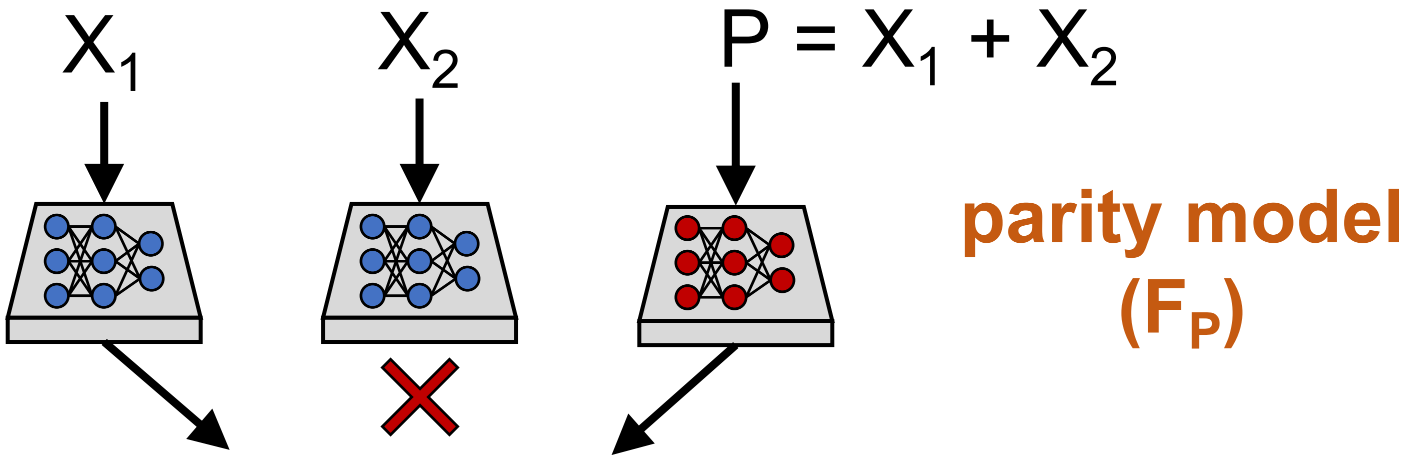
X_1 $\quad$ X_2 $\quad$ $P = X_1 + X_2$



parity model (F_P)

$$F(X_2) = F_P(P) - F(X_1)$$

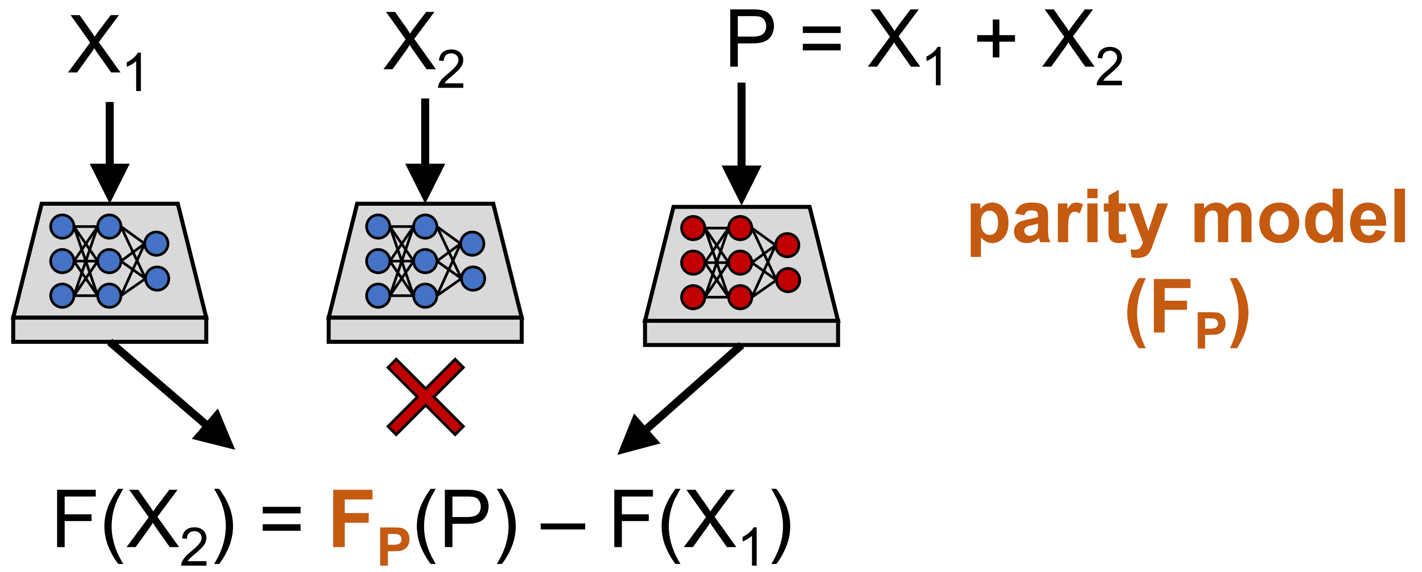
Designing parity models



$$F(X_2) = \mathbf{F_P}(P) - F(X_1)$$

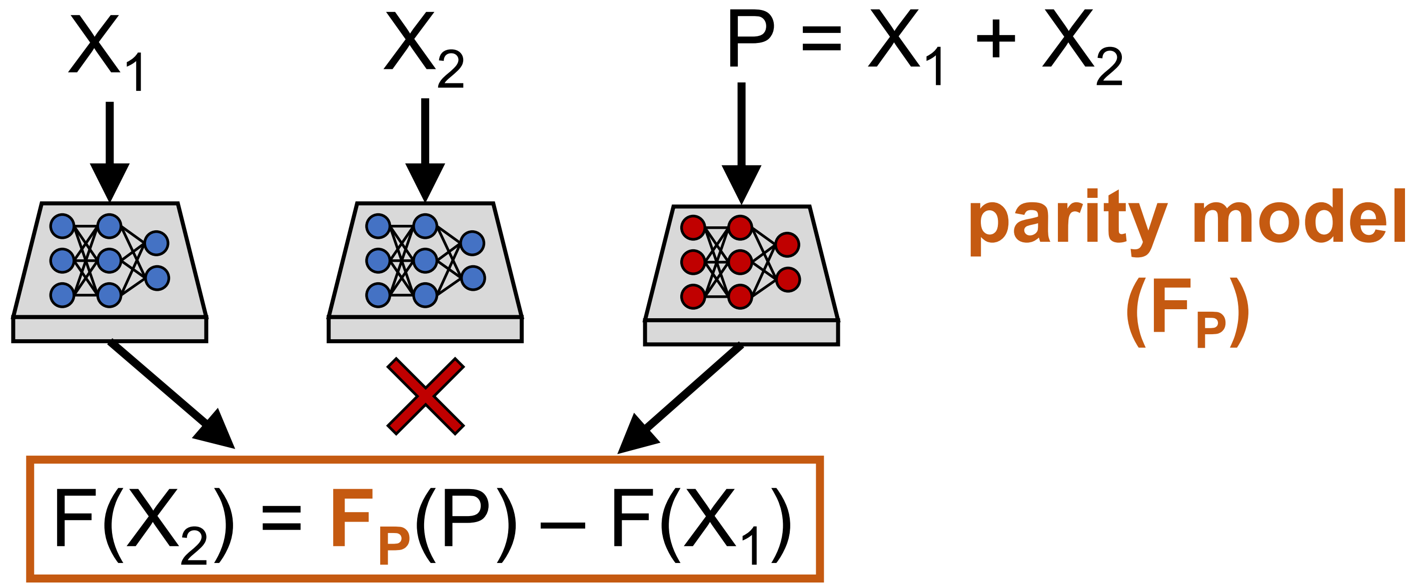
Designing parity models

Goal: transform parities into a form that enables decoder to reconstruct unavailable predictions



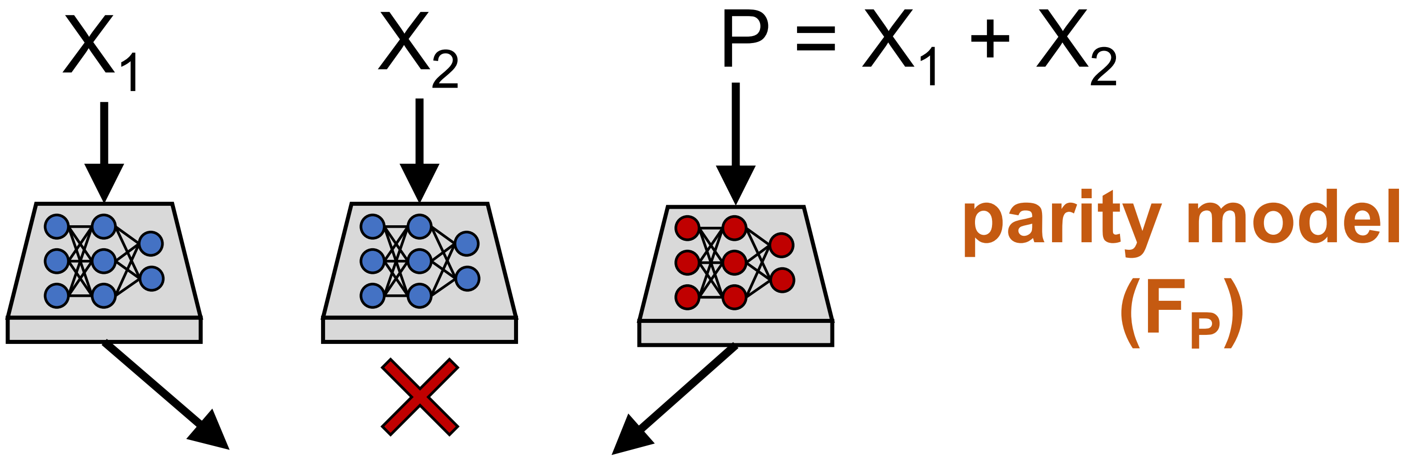
Designing parity models

Goal: transform parities into a form that enables decoder to reconstruct unavailable predictions



Designing parity models

Goal: transform parities into a form that enables decoder to reconstruct unavailable predictions

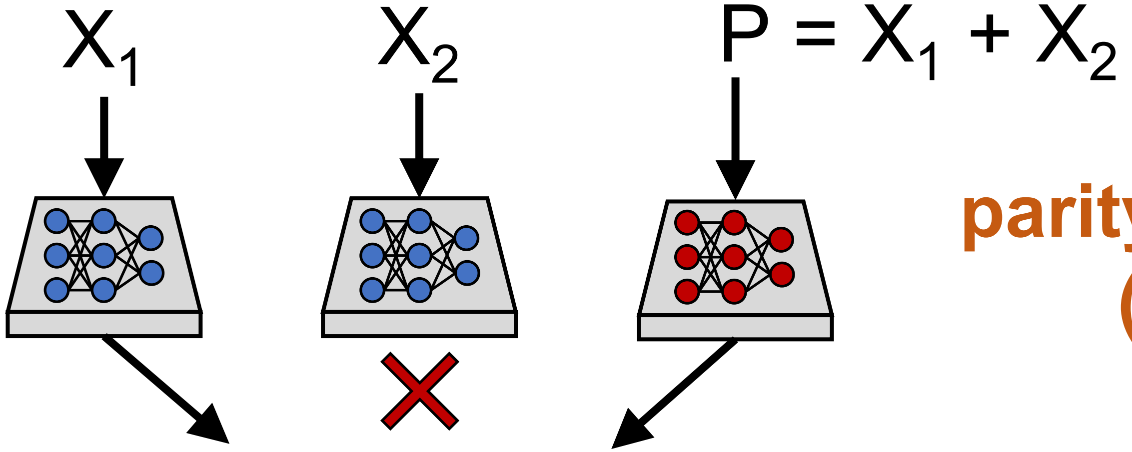


$$F(X_2) = \mathbf{F_P}(P) - F(X_1)$$

$$\boxed{\mathbf{F_P}(P) = F(X_1) + F(X_2)}$$

Designing parity models

Goal: transform parities into a form that enables decoder to reconstruct unavailable predictions



parity model (F_P)

$F(X_2) = \mathbf{F_P}(P) - F(X_1)$

$\mathbf{F_P}(P) = F(X_1) + F(X_2)$

Learn a parity model

Training a parity model

$P = X_1 + X_2$

Desired output: $F(X_1) + F(X_2)$

Training a parity model

1. Sample inputs and encode

queries

$P = X_1 + X_2$

Desired output: $F(X_1) + F(X_2)$

Training a parity model

1. Sample inputs and encode

queries

$$P = X_1 + X_2$$

Desired output: $F(X_1) + F(X_2)$

predictions

0.8	0.15	0.05	0.2	0.7	0.1

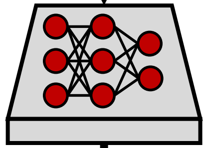
Training a parity model

queries

1. Sample inputs and encode

2. Perform inference with parity model

$P = X_1 + X_2$



$F_P(P)_1$

Desired output: $F(X_1) + F(X_2)$

predictions

0.8	0.15	0.05

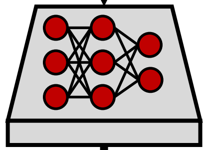
0.2	0.7	0.1

Training a parity model

1. Sample inputs and encode
2. Perform inference with parity model
3. Compute loss

queries

$P = X_1 + X_2$



$F_P(P)_1$

Desired output: $F(X_1) + F(X_2)$

compute loss

predictions

0.8	0.15	0.05

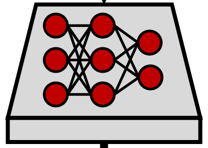
0.2	0.7	0.1

Training a parity model

1. Sample inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropogate loss


queries

$P = X_1 + X_2$



$F_P(P)_1$

Desired output: $F(X_1) + F(X_2)$

compute loss

predictions

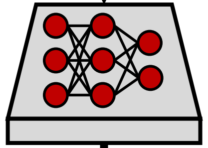
0.8	0.15	0.05	0.2	0.7	0.1

Training a parity model

1. Sample inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropogate loss
5. Repeat

queries

$P = X_1 + X_2$



$F_P(P)_1$

Desired output: $F(X_1) + F(X_2)$

compute loss

predictions

0.8	0.15	0.05

0.2	0.7	0.1

Training a parity model

1. Sample inputs and encode

2. Perform inference with parity model

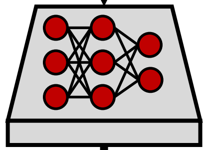
3. Compute loss

4. Backpropogate loss

5. Repeat

queries

$P = X_1 + X_2$



$F_P(P)_2$

Desired output: $F(X_1) + F(X_2)$

compute loss

predictions

0.15	0.8	0.05	0.3	0.5	0.2

Training a parity model

1. Sample inputs and encode

2. Perform inference with parity model

3. Compute loss

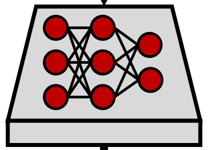
4. Backpropogate loss

5. Repeat

queries

$P = X_1 + X_2$



$F_P(P)_3$

Desired output: $F(X_1) + F(X_2)$

compute loss

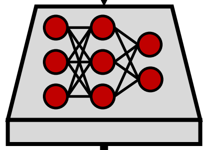
predictions

0.03	0.02	0.95	0.3	0.3	0.4

Training a parity model: higher parameter k

$P = X_1 + X_2 + X_3 + X_4$

Can use higher code parameter k

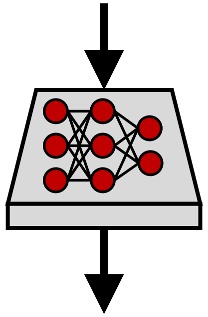


$F_P(P)_1$

Desired output: $F(X_1) + F(X_2) + F(X_3) + F(X_4)$

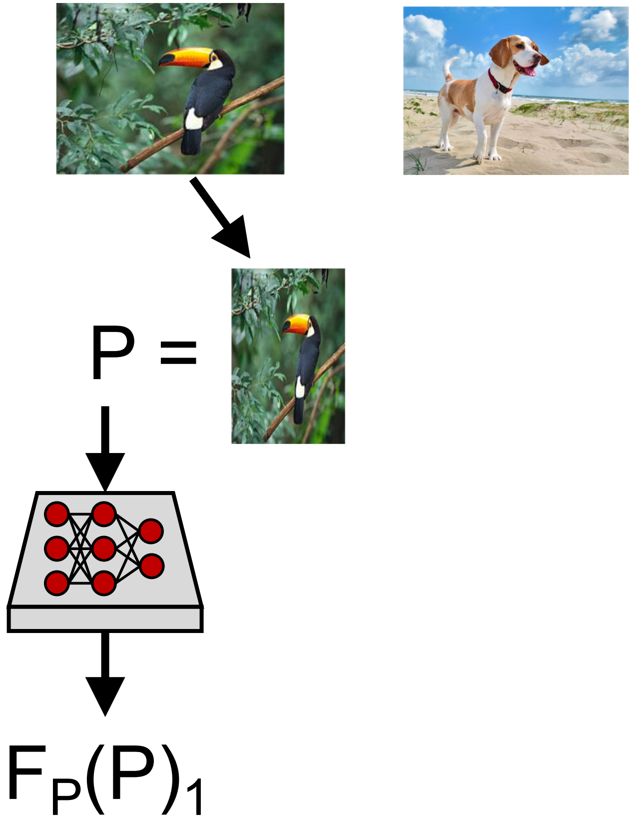
Training a parity model: different encoders

P =

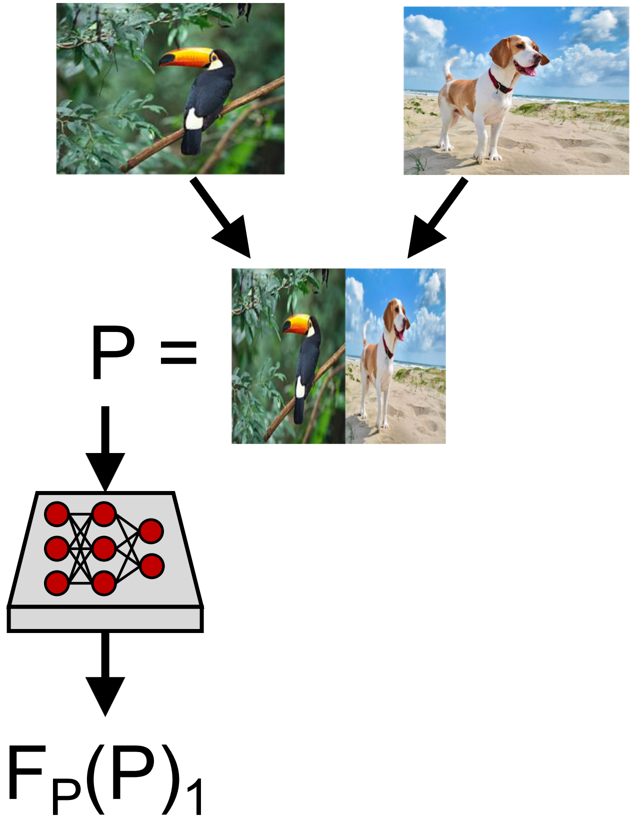


$F_P(P)_1$

Training a parity model: different encoders

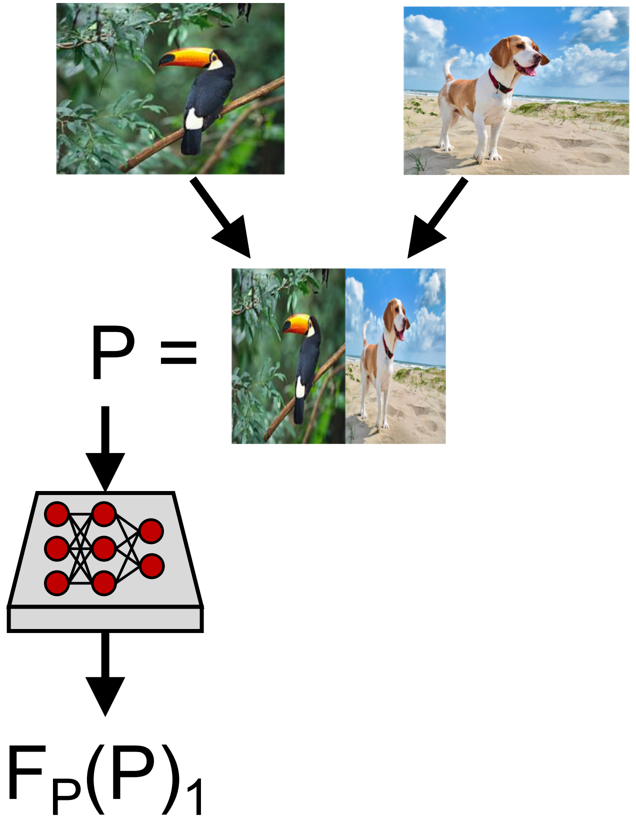


Training a parity model: different encoders



Training a parity model: different encoders

Can specialize encoders and decoders to inference task at hand



Learning results in approximate reconstructions

Learning results in approximate reconstructions

Appropriate for machine learning inference

Learning results in approximate reconstructions

Appropriate for machine learning inference

1. Predictions resulting from inference are approximations

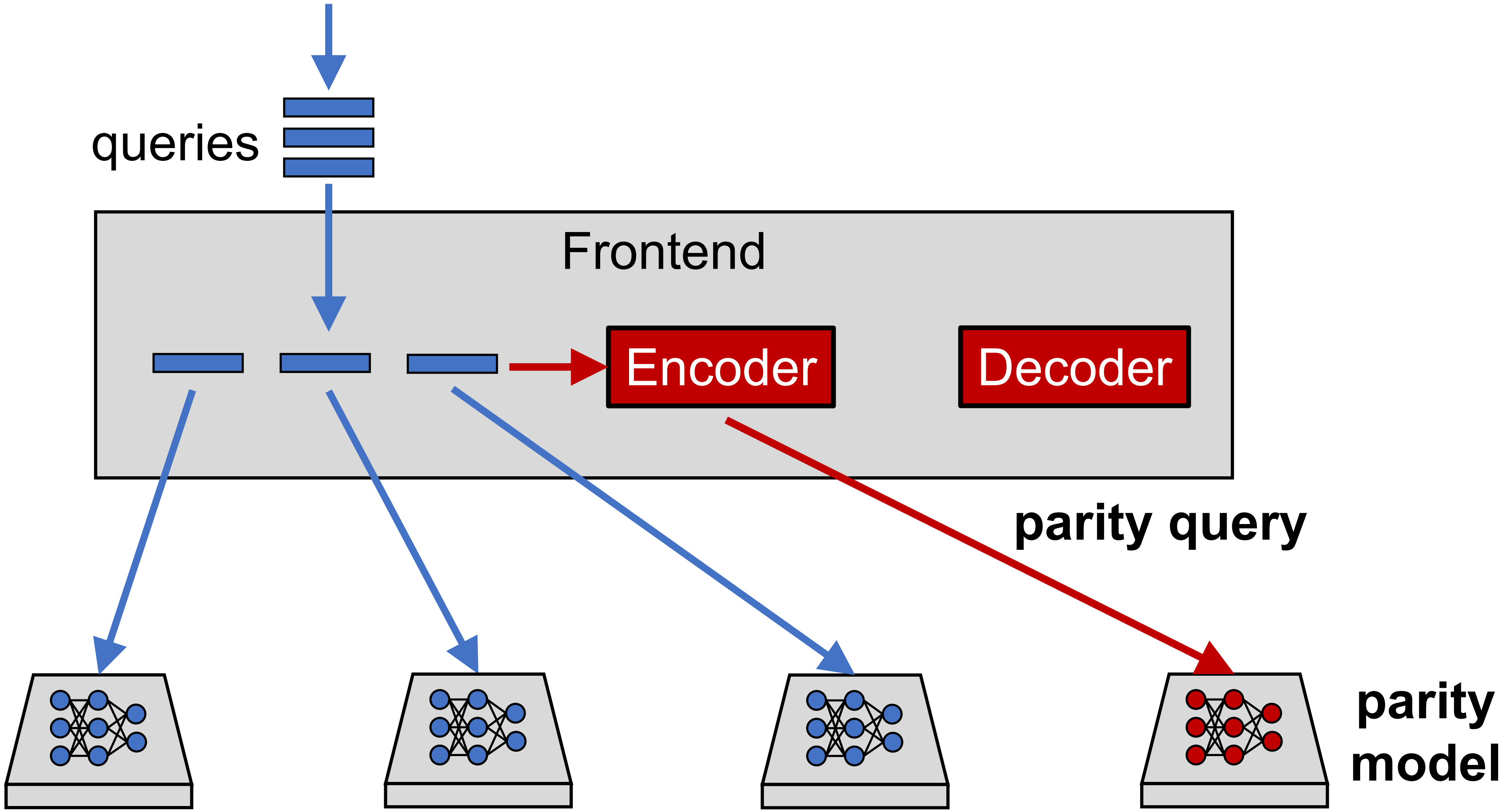
Learning results in approximate reconstructions

Appropriate for machine learning inference

1. Predictions resulting from inference are approximations

2. Inaccuracy only at play when predictions otherwise slow/failed

Parity models in action in Clipper

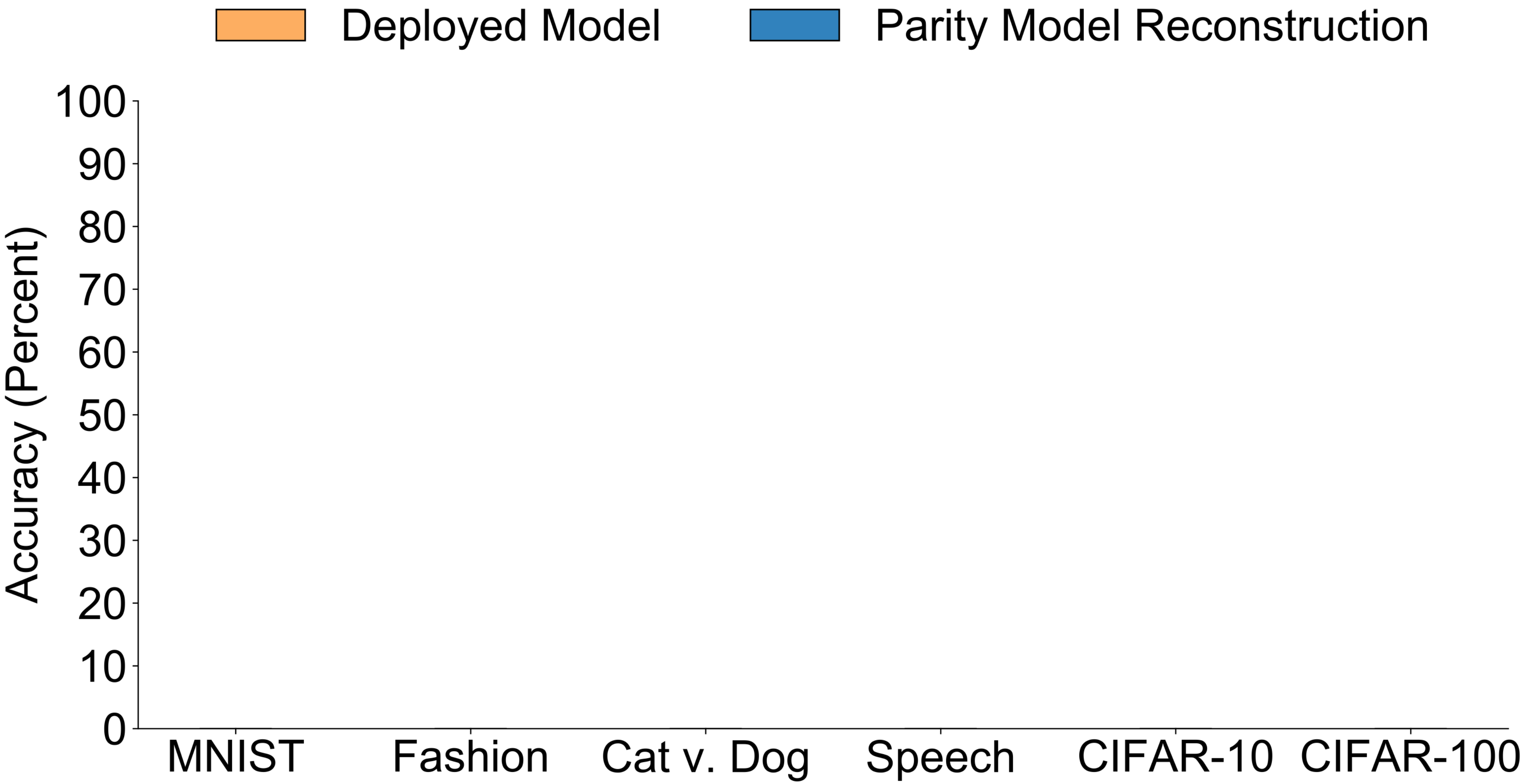


Evaluation

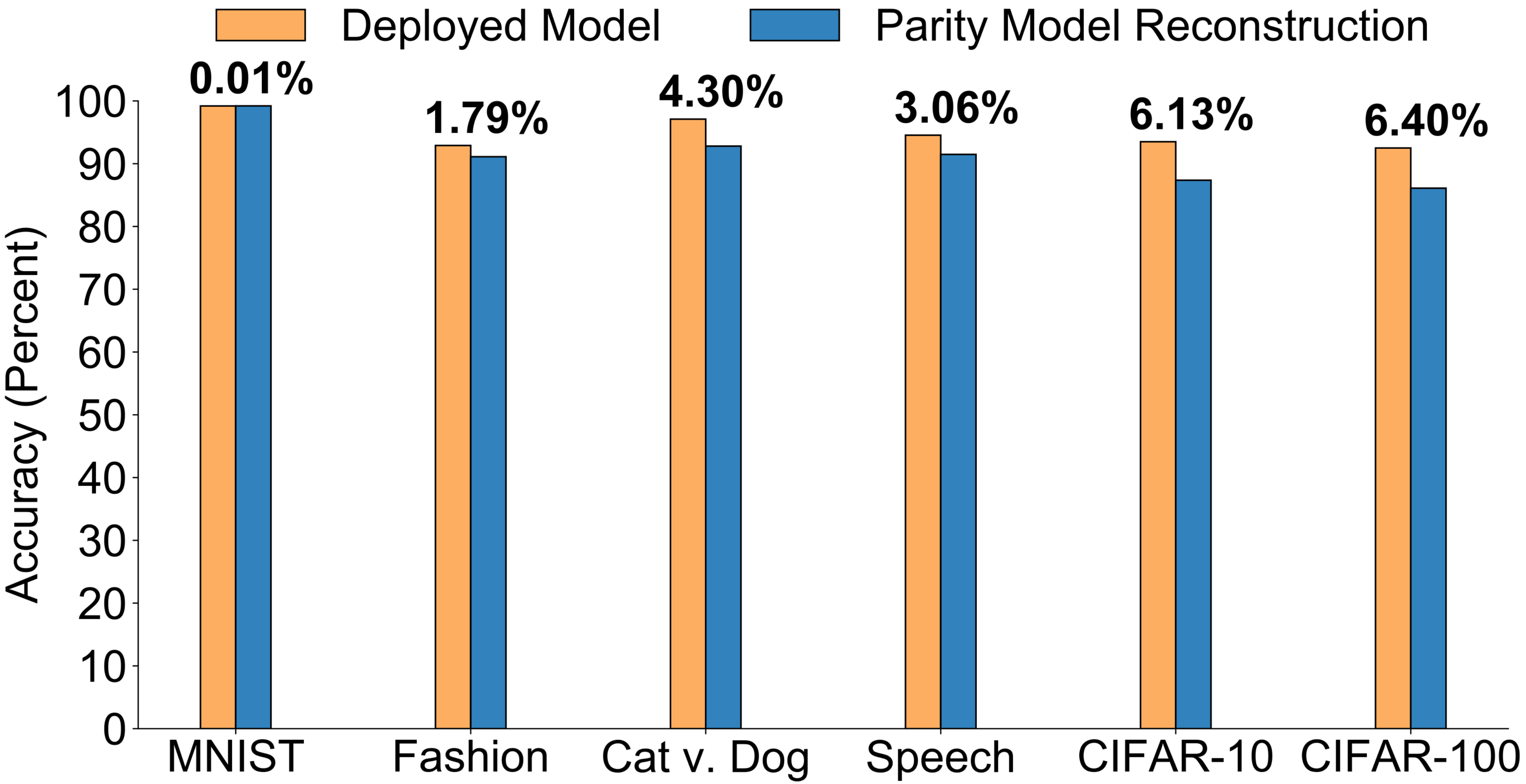
1. How **accurate** are reconstructions using parity models?

2. By how much can parity models help reduce **tail latency**?

Accuracy of parity models

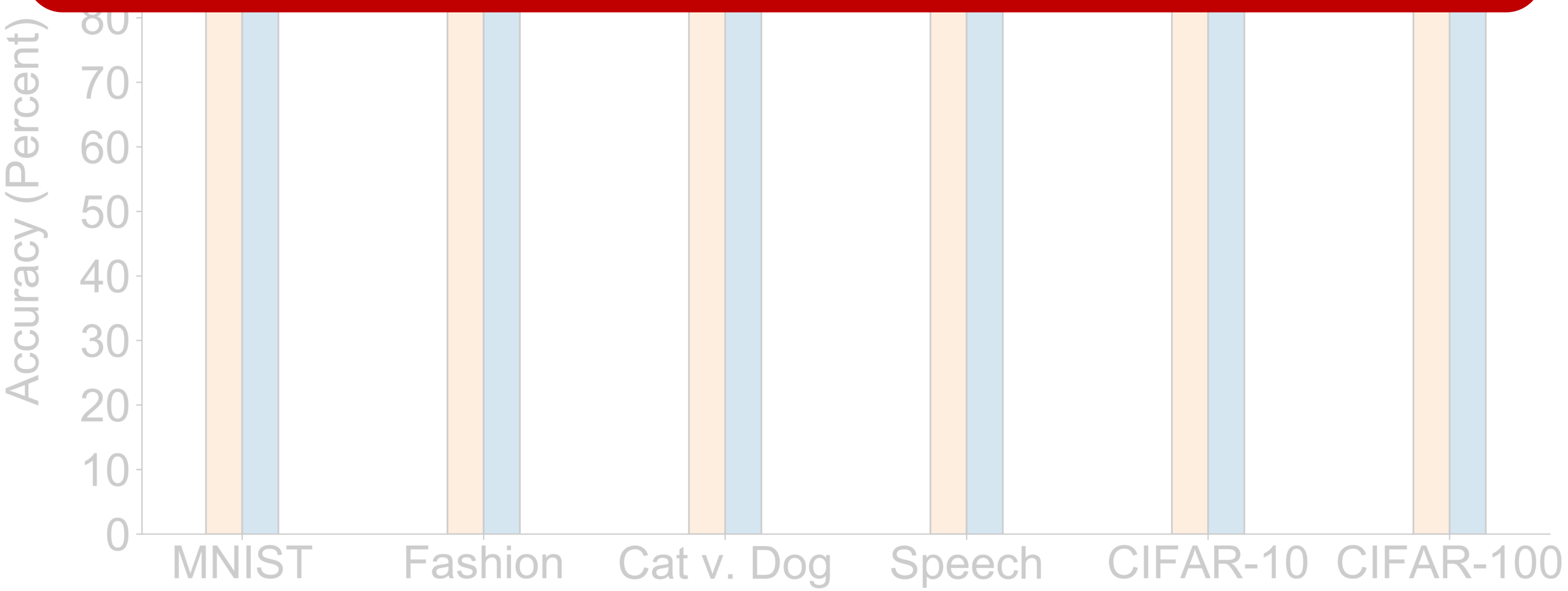


Accuracy of parity models

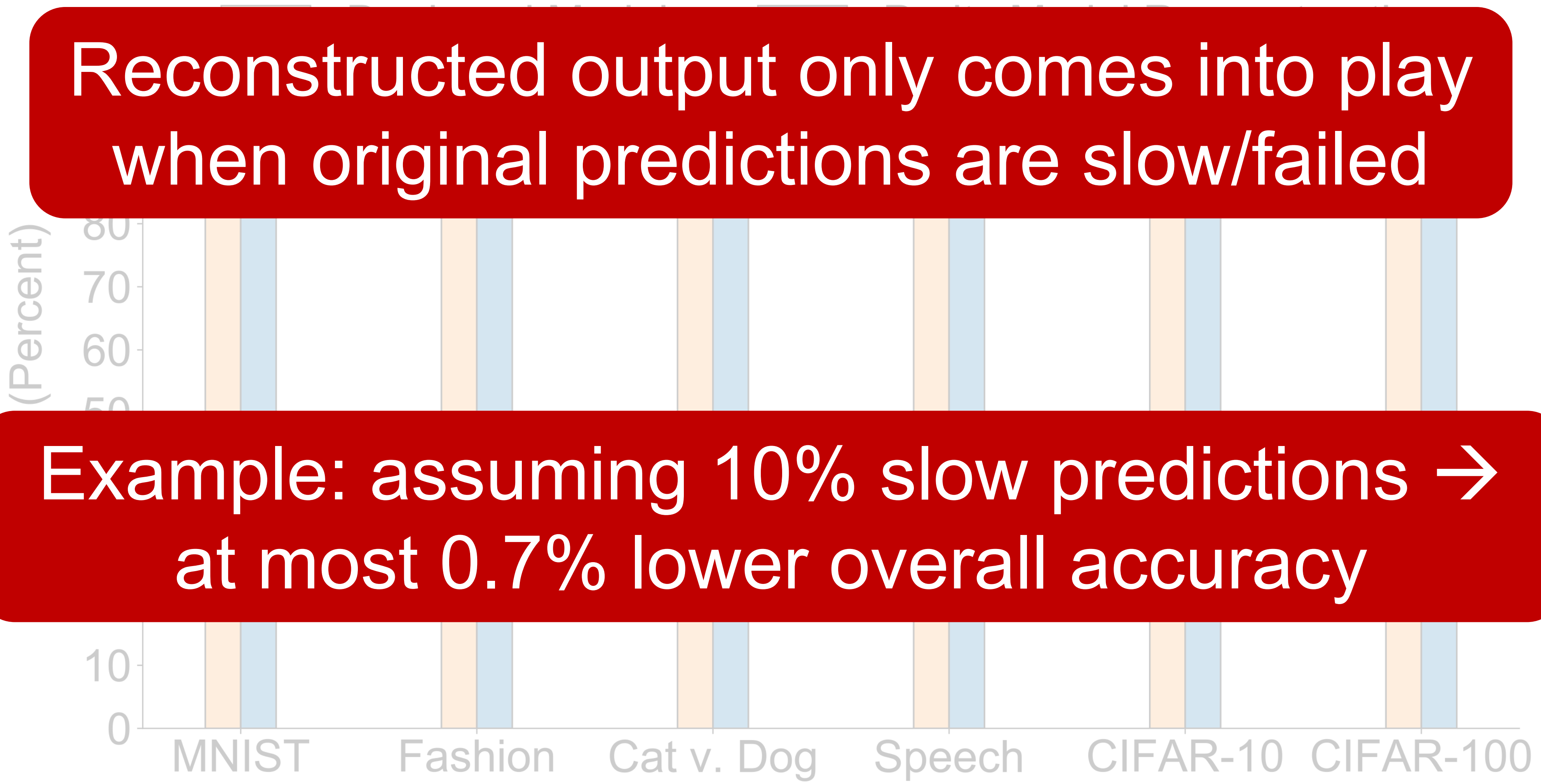


Accuracy of parity models

Reconstructed output only comes into play when original predictions are slow/failed



Accuracy of parity models

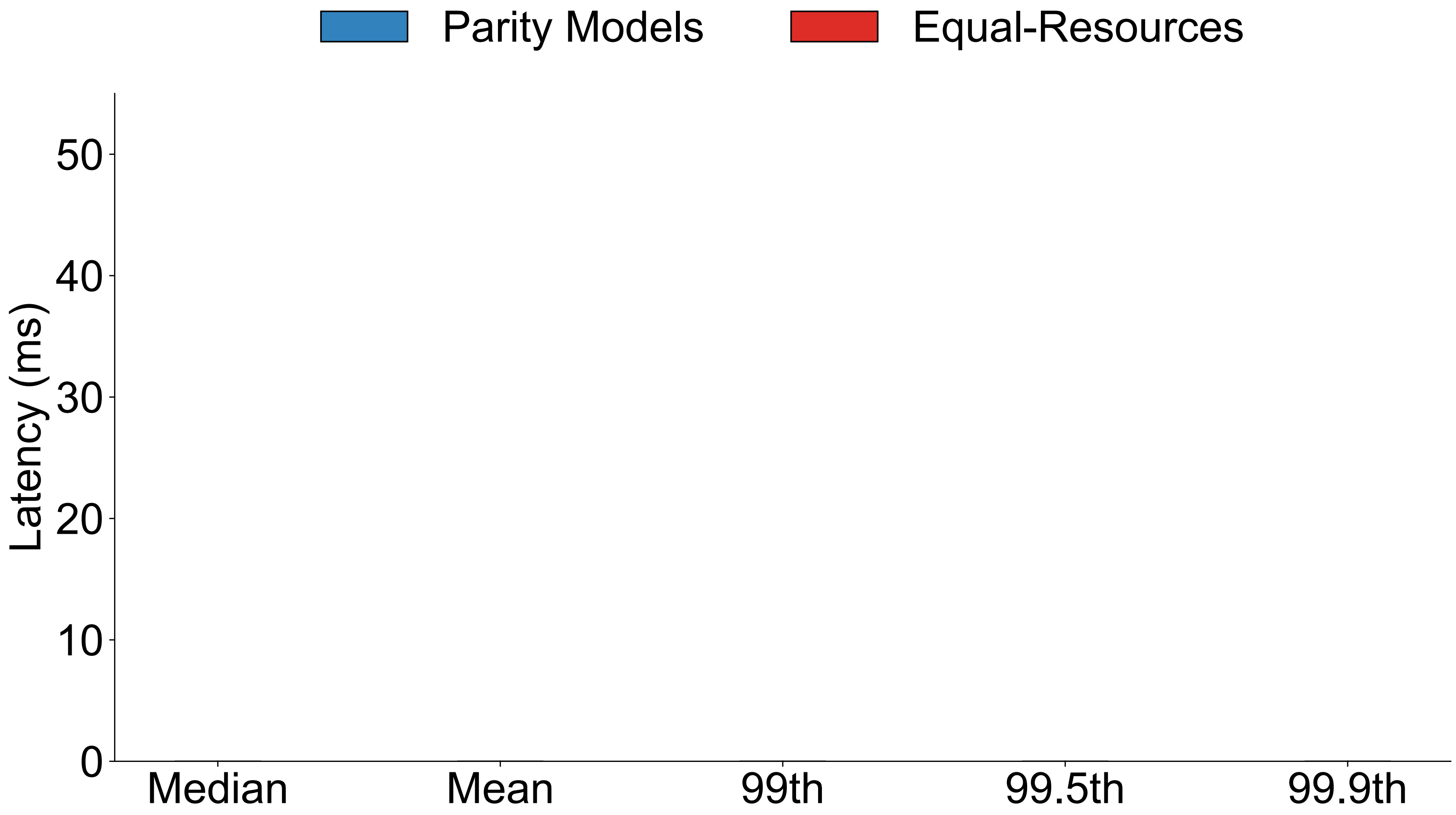


Reconstructed output only comes into play when original predictions are slow/failed

Example: assuming 10% slow predictions → at most 0.7% lower overall accuracy

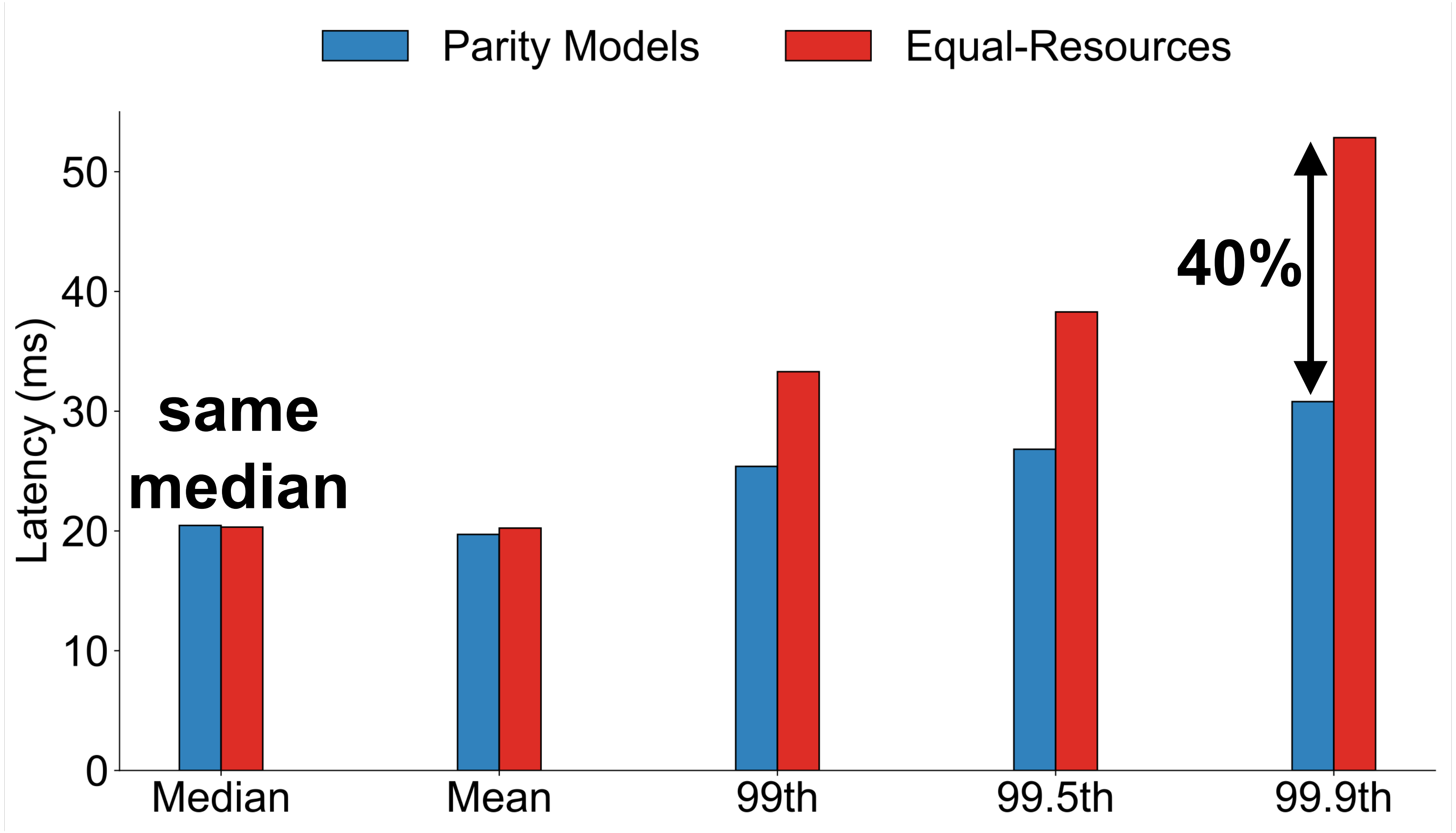
Tail latency reduction

In presence of resource contention



Tail latency reduction

In presence of resource contention



Extensive evaluation in paper

- Evaluate accuracy with different:
  - Different encoders
  - Inference tasks (image classification, object localization, speech)
  - Neural network architectures (ResNets, VGG, LeNet, MLP)
  - Code parameters (k = 3, k = 4)

- Evaluate tail latency with different:
  - Inference hardware (GPUs, CPUs)
  - Query arrival rates
  - Batch sizes
  - Levels of load imbalance
  - Amounts of redundancy
  - Baseline approaches

Parity models summary

Parity models summary

- Overcome challenges of handcrafting erasure codes for coded-computation through learning-based coded-resilience

Parity models summary

- Overcome challenges of handcrafting erasure codes for coded-computation through learning-based coded-resilience

- Parity models: transform parities to enable decoding
  - Applicable to many inference tasks, neural networks
  - Reduce tail latency in presence of resource-contention

Parity models summary

- Overcome challenges of handcrafting erasure codes for coded-computation through learning-based coded-resilience

- Parity models: transform parities to enable decoding
  - Applicable to many inference tasks, neural networks
  - Reduce tail latency in presence of resource-contention

- Bring benefits of erasure codes to ML inference

Parity models summary

- Overcome challenges of handcrafting erasure codes for coded-computation through learning-based coded-resilience

- Parity models: transform parities to enable decoding
  - Applicable to many inference tasks, neural networks
  - Reduce tail latency in presence of resource-contention

- Bring benefits of erasure codes to ML inference

Project: pdl.cmu.edu/MLCodedComputation/

Code: github.com/Thesys-lab/parity-models