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

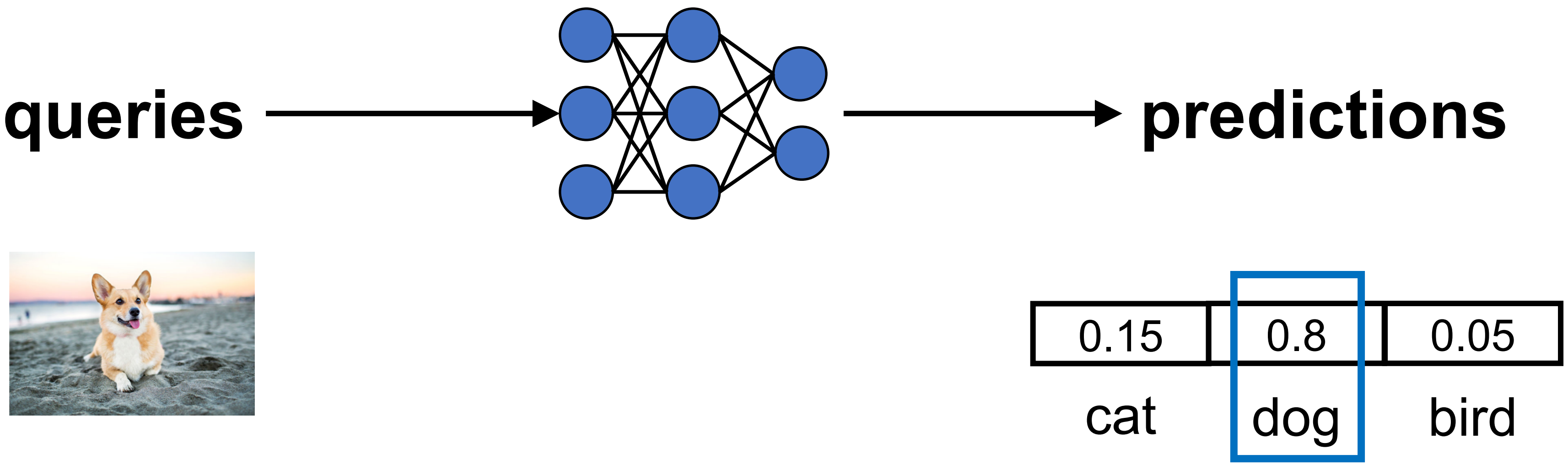
Shivaram Venkataraman

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Machine learning lifecycle

Training		Inference
Get a model to reach desired accuracy	→	Deploy model in target domain
"Batch" jobs		Online
Hours to weeks		Milliseconds

Machine learning inference



Prediction serving systems

Inference in datacenter/cluster settings

Open Source

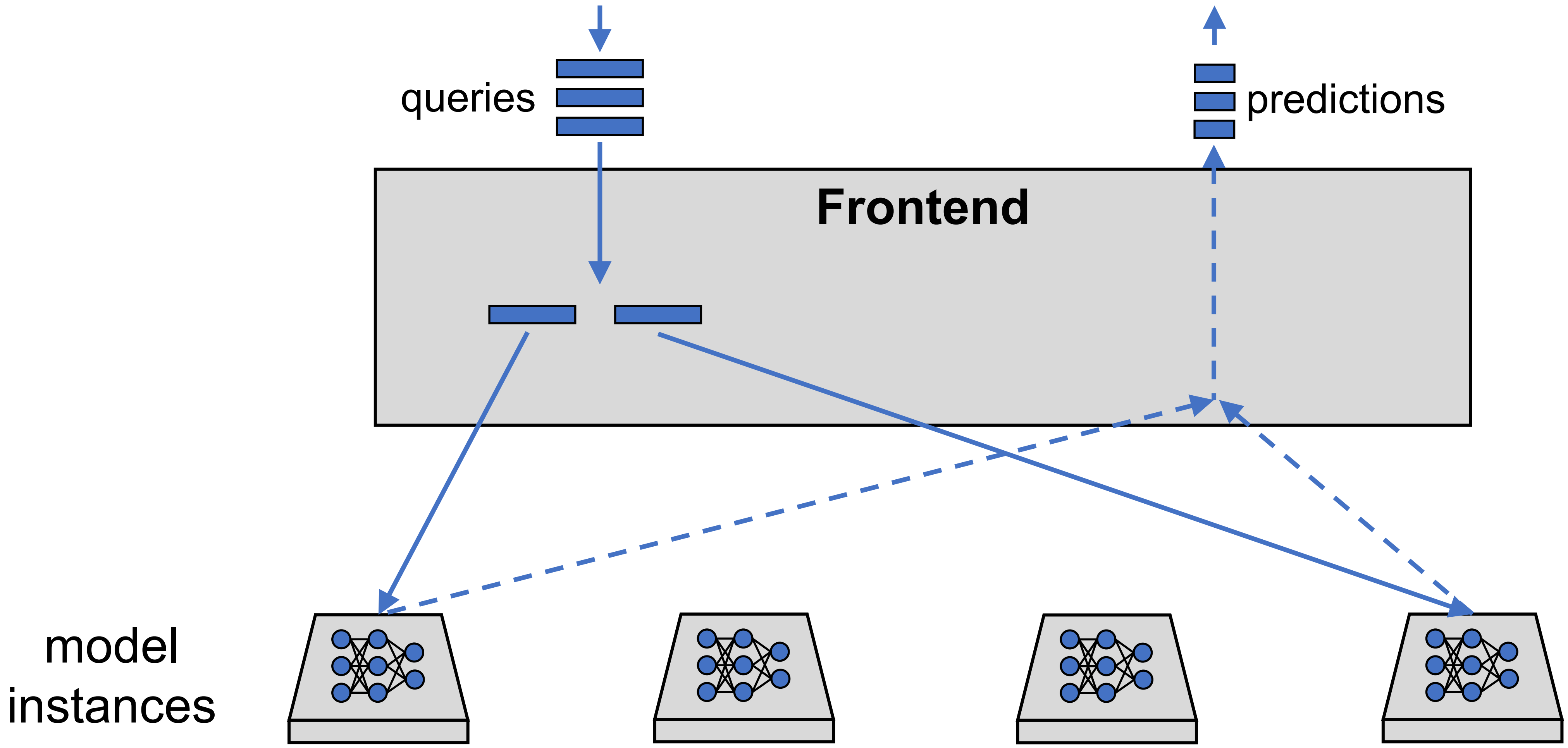
Clipper

mxnet

Cloud Services

Amazon SageMaker

Prediction serving system architectures



Machine learning inference

question-answering

translation

ranking

Must operate with low, predictable latency

Unavailability in serving systems

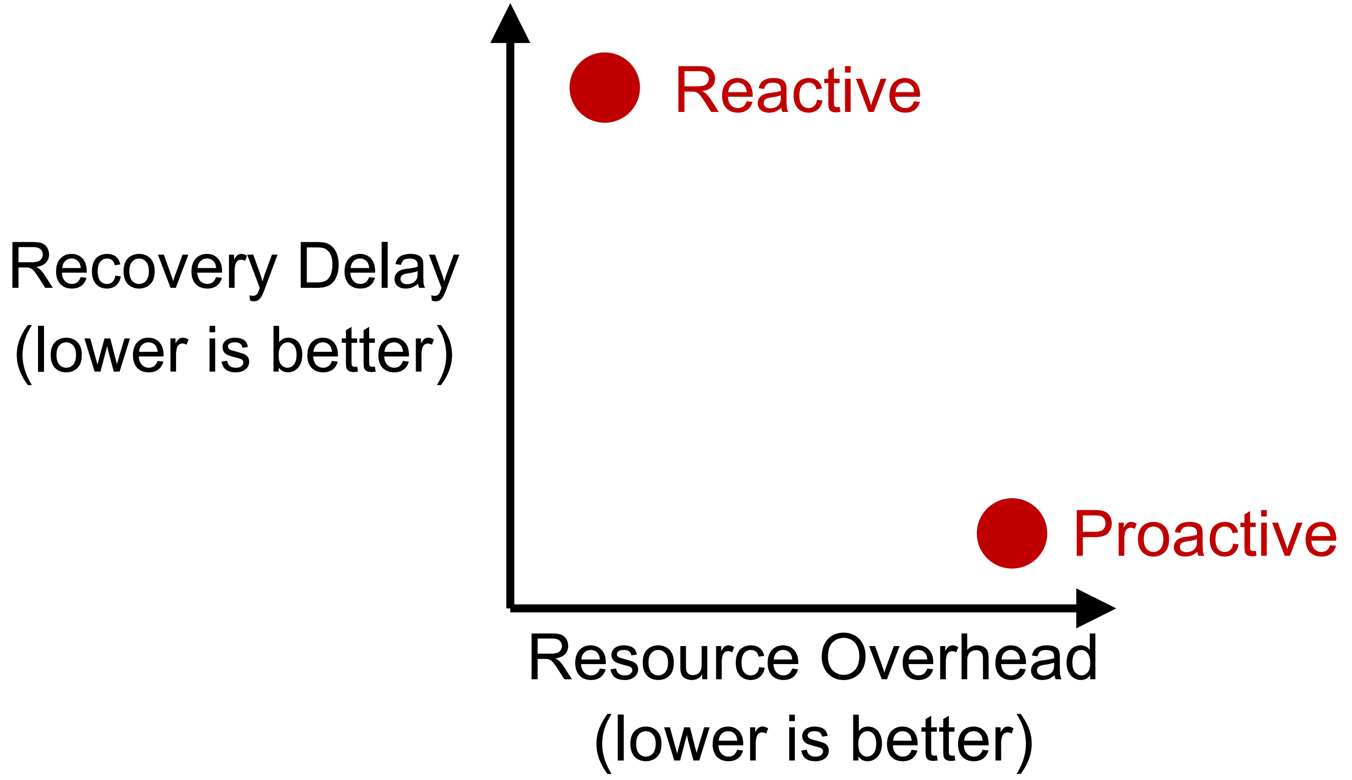
- Slowdowns and failures (unavailability)
  - Resource contention
  - Hardware failures
  - Runtime slowdowns
  - ML-specific events

- Result in **inflated tail latency**
  - Cause prediction serving systems to miss SLOs

Must alleviate slowdowns and failures

Redundancy-based resilience

- **Proactive:** send each query to 2+ servers
- **Reactive:** wait for a timeout before duplicating query



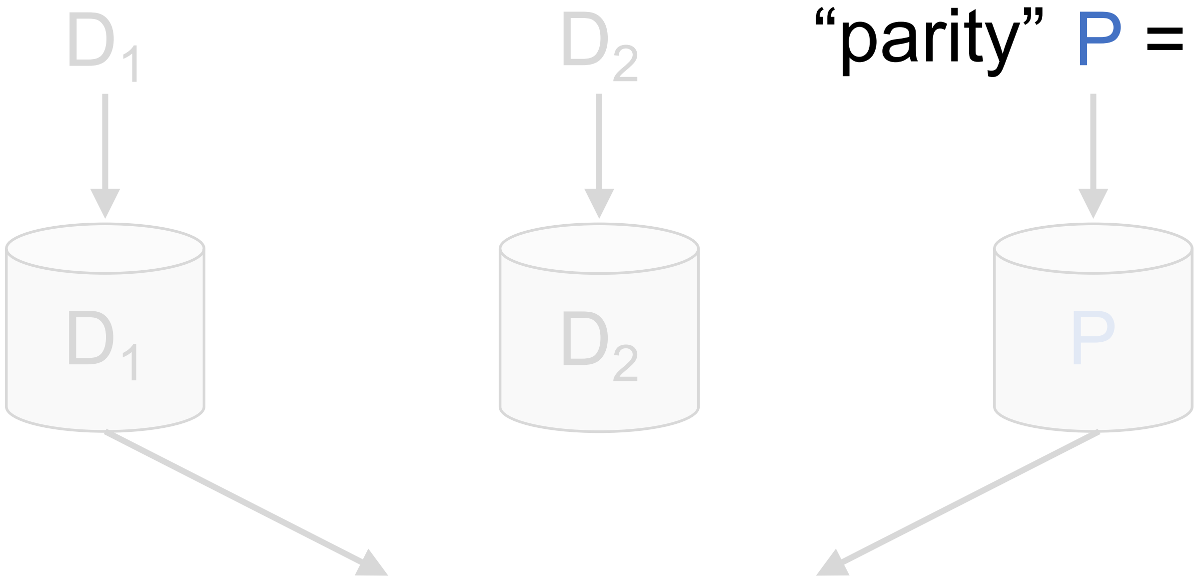
Erasure codes: proactive, resource-efficient

Relation to (n, k) notation

$$n = k + r$$

k data units → encoding → r "parity" units

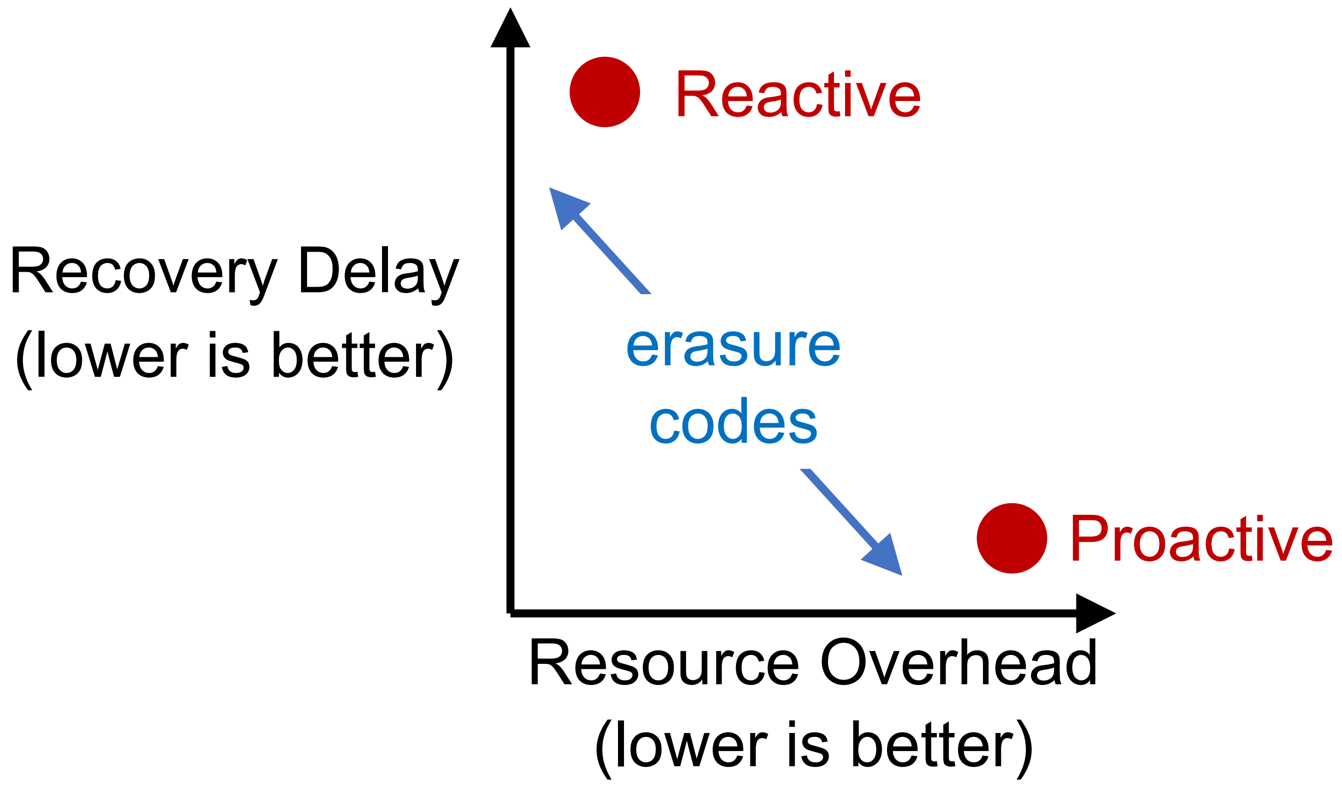
"parity" $P = D_1 + D_2$



$$D_2 = P - D_1$$

any k out of (k+r) units → decoding → original k data units

Erasure codes: proactive, resource-efficient

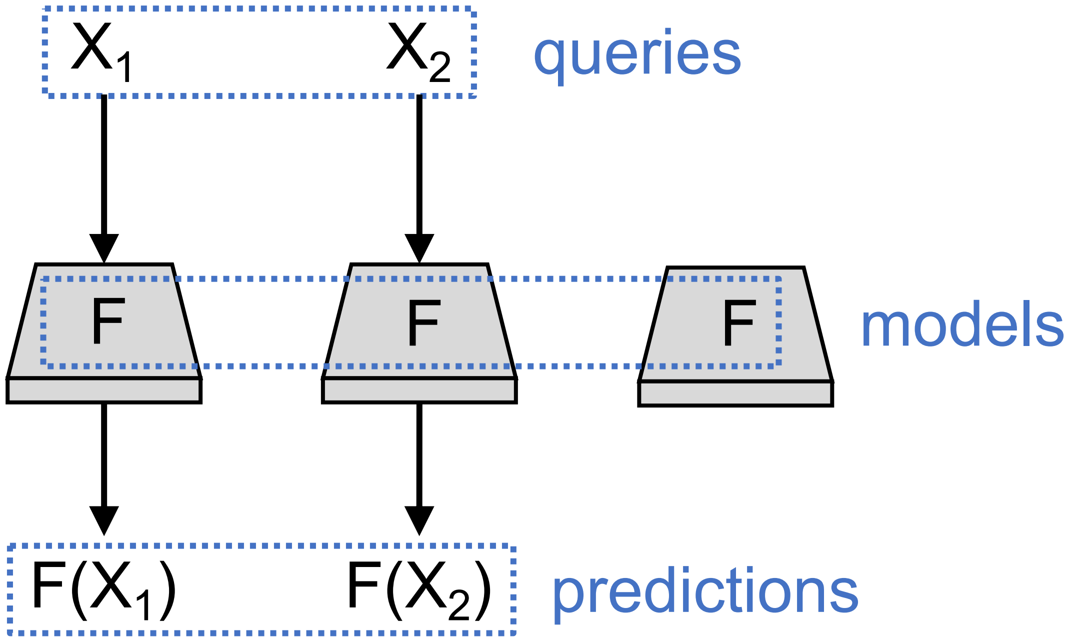


- Storage
- Communication
- Prediction Serving Systems

Coded-computation

Our goal: Using erasure codes to reduce tail latency in prediction serving

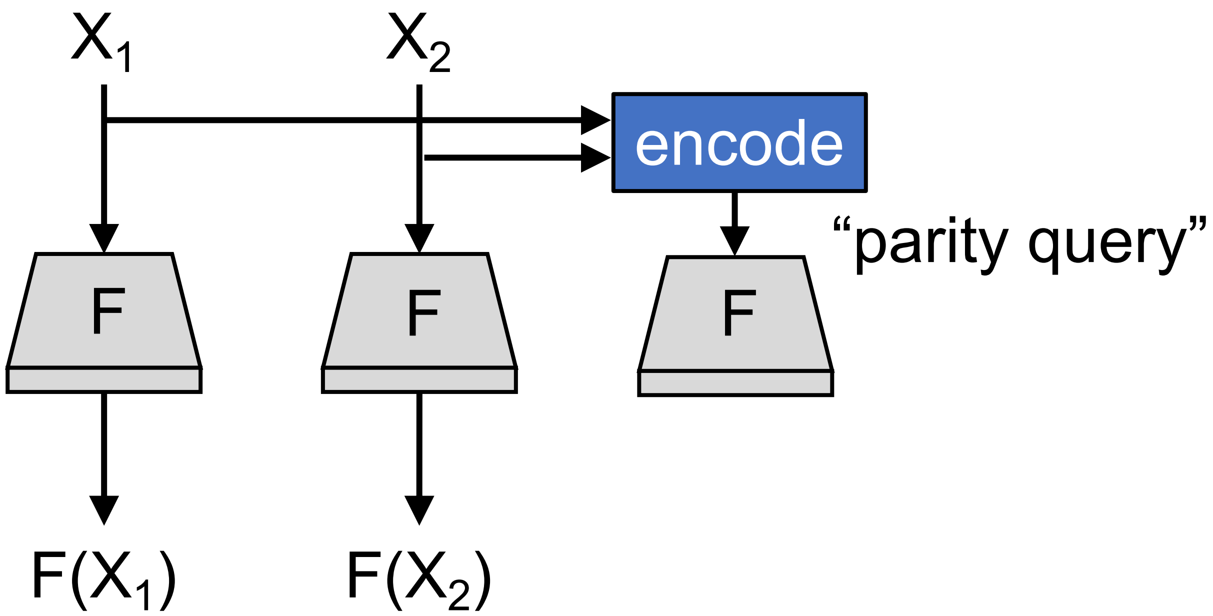
Goal: preserve results of computation over queries



Coded-computation

Our goal: Using erasure codes to reduce tail latency in prediction serving

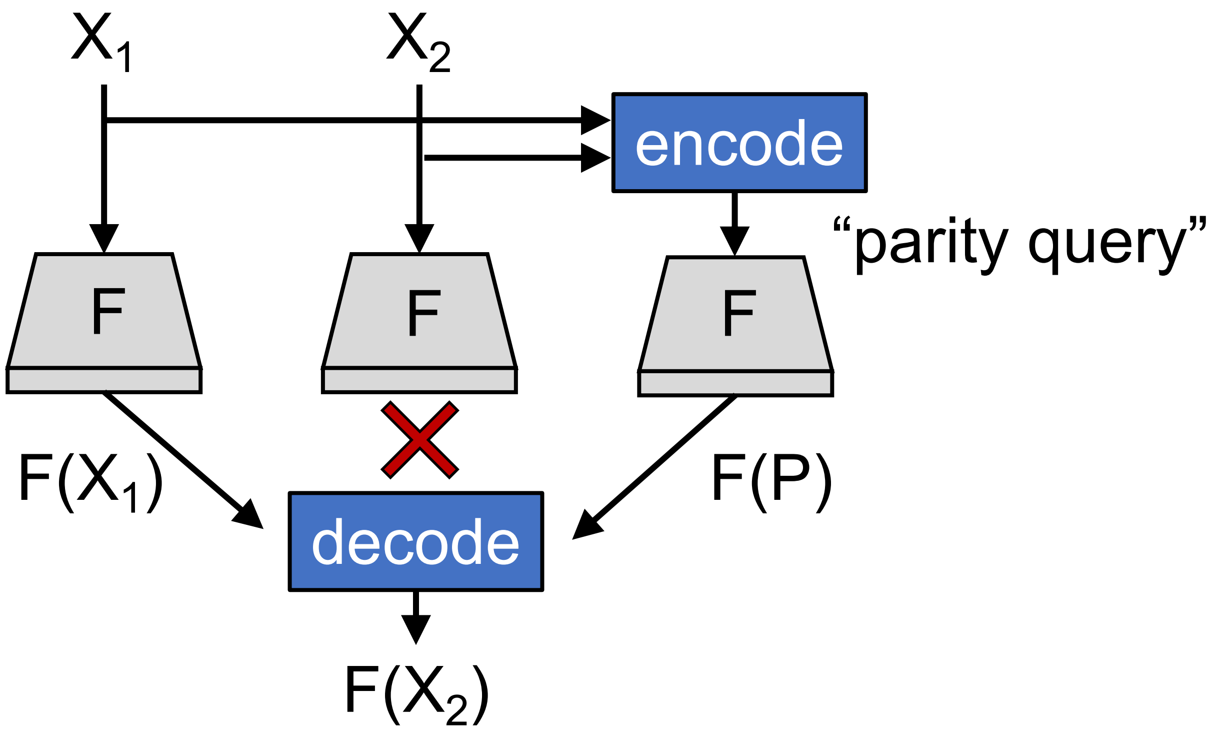
Encode queries



Coded-computation

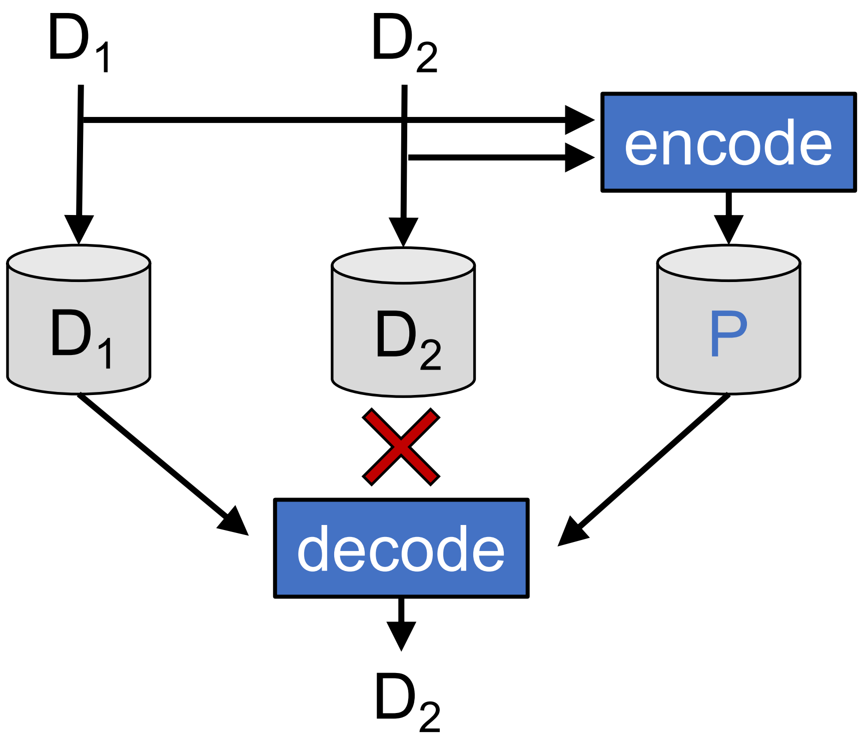
Our goal: Using erasure codes to reduce tail latency in prediction serving

Decode results of
inference over queries

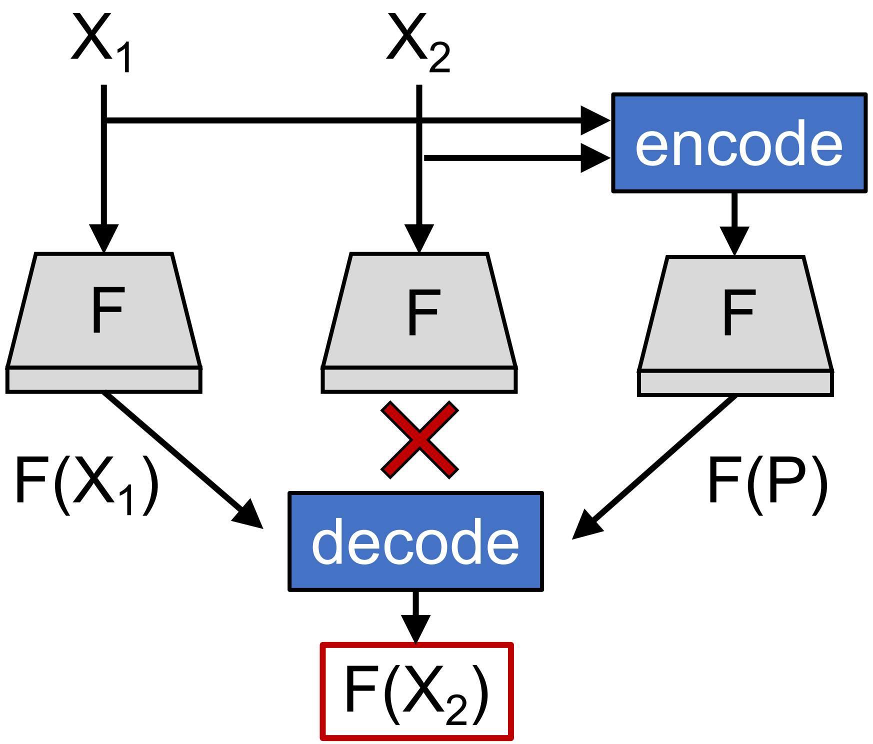


Traditional coding vs. coded-computation

Codes for storage



Coded-computation

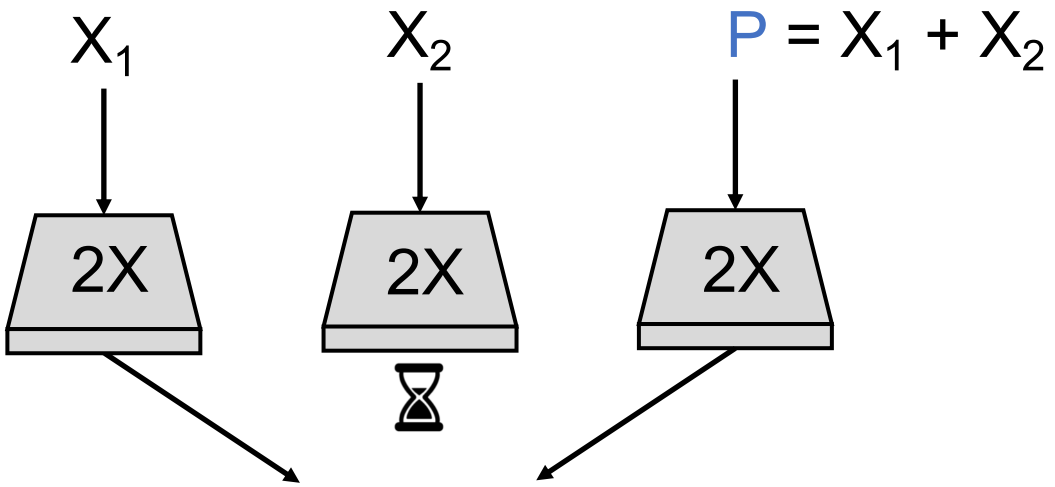


Need to recover **computation over inputs**

Challenge: Non-linear computation

Linear computation

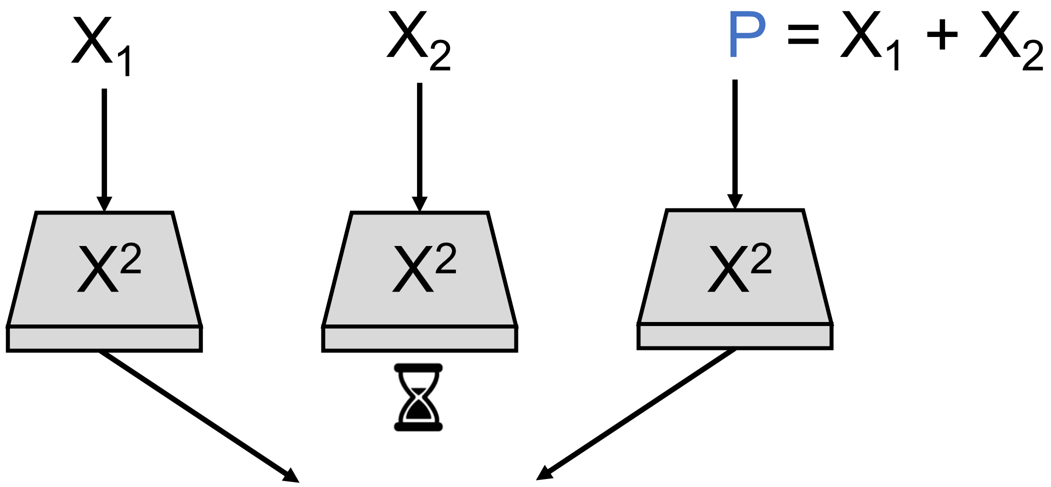
Example: $F(X) = 2X$



$$
\begin{aligned}
F(X_2) &= F(P) - F(X_1) \\
&= 2(X_1 + X_2) - X_1 \\
&= 2X_2
\end{aligned}
$$

Non-linear computation

Example: $F(X) = X^2$



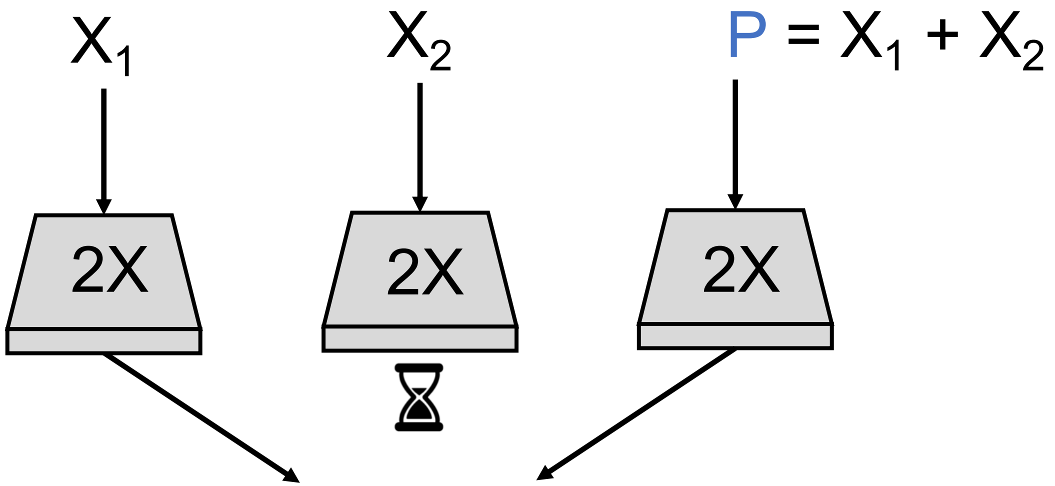
$$
\begin{aligned}
F(X_2) &= F(P) - F(X_1) \\
&= 2(X_1 + X_2)^2 - X_1^2 \\
&= X_2^2 + 2X_1X_2
\end{aligned}
$$

Actual is X_2^2

Challenge: Non-linear computation

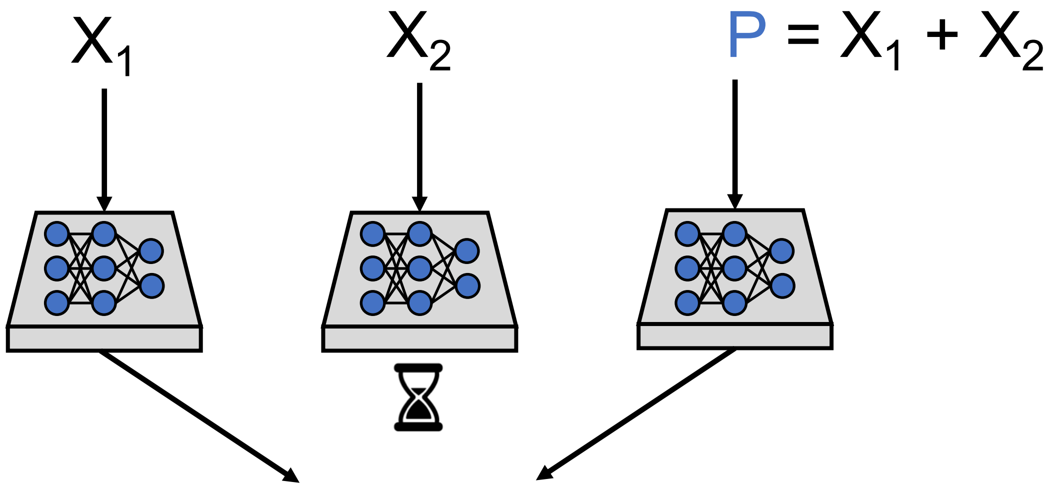
Linear computation

Example: $F(X) = 2X$



$$F(X_2) = F(P) - F(X_1)$$
$$= 2(X_1 + X_2) - X_1$$
$$= 2X_2$$

Non-linear computation



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

Current approaches to coded-computation

- Lots of great work on **linear computations**
  - Huang 1984, Lee 2015, Dutta 2016, Dutta 2017, Mallick 2018, more…
- Recent work supports restricted **nonlinear computations**
  - Yu 2018
  - At least 2x resource overhead

Current approaches insufficient for neural networks in prediction serving systems

Our approach:
Learning-based coded-computation

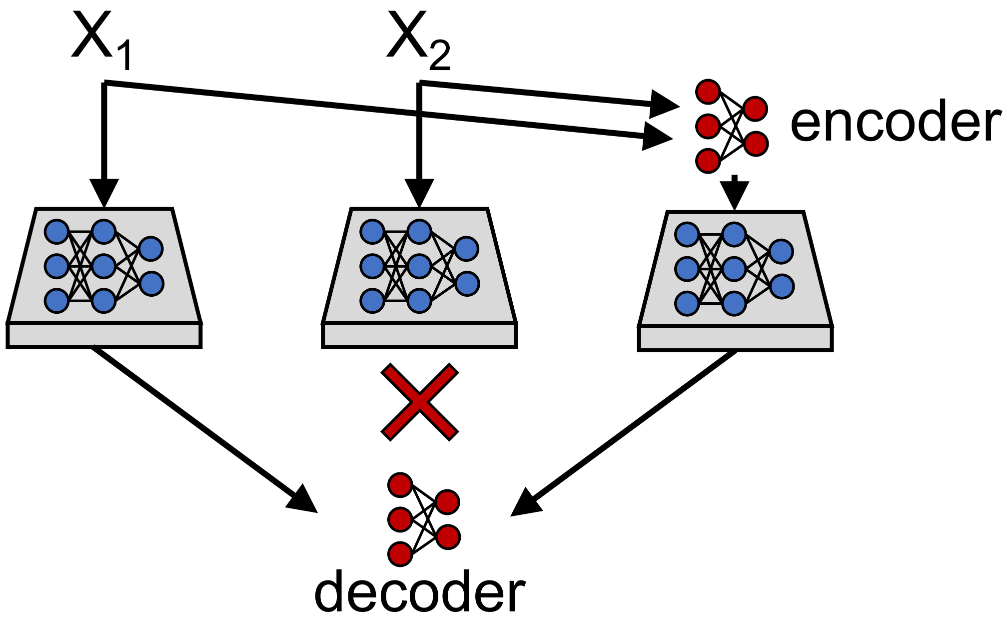
Learning a Code: Machine Learning for Approximate Non-Linear Coded Computation
https://arxiv.org/abs/1806.01259

Parity Models: Erasure-Coded Resilience for Prediction Serving Systems
To appear in ACM SOSP 2019
https://jackkosaian.github.io

Learning an erasure code?

Design encoder and decoder as neural networks

Accurate



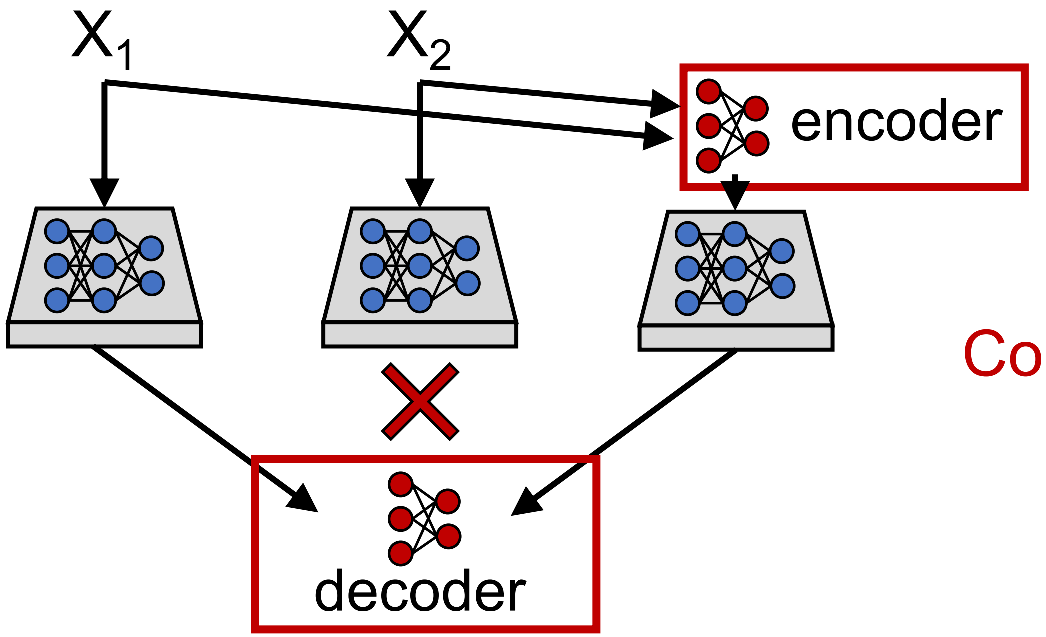
Learning a Code: Machine Learning for Approximate Non-Linear Coded Computation
https://arxiv.org/abs/1806.01259

Learning an erasure code?

Design encoder and decoder as neural networks

Accurate

Expensive
encoder/decoder



Computationally
expensive

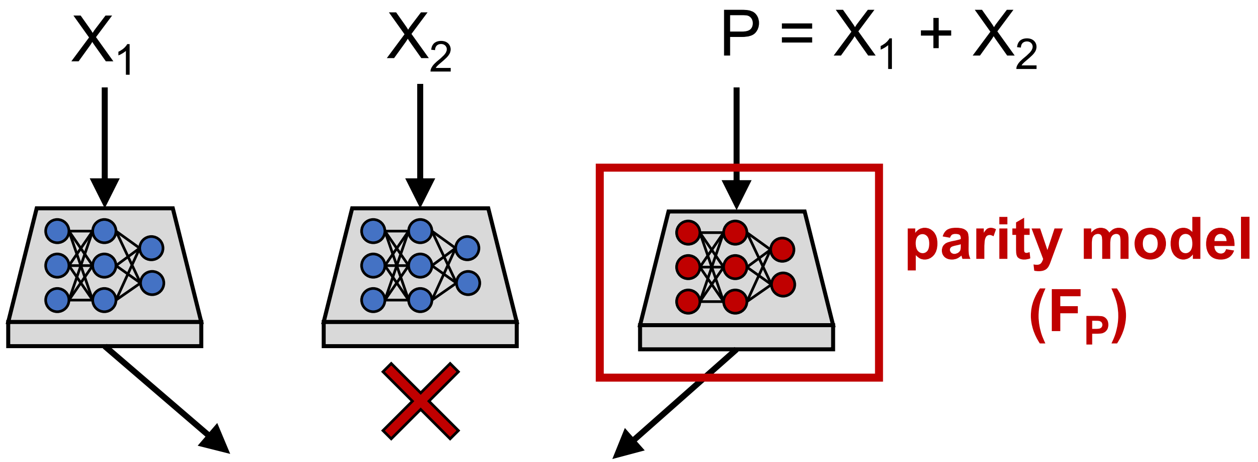
Learning a Code: Machine Learning for Approximate Non-Linear Coded Computation
https://arxiv.org/abs/1806.01259

Learn computation over parities

Use simple, fast encoders and decoders

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

- Accurate
- Efficient encoder/decoder



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

Parity Models: Erasure-Coded Resilience for Prediction Serving Systems

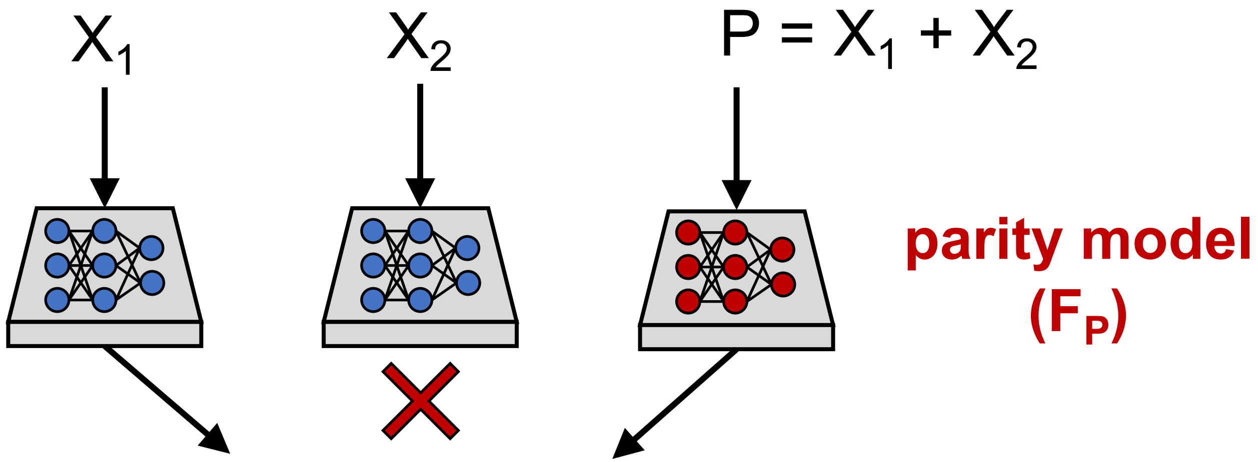
To appear in ACM SOSP 2019

https://jackkosaian.github.io

Designing parity models

Parity model goal

Transform parities such that decoder can reconstruct unavailable predictions

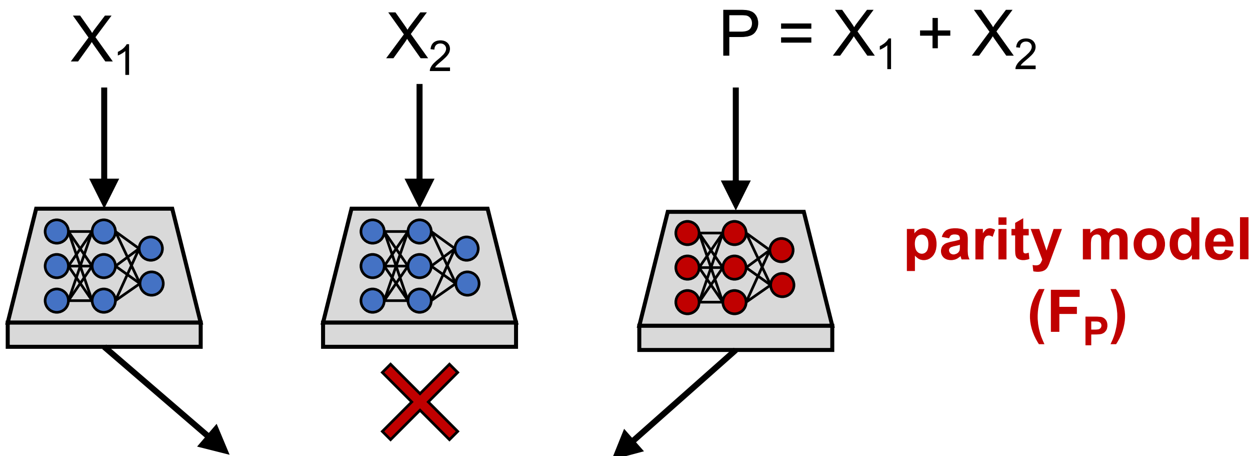


$F(X_2) = \mathbf{F_P}(P) - F(X_1)$

Designing parity models

Parity model goal

Transform parities such that decoder can reconstruct unavailable predictions



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

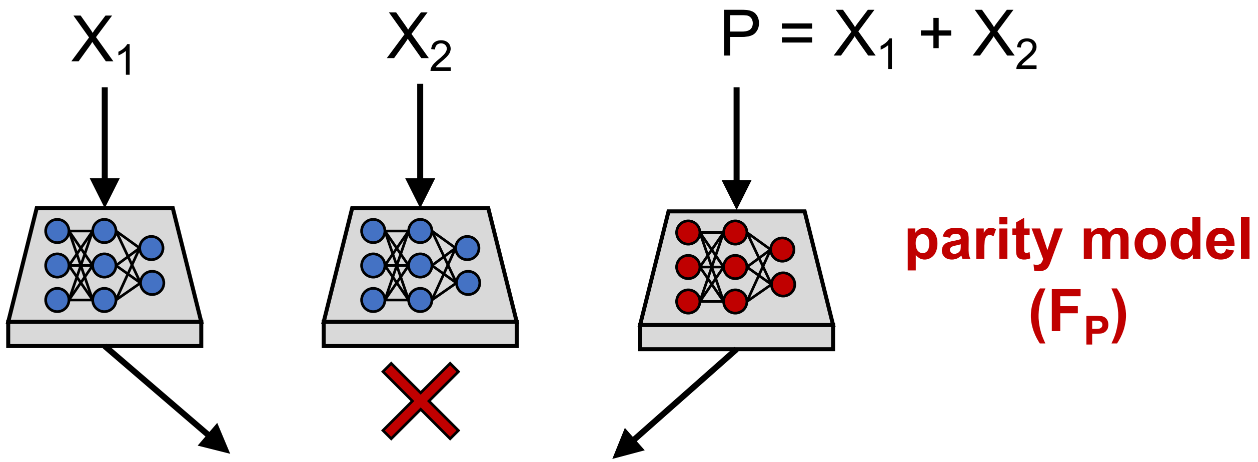
$$\boxed{F_P(P) = F(X_1) + F(X_2)}$$

Learn a parity model

Designing parity models

Parity model goal

Transform parities such that decoder can reconstruct unavailable predictions

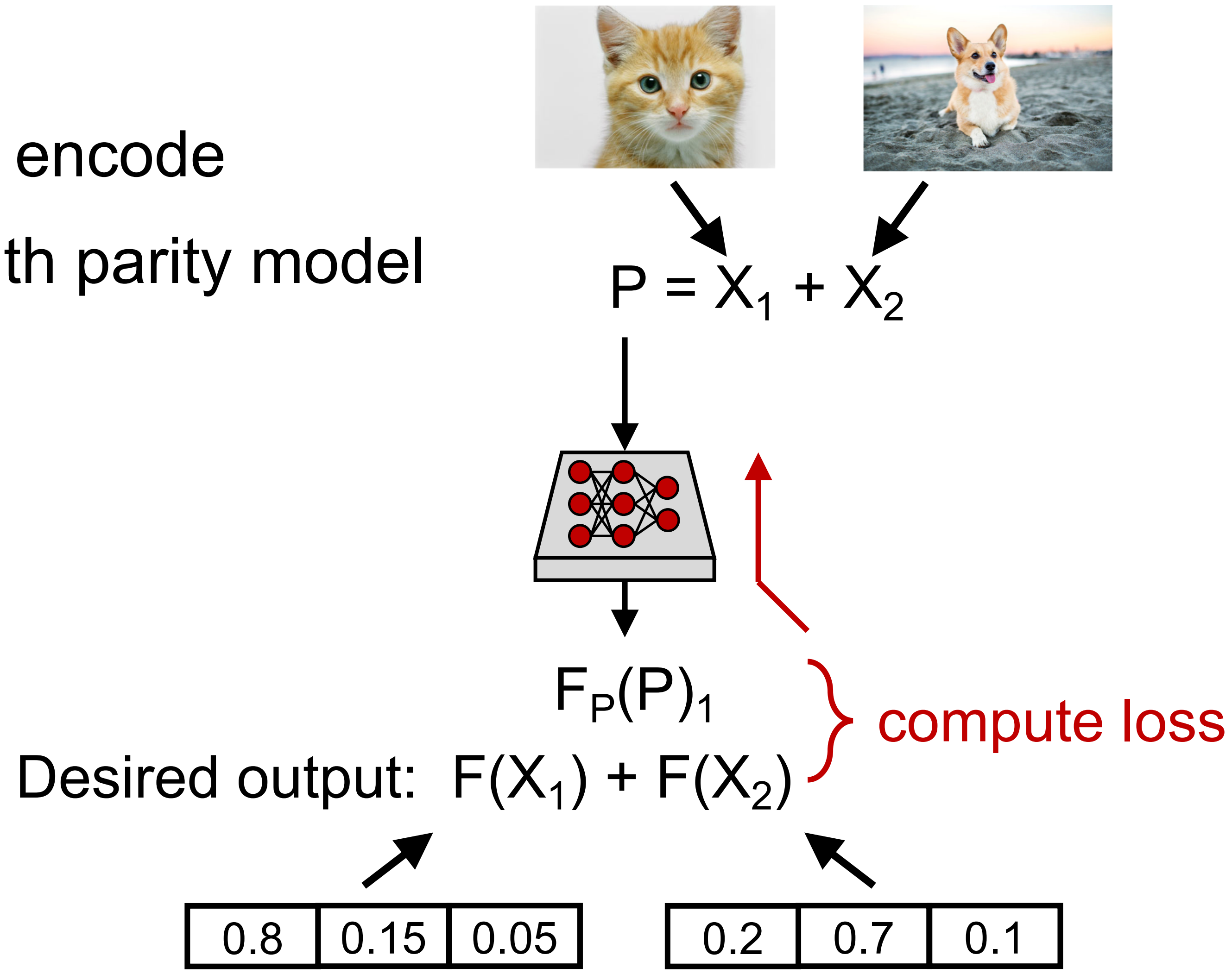


$$F(X_2) = F_P(P) - F(X_1)$$

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

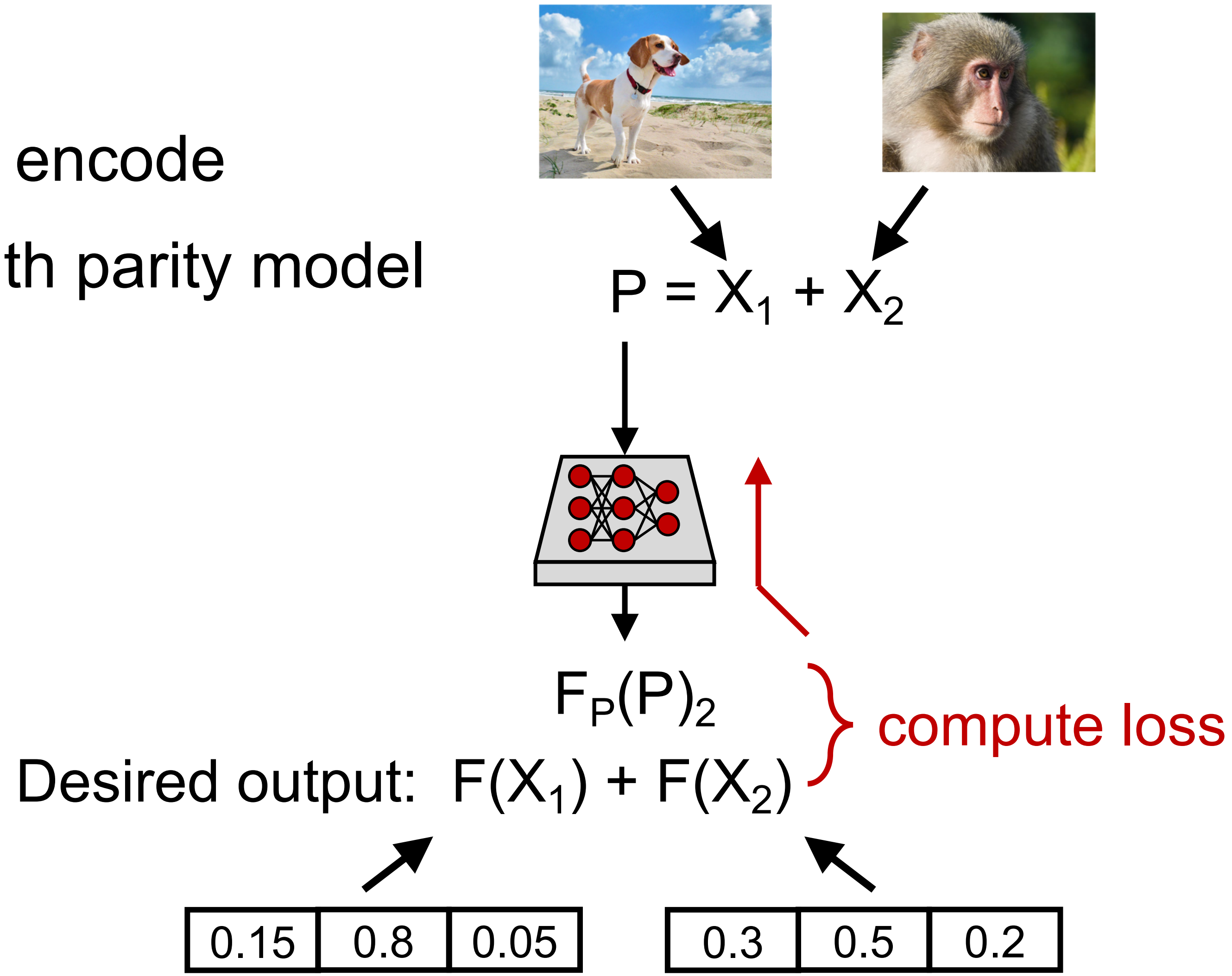
Training a parity model

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



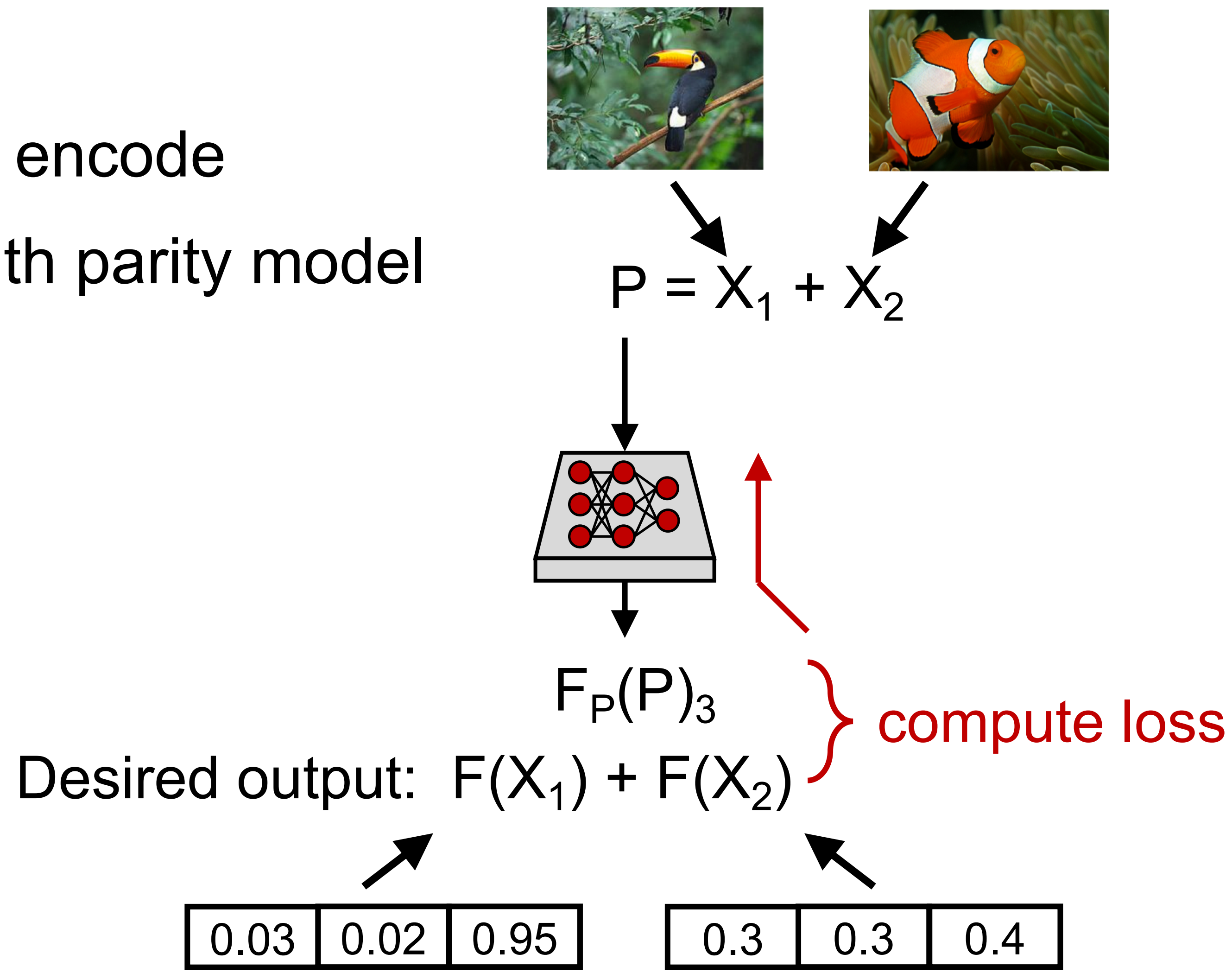
Training a parity model

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



Training a parity model

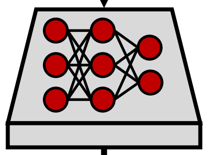
1. Sample k inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropogate loss
5. Repeat



Training a parity model: higher parameter k

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

$P = X_1 + X_2 + X_3 + X_4$

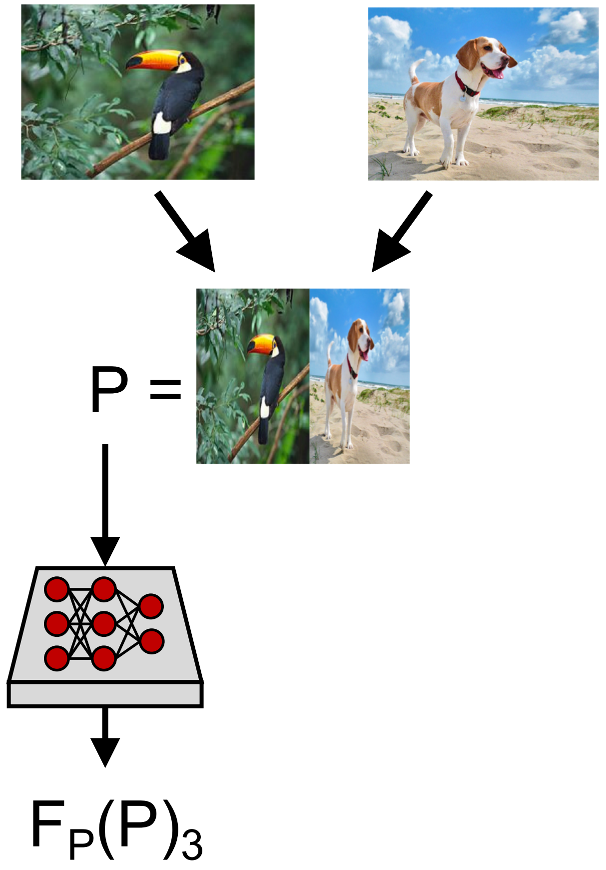


$F_P(P)_3$

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

Training a parity model: different encoder

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



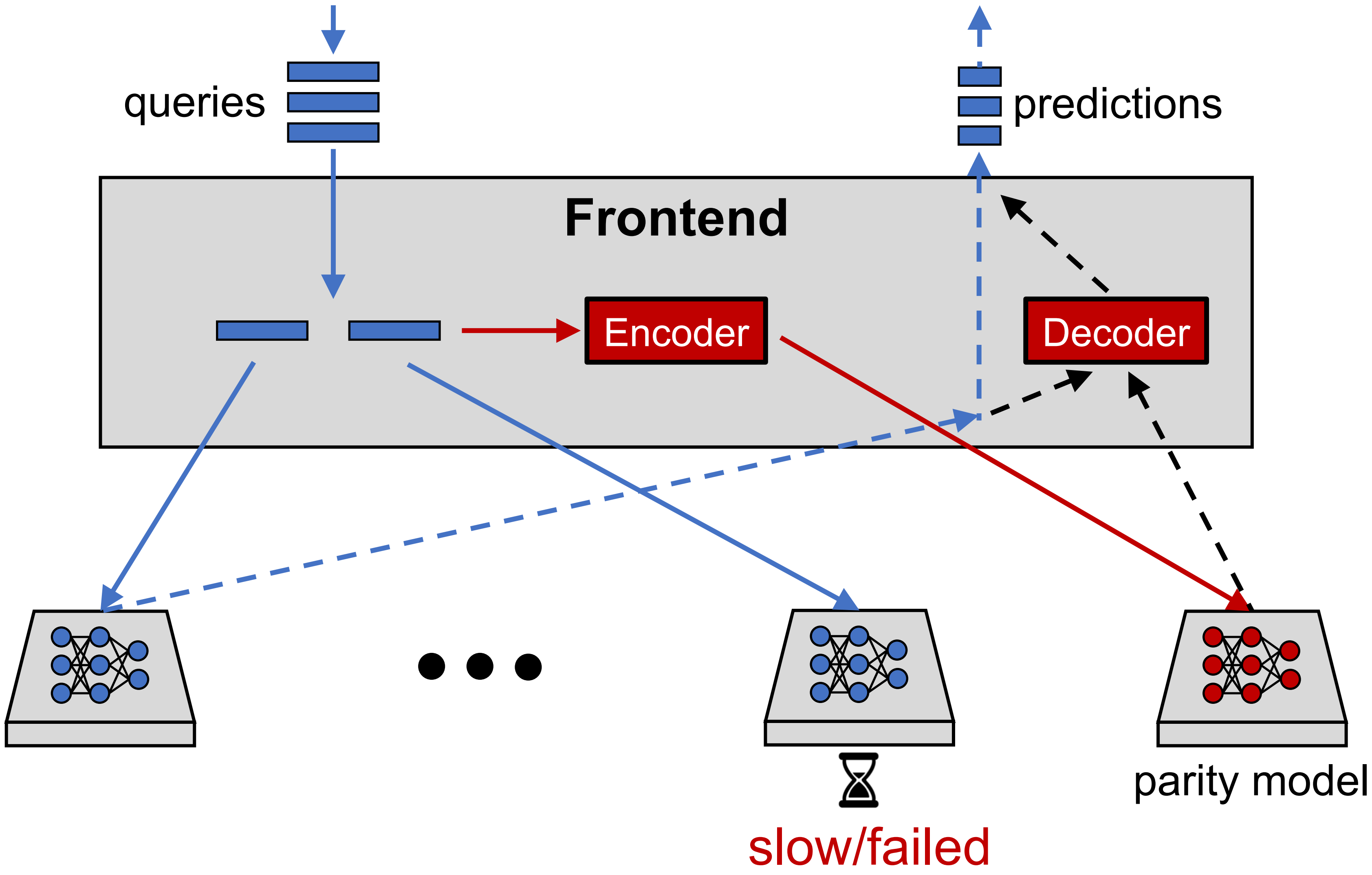
Learning results in approximate reconstructions

Appropriate for machine learning inference

1. Predictions resulting from inference are approximations

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

Implementing parity models in Clipper



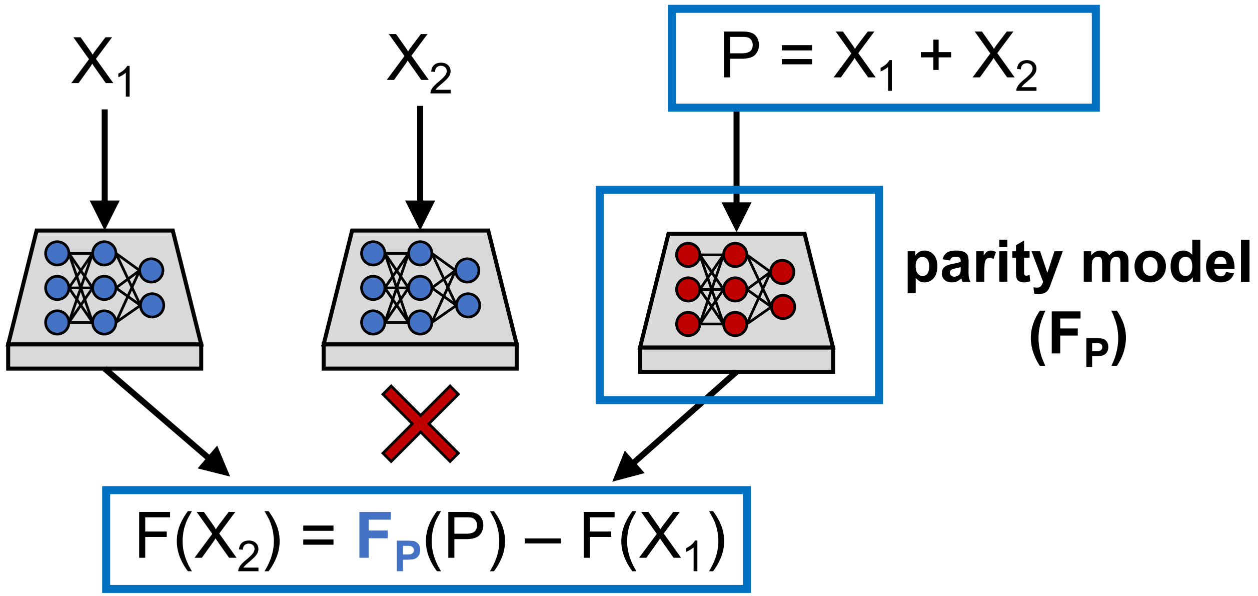
Design space in parity models framework

Encoder/decoder

- Many possibilities
- Generic: addition/subtraction
- Can specialize to task

Parity model architecture

- Again, many possibilities
- Same as original model ⇒ same latency as original

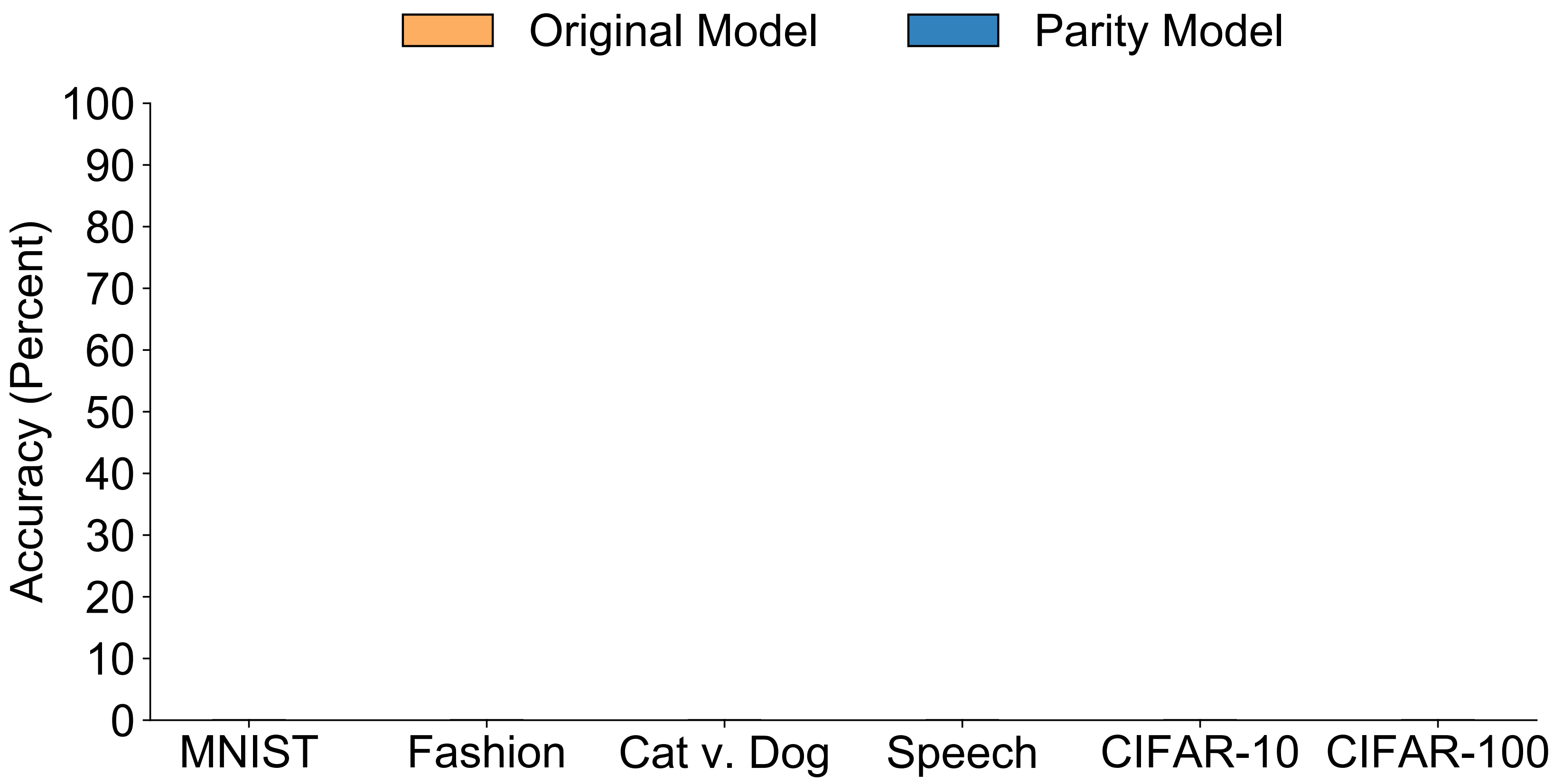


Evaluation

1. How accurate are reconstructions using parity models?

2. How much can parity models help reduce tail latency?

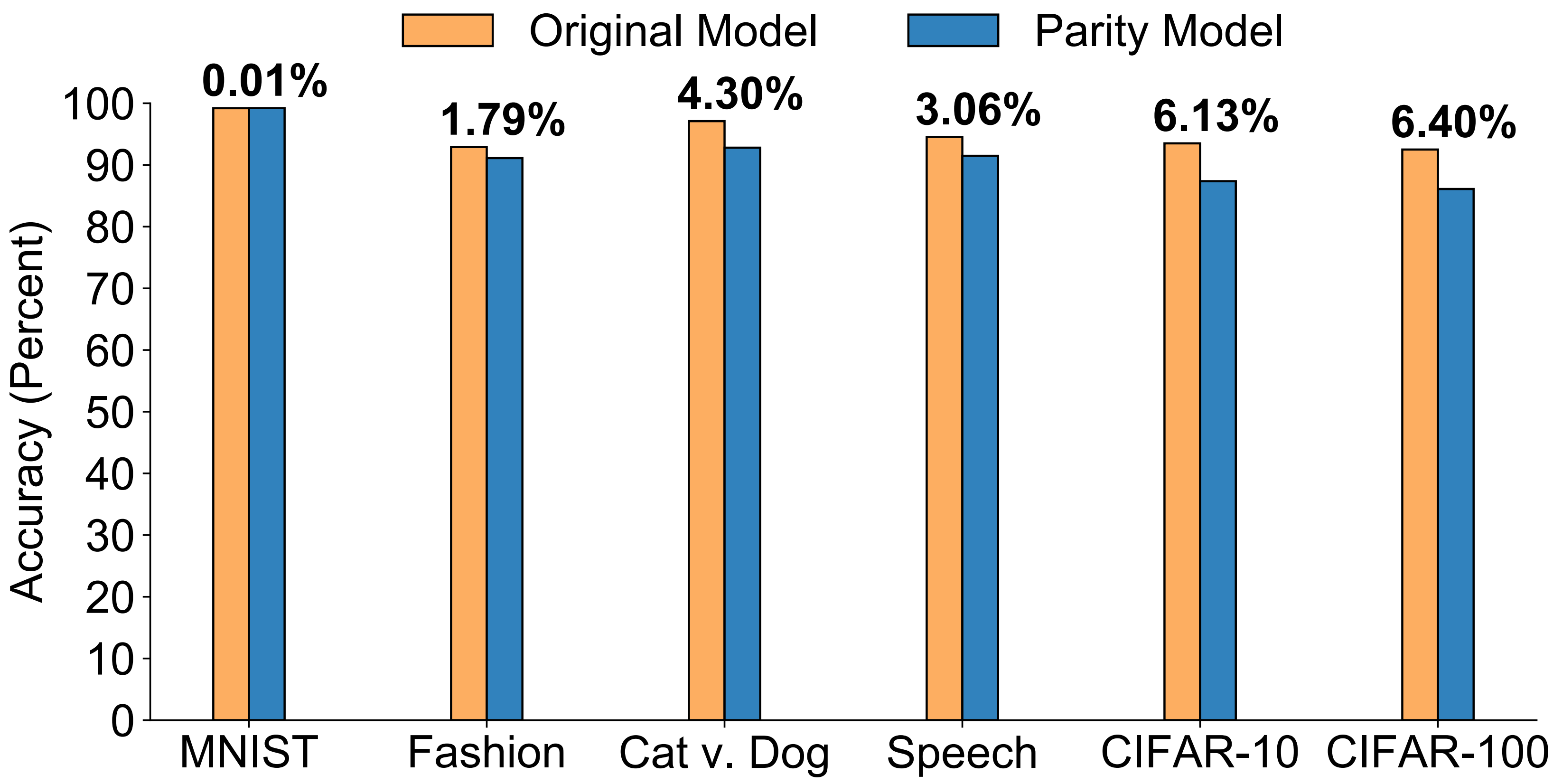
Evaluation of Accuracy



- Addition/subtraction code
- k = 2, r = 1 ($P = X_1 + X_2$)
- 2x less overhead than replication

Evaluation of Accuracy

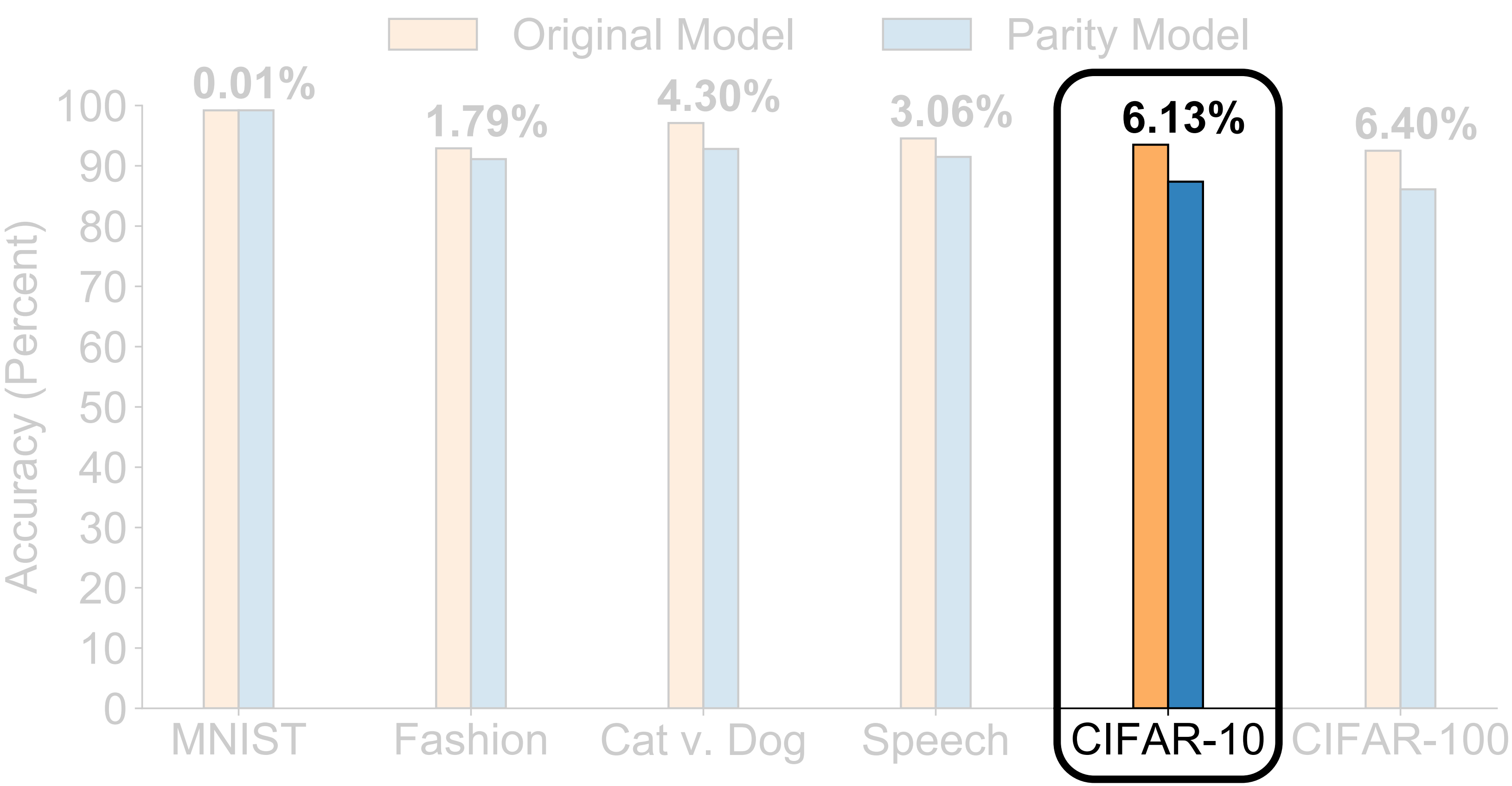
Parity model only comes into play when predictions are slow/failed



- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication

Evaluation of Accuracy

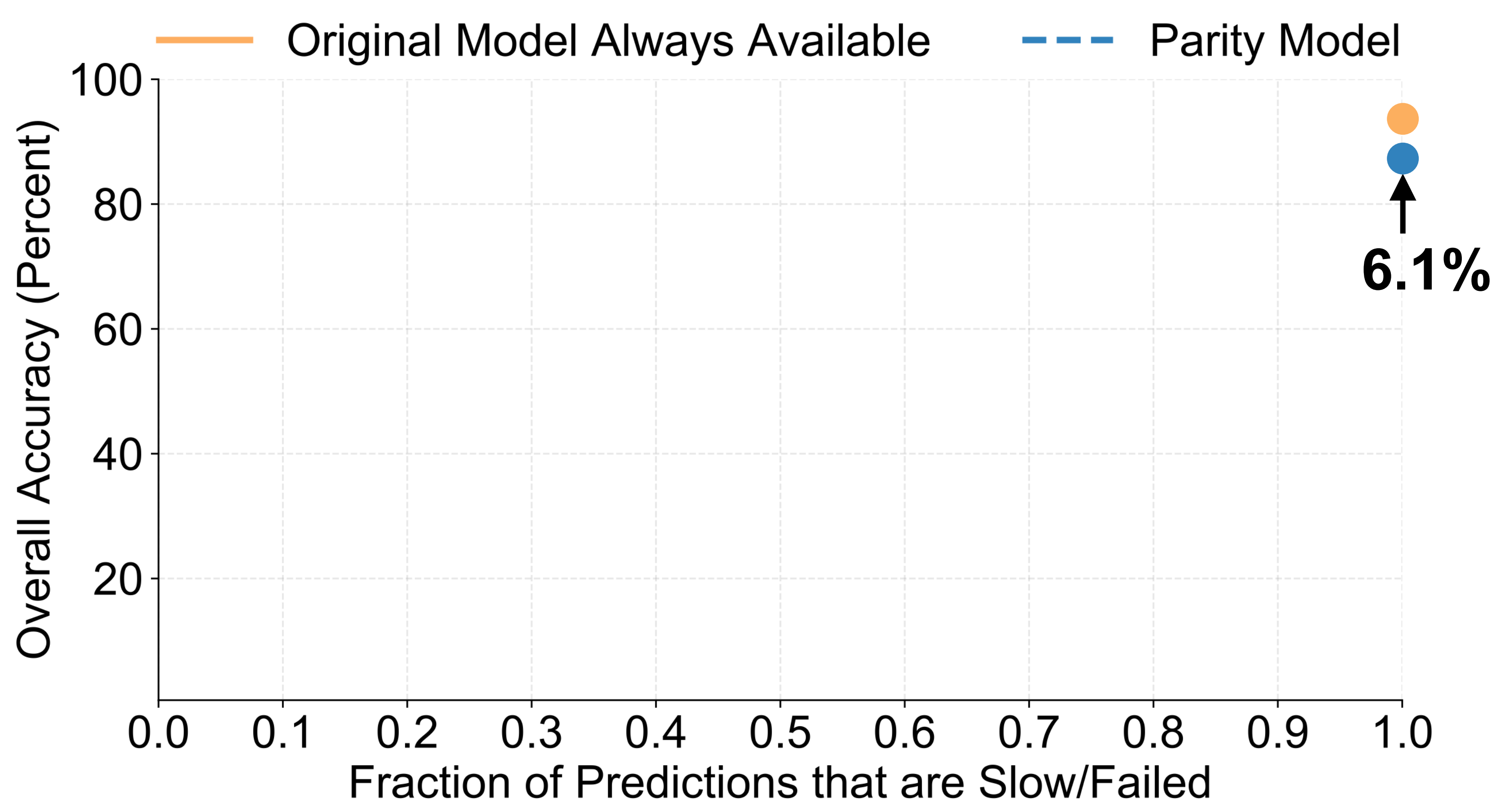
Parity model only comes into play when predictions are slow/failed



- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication

Evaluation of Overall Accuracy

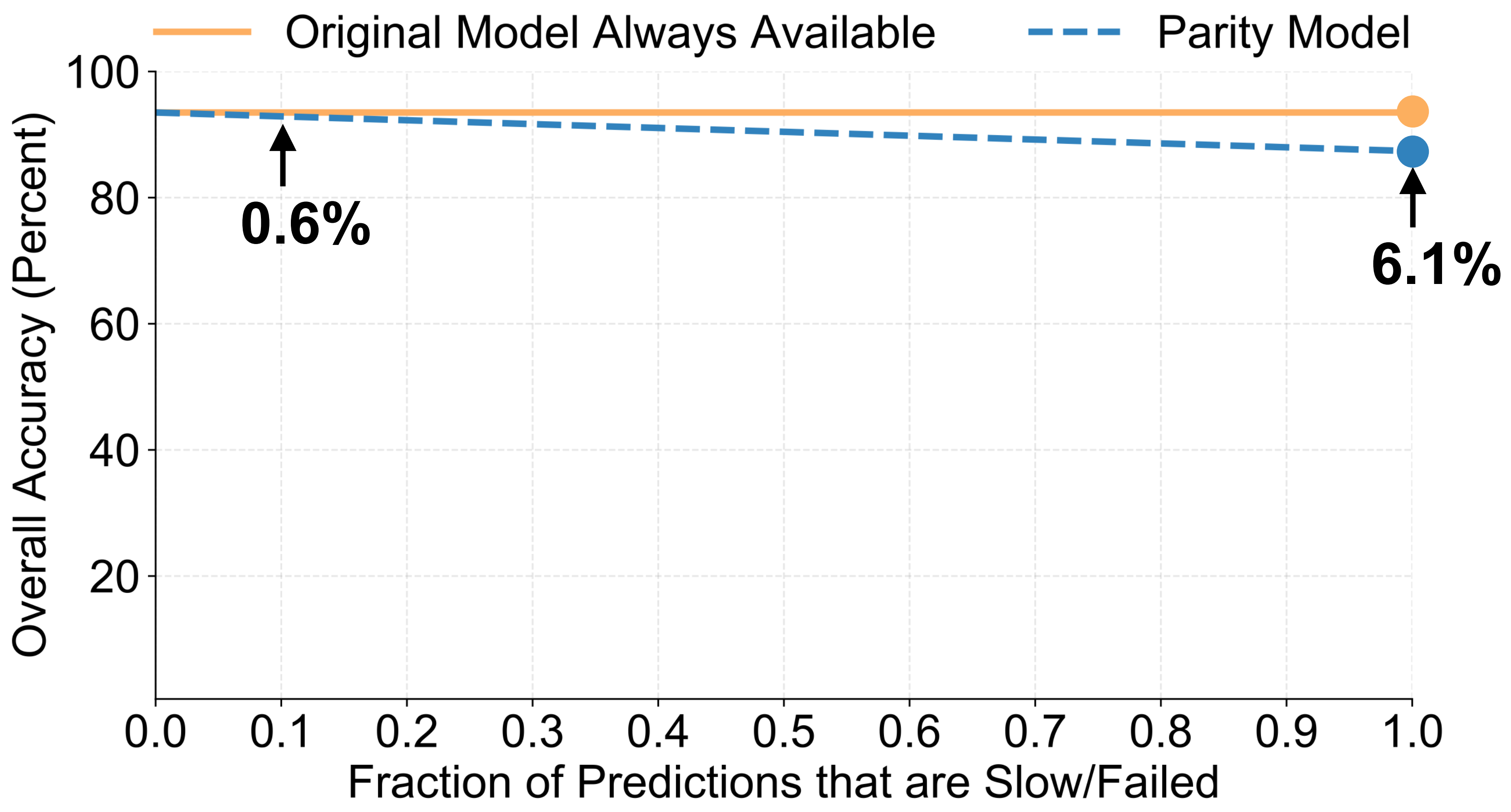
Parity model only comes into play when predictions are slow/failed



- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication

Evaluation of Overall Accuracy

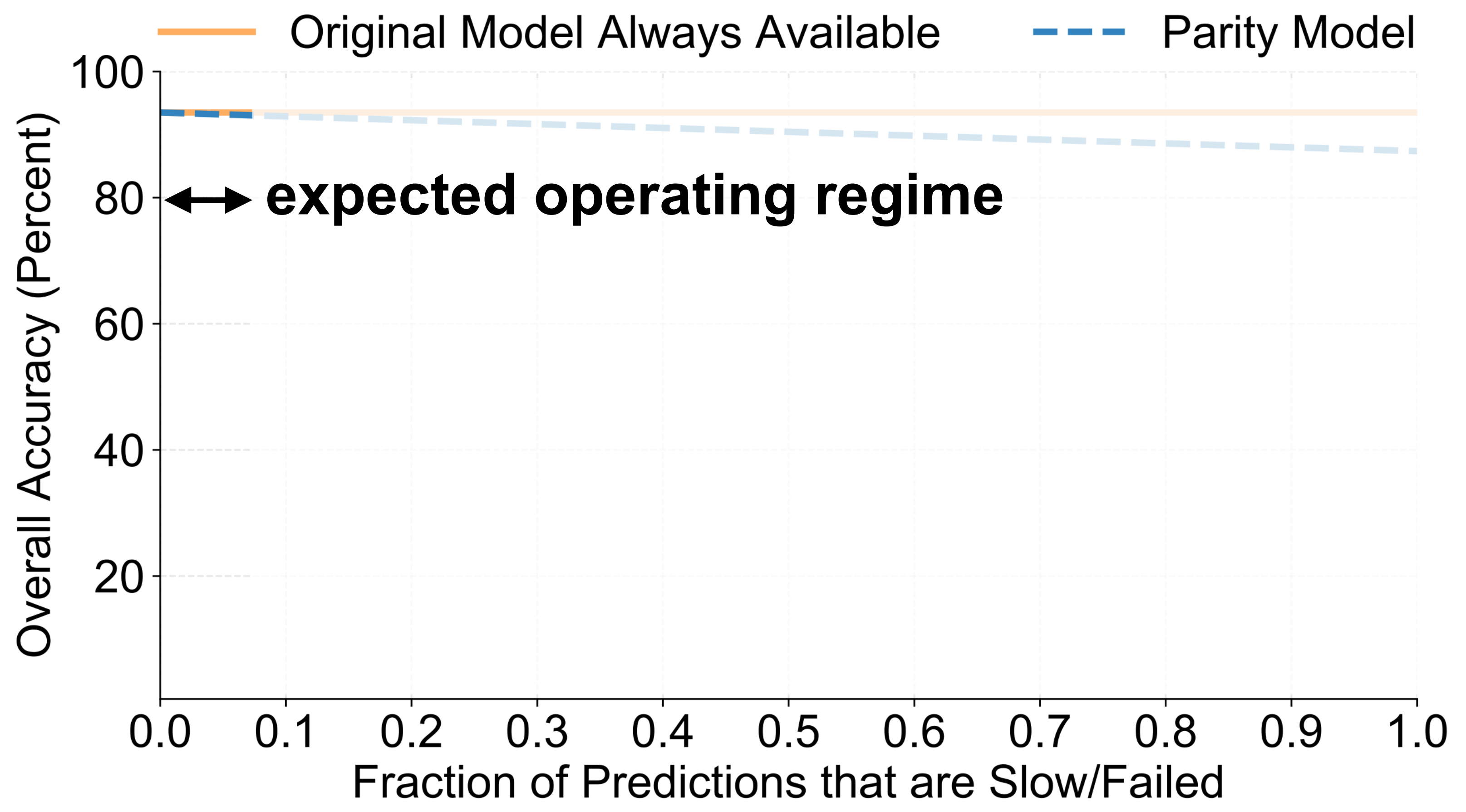
Parity model only comes into play when predictions are slow/failed



- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication

Evaluation of Overall Accuracy

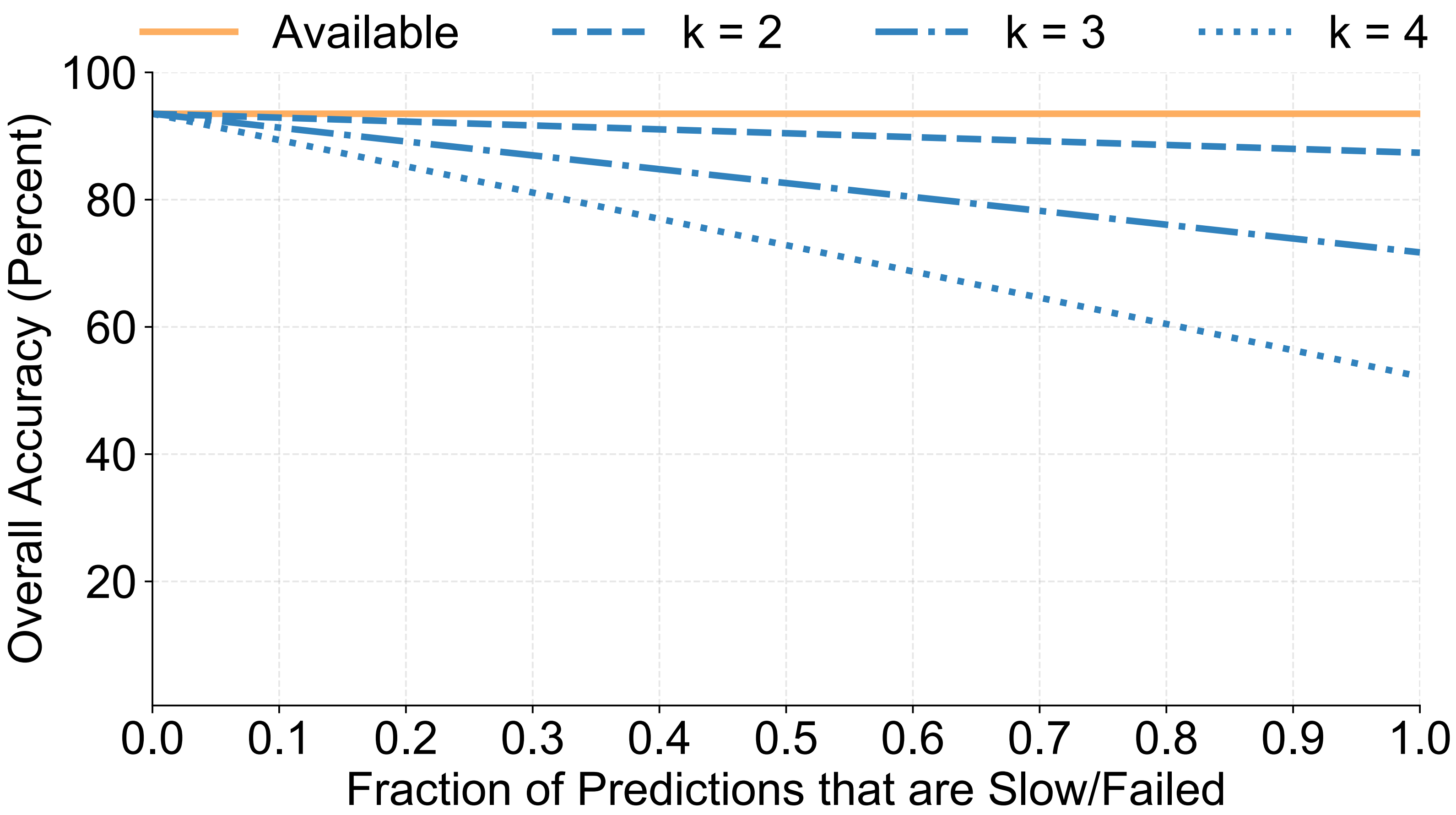
Parity model only comes into play when predictions are slow/failed



- Addition/subtraction code
- $k = 2$, $r = 1$ ($P = X_1 + X_2$)
- 2x less overhead than replication

Evaluation of Accuracy: Higher values of k

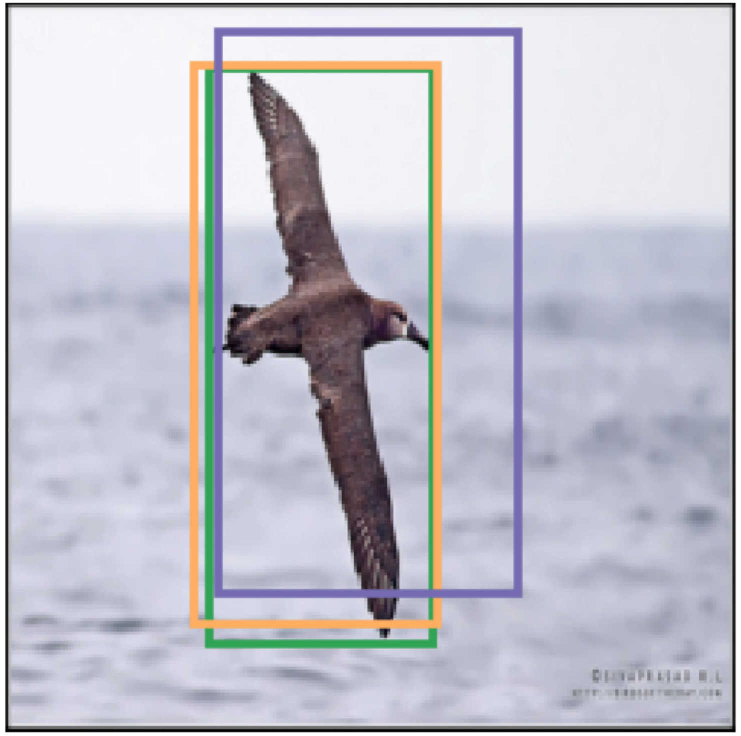
Tradeoff between resource-overhead, resilience, and accuracy



- Addition/subtraction code

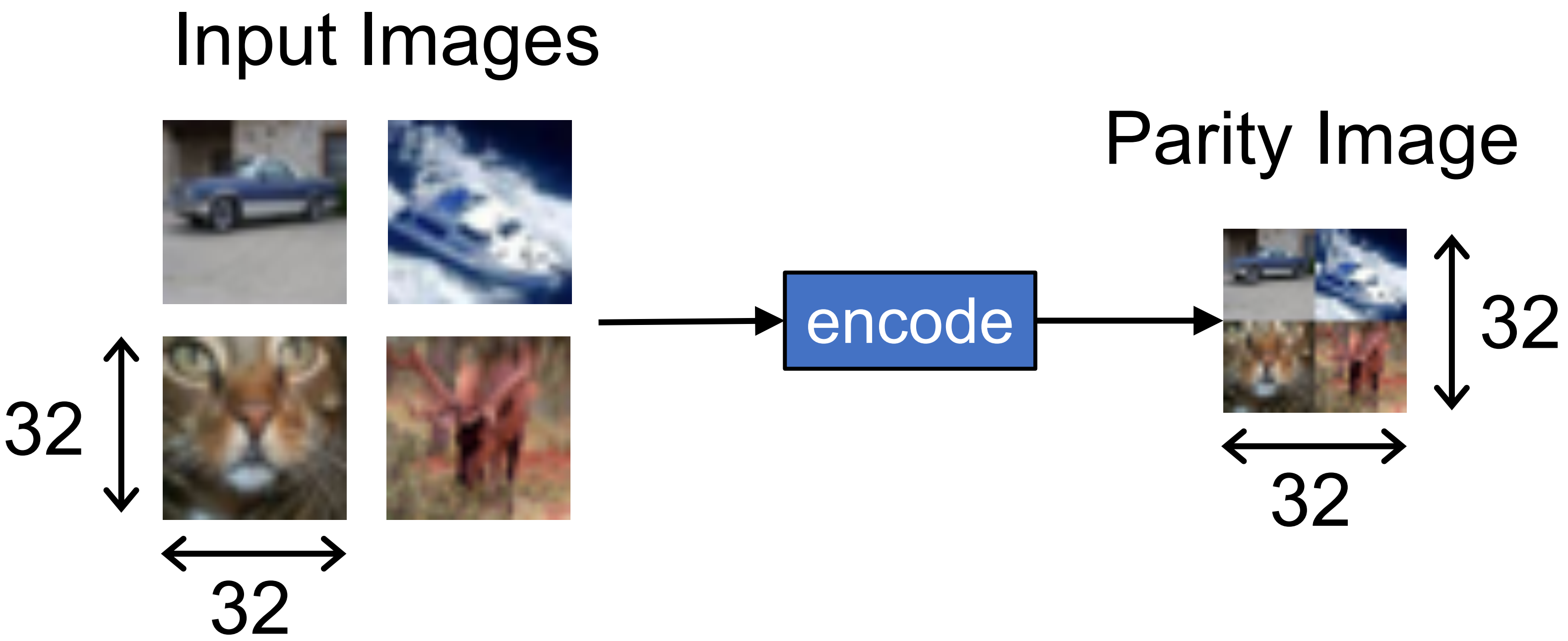
Evaluation of Accuracy: Object-localization

Ground Truth   Available   Parity Models



Evaluation of Accuracy: Task-specific encoder

22% accuracy improvement over addition/subtraction at k = 4

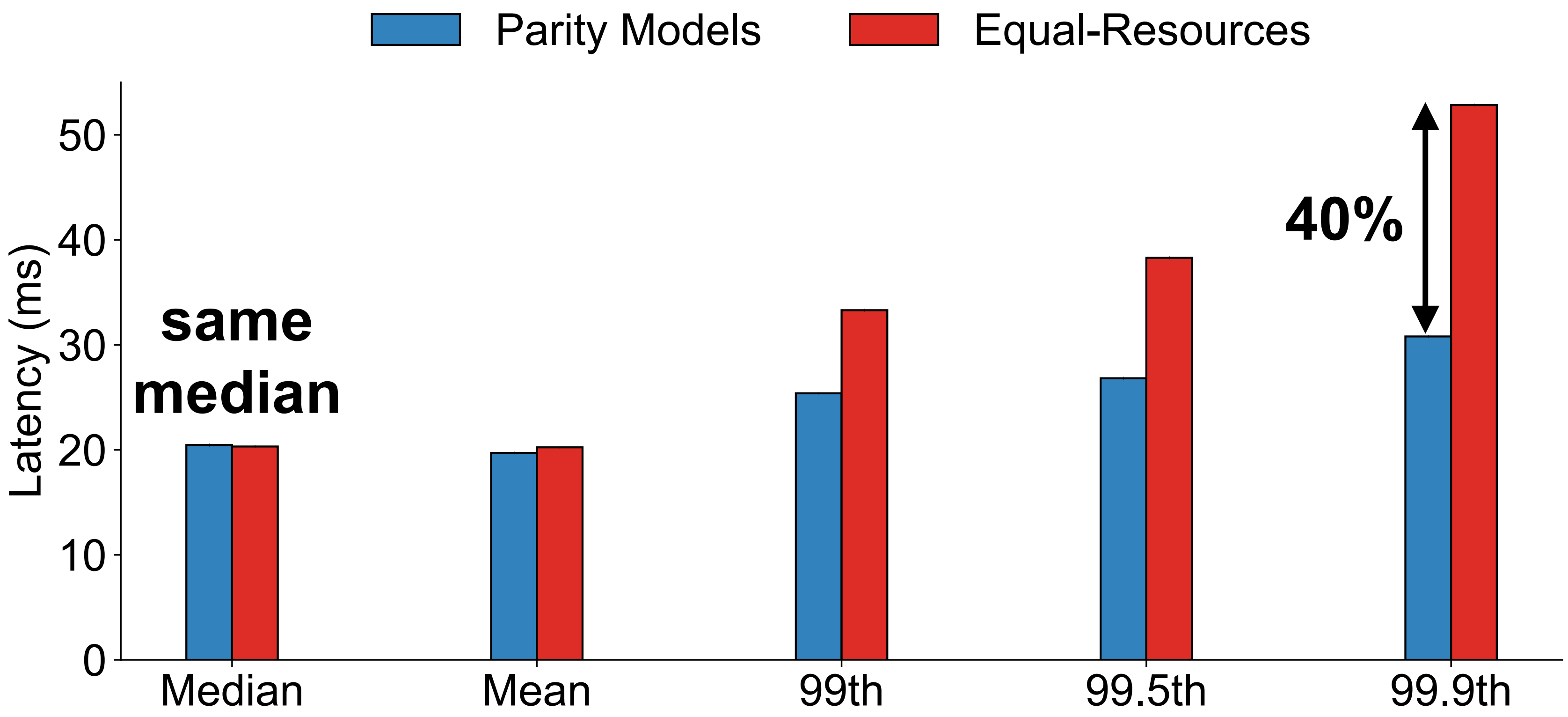


Evaluation of Tail Latency Reduction: Setup

- Implemented in Clipper prediction serving system
- Evaluate with 18-36 nodes on AWS with varying:
  - Inference hardware (GPUs, CPUs)
  - Query arrival rates
  - Batch sizes
  - Levels of load imbalance
  - Amounts of redundancy
  - Baseline approaches
- Baseline: approach with same number of resources as parity models

Evaluation of Tail Latency Reduction

In presence of resource contention

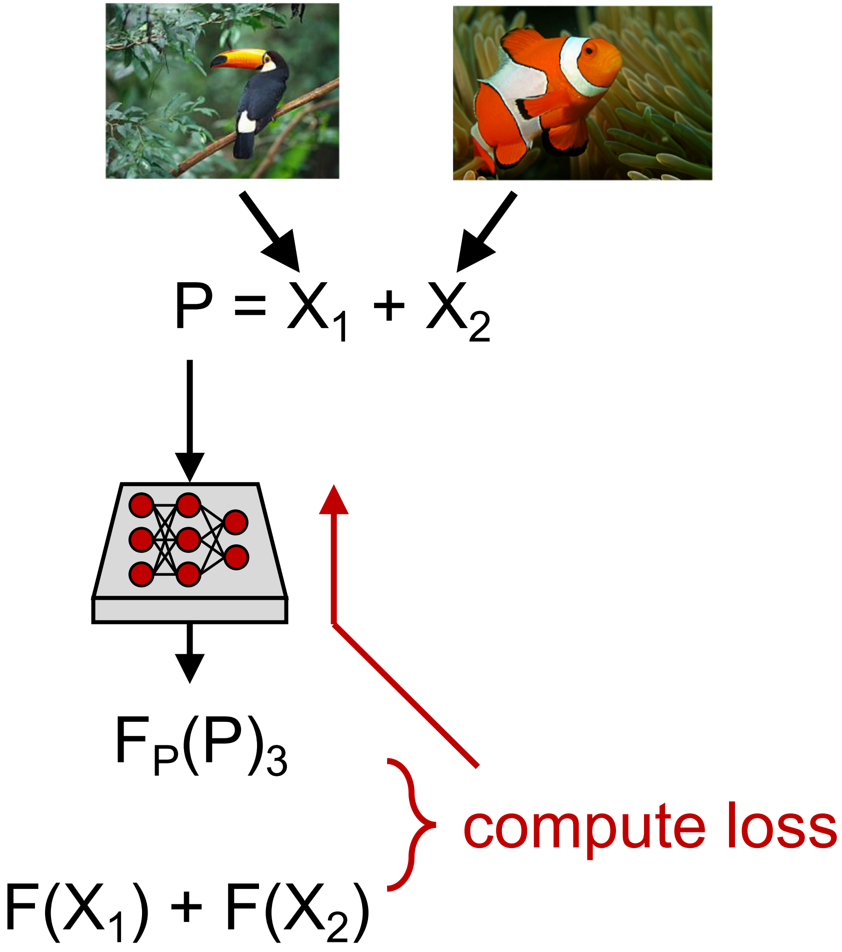


Limitations of current parity models framework

- Training a parity model is slow!
  - Dataset with N samples ⇒ parity model dataset with N^k samples

Training a parity model

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

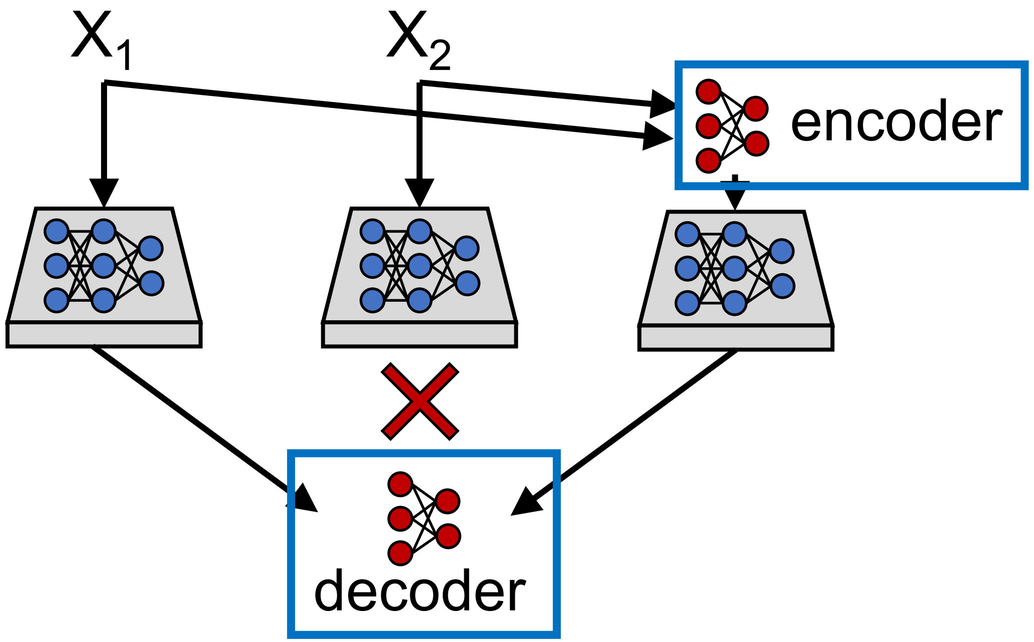


Limitations of current parity models framework

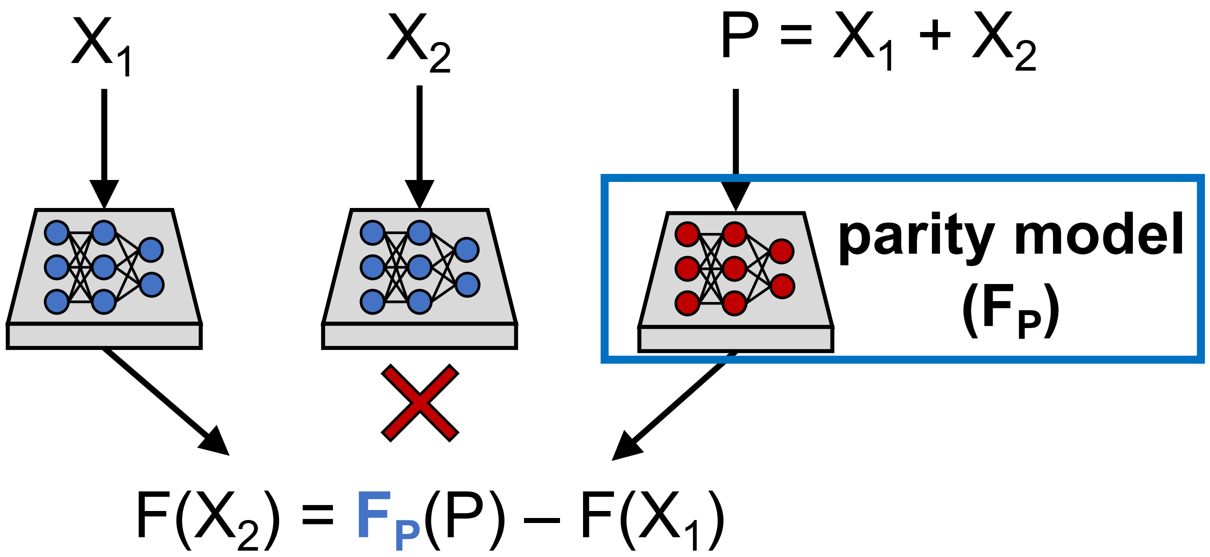
- Training a parity model is slow!
  - Dataset with N samples ⇒ parity model dataset with N^k samples
  - How to efficiently train under this combinatorial explosion?
- Theoretical understanding?
  - Subject to same problems as existing NNs (e.g., adversarial examples)
  - Can't bound inaccuracy
- Potential privacy concerns
  - Combining query A with query B into a parity query might leak info
- More research needed to tackle the above

Landscape of learning in coded-computation

Learn a code



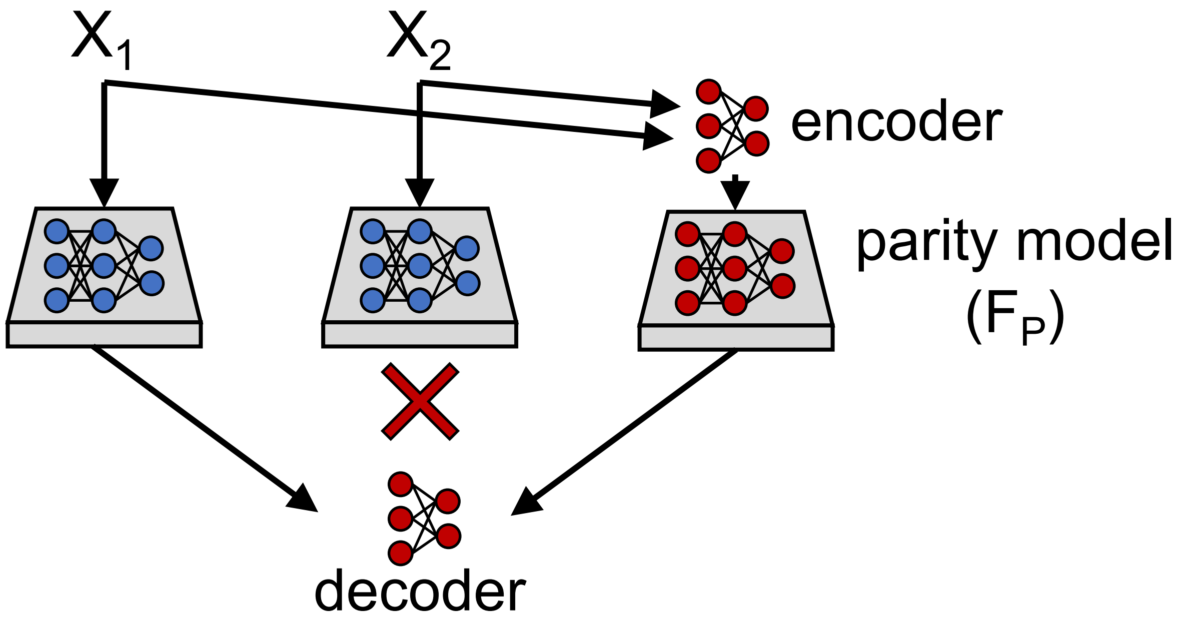
Learning a parity model



Landscape of learning in coded-computation

Jointly learn encoders, decoders, and parity models?

Balance complexity, execution time across components



Parity Models: Erasure-Coded Resilience for Prediction Serving Systems

- Coded-computation is promising, but current approaches cannot support popular machine learning models like neural networks
- Parity models: judicious use of learning allows for accurate reconstruction of unavailable ML inference predictions
- Enables erasure-coded resilience in prediction serving systems

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