Parity Models
Erasure-Coded Resilience for Prediction Serving Systems

Jack Kosaian  Rashmi Vinayak  Shivaram Venkataraman
Inference: using a trained ML model
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

queries → predictions

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>dog</th>
<th>bird</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.15</td>
<td>0.8</td>
<td>0.05</td>
<td></td>
</tr>
</tbody>
</table>
Inference: using a trained ML model

queries

predictions

0.15 0.8 0.05

cat dog bird
Inference used in latency-sensitive apps
Inference used in latency-sensitive apps

- translation
- search
- ranking
Inference used in latency-sensitive apps

Translation

Search

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
- Google
- AWS
- Microsoft Azure
Prediction serving systems: inference in clusters
Prediction serving systems: inference in clusters

Frontend
Prediction serving systems: inference in clusters

Frontend

model instances
Prediction serving systems: inference in clusters

queries

Frontend

model instances
Prediction serving systems: inference in clusters
Prediction serving systems: inference in clusters

queries → Frontend → predictions

model instances
Slowdowns and failures in inference
Slowdowns and failures in inference

queries → predictions

Frontend

network contention

compute contention
Slowdowns and failures in inference

queries → predictions

network contention
compute contention
failures
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

D₁ △ P
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

queries → Encoder

Frontend

parity query
End goal: erasure-coded prediction serving
End goal: erasure-coded prediction serving

queries

Frontend

Encoder

parity model
End goal: erasure-coded prediction serving
End goal: erasure-coded prediction serving

queries → Frontend → Encoder
End goal: erasure-coded prediction serving

queries

Frontend

Encoder

Decoder

parity model
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

D_1 
\downarrow
D_1

D_2 
\downarrow
D_2

encoding

“parity” P = D_1 + D_2

P
Quick recap of erasure codes

\[ P = D_1 + D_2 \]

encoding

"parity"
Quick recap of erasure codes

encoding

D₁

D₂

P

“parity”  \( P = D₁ + D₂ \)

D₂ = \( P - D₁ \)

decoding
Quick recap of erasure codes: parameter $k$

$$P = D_1 + D_2 + \ldots + D_k$$
Quick recap of erasure codes: benefits

Replication

Erasure Coding

D_1 \rightarrow D_1 \rightarrow D_1 \rightarrow D_2 \rightarrow D_2

D_1 \rightarrow D_1 \rightarrow D_2 \rightarrow P = D_1 + D_2

same resilience

lower resource-overhead
Using erasure codes for inference
Using erasure codes for inference
Using erasure codes for inference

models
Using erasure codes for inference

\[ X_1 \quad X_2 \] queries

\[ F \quad F \quad F \] models
Using erasure codes for inference

$X_1$, $X_2$ \hspace{1cm} \textit{queries}

$F$, $F$, $F$ \hspace{1cm} \textit{models}
Using erasure codes for inference

\[ F(X_1) \quad F(X_2) \]

queries

models

predictions
Using erasure codes for inference

Goal: preserve results of inference over queries
Using erasure codes for inference

Encode queries

X₁ → F → F(X₁)

X₂ → F → F(X₂)

"parity query"
Using erasure codes for inference

\[ \text{decode results of \ inference over queries} \]
Traditional coding vs. codes for inference

Codes for storage

D₁ \rightarrow D₂ \leftarrow \text{encode} \rightarrow \text{decode} \rightarrow D₂

D₁ \rightarrow D₂ \rightarrow \text{encode} \rightarrow \text{decode} \rightarrow \text{P}

Codes for inference

X₁ \rightarrow X₂ \leftarrow \text{encode} \rightarrow \text{decode} \rightarrow F(X₂)

X₁ \rightarrow X₂ \rightarrow \text{encode} \rightarrow \text{decode} \rightarrow F(P)

Need to handle computation over inputs
Traditional coding vs. codes for inference

Codes for storage

$D_1$ \rightarrow $D_2$

**encode**

$D_1$

**decode**

$D_2$

Codes for inference

$X_1$ \rightarrow $X_2$

**encode**

$F(X_1)$ \rightarrow $F(P)$

**decode**

$F(X_2)$

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

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

Current handcrafted coded-computation approaches cannot support neural networks
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

X₁ X₂

encoder

decoder
Learning an erasure code?

Design encoder and decoder as neural networks

- **Accurate**
- **Expensive encoder/decoder**

Diagram:
- Encoder
- Decoder
- X₁
- X₂
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”

\[
P = X_1 + X_2
\]

\[
F(X_2) = F_P(P) - F(X_1)
\]
Learn computation over parities

Use simple, fast encoders and decoders
Learn computation over parities: “parity model”

\[ P = X_1 + X_2 \]

\[ F(X_2) = F_P(P) - F(X_1) \]
Learn computation over parities

Use simple, fast encoders and decoders

Learn computation over parities: “parity model”

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

$P = X_1 + X_2$
Learn computation over parities

Use simple, fast encoders and decoders
Learn computation over parities: “parity model”

Accurate
Efficient encoder/decoder

\[ X_1 \rightarrow \begin{array}{c}
F(X_2) = F_P(P) - F(X_1)
\end{array} \]

parity model \((F_P)\)
Designing parity models

\[
P = X_1 + X_2
\]

\[
F(X_2) = F_P(P) - F(X_1)
\]
Designing parity models

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

\[
P = X_1 + X_2
\]

\[
F(X_2) = F_P(P) - F(X_1)
\]
Designing parity models

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

\[ P = X_1 + X_2 \]

\[ F(X_2) = F_P(P) - F(X_1) \]
Designing parity models

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

\[
F(P) = F(X_1) + F(X_2)
\]

\[
F(X_2) = F_P(P) - F(X_1)
\]

\[
F_P(P) = F(X_1) + F(X_2)
\]
Designing parity models

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

\[
F(P) = F(X_1) + F(X_2)
\]

\[
F(X_2) = F_P(P) - F(X_1)
\]

\[
F_P(P) = F(X_1) + F(X_2)
\]

Learn a parity model

parity model \( (F_P) \)
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

\[ P = X_1 + X_2 \]

Desired output: \( F(X_1) + F(X_2) \)
1. Sample inputs and encode

\[ P = X_1 + X_2 \]

**Desired output:** \( F(X_1) + F(X_2) \)

**predictions**

\[
\begin{array}{ccc}
0.8 & 0.15 & 0.05 \\
0.2 & 0.7 & 0.1 \\
\end{array}
\]
Training a parity model

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

\[ P = X_1 + X_2 \]

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

Predictions:

- \( F_P(P)_1 \)
- \( 0.8 \)
- \( 0.15 \)
- \( 0.05 \)
- \( 0.2 \)
- \( 0.7 \)
- \( 0.1 \)
Training a parity model

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

\[ P = X_1 + X_2 \]

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

Predictions

- 0.8
- 0.15
- 0.05

Queries

- 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

\[ P = X_1 + X_2 \]

Desired output: \[ F(X_1) + F(X_2) \]

Predictions: \[ \begin{bmatrix} 0.8 & 0.15 & 0.05 \end{bmatrix} \]

Queries: \[ \begin{bmatrix} 0.2 & 0.7 & 0.1 \end{bmatrix} \]
Training a parity model

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

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

<table>
<thead>
<tr>
<th>predictions</th>
<th>0.8</th>
<th>0.15</th>
<th>0.05</th>
</tr>
</thead>
<tbody>
<tr>
<td>queries</td>
<td>0.2</td>
<td>0.7</td>
<td>0.1</td>
</tr>
</tbody>
</table>

\( P = X_1 + X_2 \)
1. Sample inputs and encode
2. Perform inference with parity model
3. Compute loss
4. Backpropagate loss
5. Repeat

Training a parity model

\[ F(X_1) + F(X_2) \]

\[ P = X_1 + X_2 \]

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

Predictions:

\[
\begin{array}{c|c|c}
0.15 & 0.8 & 0.05 \\
\end{array}
\quad
\begin{array}{c|c|c}
0.3 & 0.5 & 0.2 \\
\end{array}
\]
Training a parity model

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

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

predictions: 0.03 0.02 0.95 0.3 0.3 0.4
Training a parity model: higher parameter $k$

Can use higher code parameter $k$

$P = X_1 + X_2 + X_3 + X_4$

$F_P(P)_1$

Desired output: $F(X_1) + F(X_2) + F(X_3) + F(X_4)$
Training a parity model: different encoders

\[ P = F_P(P) \]
Training a parity model: different encoders

\[ P = F_P(P)_1 \]
Training a parity model: different encoders
Training a parity model: different encoders

Can **specialize** encoders and decoders to **inference** task at hand

\[ P = F_{P(P)} \]
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

queries

Frontend

Encoder
Decoder

parity query

parity model
Evaluation

1. How accurate are reconstructions using parity models?

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

- Deployed Model
- Parity Model Reconstruction

Accuracy (Percent)

MNIST  Fashion  Cat v. Dog  Speech  CIFAR-10  CIFAR-100
Accuracy of parity models

Accuracy (%)

- MNIST: Deployed Model 0.01%, Parity Model Reconstruction 1.79%
- Fashion: Deployed Model 4.30%, Parity Model Reconstruction 3.06%
- Cat v. Dog: Deployed Model 6.13%, Parity Model Reconstruction 6.40%
- Speech: Deployed Model 6.13%, Parity Model Reconstruction 6.40%
- CIFAR-10: Deployed Model 6.13%, Parity Model Reconstruction 6.40%
- CIFAR-100: Deployed Model 6.13%, Parity Model Reconstruction 6.40%
Accuracy of parity models

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

Example: assuming 10% slow predictions \( \rightarrow \) at most 0.7% lower overall accuracy.
Tail latency reduction

In presence of resource contention

- Parity Models
- Equal-Resources
Tail latency reduction

In presence of resource contention

- Parity Models
- Equal-Resources

Latency (ms)

- Median
- Mean
- 99th
- 99.5th
- 99.9th

40% reduction in tail latency
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