Cocktail: A Multidimensional Optimization for Ensemble Learning

Jashwant Raj Gunasekaran, Cyan Subhra Mishra
Prashanth Thinakaran, Bikash Sharma, Mahmut T. Kandemir, Chita R. Das
Penn State, CSE, High Performance Computing Lab

nsdi ’22
19th USENIX Symposium on Networked Systems Design and Implementation
April’6-2022 | Renton, WA, USA
MODEL SERVING HOSTED ON CLOUD

Different Model Serving-based Applications

- Image Classification
- Natural Language Processing
- Question Answering

Request

Response

Users

MODELSERVINGHOSTEDONCLOUD
MODEL SERVING HOSTED ON CLOUD

Different Model Serving-based Applications

Image Classification

Natural Language Processing

Question Answering

Request

Response

Training
MODEL SERVING HOSTED ON CLOUD

Different Model Serving-based Applications

- Service Level Objective (SLO) 1500ms
- Image Classification
- Natural Language Processing
- Question Answering

Users → Request → Response
**Model Serving Hosted on Cloud**

- **Resources for Applications**
  - VMs
  - Containers

- **Different Model Serving-based Applications**
  - Image Classification
  - Natural Language Processing
  - Question Answering

- **Request-Response Cycle**
  - Users
  - Service Level Objective (SLO): 1500ms
  - Training Model Serving Hosted on Cloud

2
Different Model Serving-based Applications

- Image Classification
- Natural Language Processing
- Question Answering

Request → Users → Response

Service Level Objective (SLO) 1500ms

Provisioning Time

Resources for Applications

VMs

Containers
Different Model Serving-based Applications

- Image Classification
- Natural Language Processing
- Question Answering

Users

Request

Service Level Objective (SLO) 1500ms

Response

Model Latency

Provisioning Time

Resources for Applications
MODEL SERVING CHALLENGES

Model Serving

- Resource Type
- Model Type

Provisioning Latency
Model Latency
Accuracy

Cost
MODEL SERVING CHALLENGES

- Resource Type
- Model Type

Model Serving

Provisioning Latency
- Wang et al. ATC'19
- Jain et al. Middleware'19
- Li et al. SC'21

Model Latency
- Accuracy

Cost

PennState
College of Engineering
Model Serving Challenges

- Model Serving
  - Resource Type
  - Model Type

Provisioning Latency

- Model Latency
- Accuracy

Cost

Wang et al. ATC’19
Jain et al. Middleware’19
Li et al. SC’21
In Netflix, 75% of viewer activity is based on these accurate suggestions.
How to improve accuracy with low latency and low cost?

Model Serving Challenges

- Wang et al. ATC’19
- Jain et al. Middleware’19
- Li et al. SC’21
PRIOR WORK IN MODEL SERVING

- **InFaas** uses different resource types to ensure low latency at low cost.
- **Clipper** achieves higher accuracy while compromising latency.

Crankshaw et al. CIDR’15, NSDI’17, SoCC’20
Yadawkar et al. ATC’21
Prior Work in Model Serving

- **InFaas** uses different resource types to ensure low latency at low cost.
- **Clipper** achieves higher accuracy while compromising latency.

Crankshaw et al. CIDR’15, NSDI’17, SoCC’20
Yadawkar et al. ATC’21
**Prior Work in Model Serving**

- **InFaas** uses different resource types to ensure low latency at low cost.
- **Clipper** achieves higher accuracy while compromising latency.

Crankshaw et al. CIDR’15, NSDI’17, SoCC’20
Yadawkar et al. ATC’21
- **InFaas** uses different resource types to ensure low latency at low cost.
- **Clipper** achieves higher accuracy while compromising latency.

Crankshaw et al. CIDR’15, NSDI’17, SoCC’20
Yadawkar et al. ATC’21
Prior Work in Model Serving

How to do ensembling?

- **InFaas** uses different resource types to ensure low latency at low cost.
- **Clipper** achieves higher accuracy while compromising latency.

Crankshaw et al. CIDR’15, NSDI’17, SoCC’20
Yadawkar et al. ATC’21
Model Ensembling Framework

Model Selection
- NasNetMobile
- MobileNetV2
- InceptionV3

Requests

Host Server

Majority Voting

Cloud Resources for Individual Models (Virtual Machines)

Model Ensembling Framework
Model Ensembling Framework

**Model Selection**
- NasNetMobile
- MobileNetV2
- InceptionV3

**Requests**

**Host Server**

**Cloud Resources for Individual Models (Virtual Machines)**

**Majority Voting**

**Aggregator**
Model Ensembling Framework

Model Selection
- NasNetMobile
- MobileNetV2
- InceptionV3

Host Server

Requests

Aggregator

Majority Voting

Cloud Resources for Individual Models (Virtual Machines)

Model Ensembling Framework
Model Ensembling Framework

Model Selection
- NasNetMobile
- MobileNetV2
- InceptionV3

Requests

Host Server

Aggregator

Majority Voting

Cloud Resources for Individual Models (Virtual Machines)

Model-1

Model-2

Model-N

Model-1

Model-2

Model-N

Model-1

Model-2

Model-N
Model Ensembling Framework

Model Selection
- NasNetMobile
- MobileNetV2
- InceptionV3

Host Server

Requests

Aggregator

Majority Voting

Cloud Resources for Individual Models (Virtual Machines)
Model Ensembling Framework

Model Selection
- NasNetMobile
- MobileNetV2
- InceptionV3

Requests

Host Server

Cloud Resources for Individual Models (Virtual Machines)

Aggregator

Majority Voting

Model-1

Model-2

Model-N

NasNetMobile
MobileNetV2
InceptionV3
High Resource Footprint
What about Model Selection?

Cloud Resources for Individual Models (Virtual Machines)
IEEE Access'18 Benchmark Analysis of Representative Deep Neural Network Architectures
Most accurate model

~2x parameters, latency

~2% more accuracy

IEEE Access'18 Benchmark Analysis of Representative Deep Neural Network Architectures
Most accurate model

- ~2x parameters, latency
- ~2% more accuracy

- How to bridge the 2% accuracy gap?
- What about cost?
MODEL SPACE EXPLORATION

Most accurate model

-~2x parameters, latency
-~2% more accuracy

How to ensemble?

- What about cost?

IEEE Access ’18 Benchmark Analysis of Representative Deep Neural Network Architectures
Model Set: Top 12 frequently used models from Keras Tensorflow

Choose baseline models in decreasing order of accuracy

Combine all models which are under the latency of baseline model.

<table>
<thead>
<tr>
<th>Model (Acronym)</th>
<th>Params (10k)</th>
<th>Top-1 Accuracy(%)</th>
<th>Latency (ms)</th>
<th>$P_f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNetV1 (MNet)</td>
<td>4,253</td>
<td>70.40</td>
<td>43.45</td>
<td>10</td>
</tr>
<tr>
<td>MobileNetV2 (MNetV2)</td>
<td>4,253</td>
<td>71.30</td>
<td>41.5</td>
<td>10</td>
</tr>
<tr>
<td>NASNetMobile (NASMob)</td>
<td>5,326</td>
<td>74.40</td>
<td>78.18</td>
<td>3</td>
</tr>
<tr>
<td>DenseNet121 (DNet121)</td>
<td>8,062</td>
<td>75.00</td>
<td>102.35</td>
<td>3</td>
</tr>
<tr>
<td>DenseNet201 (DNet201)</td>
<td>20,242</td>
<td>77.30</td>
<td>152.21</td>
<td>2</td>
</tr>
<tr>
<td>Xception (Xcep)</td>
<td>22,910</td>
<td>79.00</td>
<td>119.2</td>
<td>4</td>
</tr>
<tr>
<td>Inception V3 (Incep)</td>
<td>23,851</td>
<td>77.90</td>
<td>89</td>
<td>5</td>
</tr>
<tr>
<td>ResNet50-V2 (RNet50)</td>
<td>25,613</td>
<td>76.00</td>
<td>89.5</td>
<td>6</td>
</tr>
<tr>
<td>ResNet50 (RNet50)</td>
<td>25,636</td>
<td>74.90</td>
<td>98.22</td>
<td>5</td>
</tr>
<tr>
<td>IncepResNetV2 (IRV2)</td>
<td>55,873</td>
<td>80.30</td>
<td>151.96</td>
<td>1</td>
</tr>
<tr>
<td>NasNetLarge (NasLarge)</td>
<td>343,000</td>
<td>82.00</td>
<td>311</td>
<td>1</td>
</tr>
</tbody>
</table>
Combine all models which are under the latency of baseline model.

Model Set: Top 12 frequently used models from Keras Tensorflow
Choose baseline models in decreasing order of accuracy

Latency Comparison

Accuracy Comparison

Latency (ms)

Top1-Accuracy
Model Set: Top 12 frequently used models from Keras Tensorflow

Choose baseline models in decreasing order of accuracy

Combine all models which are under the latency of baseline model.

**Latency Comparison**
- Single
- Ensemble

**Accuracy Comparison**
- Top1- Accuracy
What about Cost?
FULL ENSEMBLING COST

- Single-OD
- Ensemble-OD
- Ensemble-spot

Cost ($)

NASLarge IRV2 XceptionDNet121 NASMob
Ensembling is up-to 2x expensive.
Ensembling is up-to 2x expensive.

Spot instances can reduce cost by 2x.
Ensembling is up to 2x expensive.

Spot instances can reduce cost by 2x.

Transient instances- 70-80% cheaper. Can be revoked with short notice.
## What can we do?

<table>
<thead>
<tr>
<th>Baseline(BL)</th>
<th>NASLarge</th>
<th>IRV2</th>
<th>Xception</th>
<th>DNet121</th>
<th>NASMob</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Models</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>
WHAT CAN WE DO?

<table>
<thead>
<tr>
<th>Baseline(BL)</th>
<th>NASLarge</th>
<th>IRV2</th>
<th>Xception</th>
<th>DNet121</th>
<th>NASMob</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Models</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

- Do we need so many models?
- How to autoscale resources for each model?
- How to handle instance failures?
Compared to Full-Ensemble (N models)

Most accurate $N/2$ models
**STATIC ENSEMBLING**

Compared to Full-Ensemble (N models)

- **Most accurate N/2 models**
  - Accuracies: NASLarge, IRV2, Xception, DNet121, NASMob

Accuracy (°)
STATIC ENSEMBLING

Compared to Full-Ensemble (N models)

Most accurate N/2 models

How to dynamically select the models?
DYNAMIC MODEL SELECTION
DYNAMIC MODEL SELECTION
Dynamic Model Selection

![ImageNet dataset example]

![Bar chart showing accuracy for different classes and models]

- **Peacock**
- **Panda**
- **Quill Class**
- **Slug**
- **Cup**

Models:
- MNetV2
- IRV2
- NASLarge
Dynamic Model Selection

Mobilenet (MNet) ⇒ Slug

Mobilenet (MNet) ⇒ Quill
Leverage Class-wise Accuracy

Class

- Mobilenet (MNet) → Slug
- Mobilenet (MNet) → Quill
COCKTAIL- MULTIDIMENSIONAL OPTIMIZATION FOR ENSEMBLE LEARNING IN CLOUD

User Requests

Dynamic Model Selection

Aggregator

Weight Matrix

Queries

w1  w2  w3  w4  \ldots  wk

output
Cocktail- Multidimensional Optimization for Ensemble Learning in Cloud

User Requests

Dynamic Model Selection

Aggregator

Weight Matrix

Class-wise dictionary

Weighted Selection

Queries
OBJECTIVE FUNCTION

- Three optimization points: cost, latency and accuracy
- Metrics $\mu_1 = \frac{Acc}{Lat}$; $\mu_2 = k \sum_{i=1}^{n} \frac{\text{inst}\_\text{cost}}{p_{m_i}}$
  - Where we use n models (model $m_i$, $i = 1$ to $n$) to ensemble
  - Each model $m_i$ has a packing factor of $p_{m_i}$. $k$ is a constant which is dependent on the resource and the instance type
- Our objective:

  Obj1:
  $$\max \mu_1 : \begin{cases} Acc \geq Acc_{SLO} \pm Acc_{\text{margin}} \\ Lat \leq Lat_{SLO} \pm Lat_{\text{margin}} \end{cases}$$

  Obj2:
  $$\min_{\text{Cost} \leq \text{Cost}_{\text{Baseline}}} \mu_2 : Acc \geq Acc_{SLO} \pm Acc_{\text{margin}}$$
Can we select the models apriori?
Ensuring $\text{Acc} \geq \text{Acc}_{\text{SLO}} \pm \text{Acc}_{\text{margin}}$

- Say we have ‘$n$’ models with minimum accuracy of ‘$a$’
- We use majority voting ensemble: we need at least $\frac{n}{2} + 1$ give correct results.

- Prob correct $= \binom{n}{\lfloor \frac{n}{2} + i \rfloor} a^{\lfloor \frac{n}{2} + 1 \rfloor + i} (1 - a)^n - \binom{n}{\lfloor \frac{n}{2} + 1 \rfloor + i}; \text{for } i = 0 \text{ to } \lceil n/2 + 1 \rceil$
COCKTAIL - MULTIDIMENSIONAL OPTIMIZATION FOR ENSEMBLE LEARNING IN CLOUD

User Requests

Dynamic Model Selection

Aggregator

Weight Matrix

output

Queries

Model-1

Model-2

Model-3

Model-4

Model-n

CPU

GPU

CPU

CPU

GPU

Prediction Policy

Importance Sampling

Autoscaler

Class-wise dictionary

Weighted Selection

Dedicated Pools

Per model Scaling

Fault tolerant
EVALUATION AND SETUP

Baselines
- InFaaS: Single Models
- Clipper: Static Ensemble
- Clipper-X: Dynamic ensemble

Workloads

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Application</th>
<th>Classes</th>
<th>Train-set</th>
<th>Test-set</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet [56]</td>
<td>Image</td>
<td>1000</td>
<td>1.2M</td>
<td>50K</td>
</tr>
<tr>
<td>CIFAR-100 [116]</td>
<td>Image</td>
<td>100</td>
<td>50K</td>
<td>10K</td>
</tr>
<tr>
<td>SST-2 [117]</td>
<td>Text</td>
<td>2</td>
<td>9.6K</td>
<td>1.8K</td>
</tr>
<tr>
<td>SemEval [118]</td>
<td>Text</td>
<td>3</td>
<td>50.3K</td>
<td>12.2K</td>
</tr>
</tbody>
</table>

Experiment Setup
- 40 EC2 CPU/GPU VMs
- Wiki Twitter Traces
MAJOR RESULTS

- Cocktail incurs ~32% lower cost
- Cocktail reduces #models by ~50% on average
- Cocktail yields ~2x lower latency
- Cocktail gains upto ~1.25% more accuracy
MAJOR RESULTS

- Cocktail quickly adapts the number of models.
- Cocktail on average uses 5 models.
- Cocktail incurs a modest accuracy loss of up to 0.7%.
- Cocktail avoids inference failures while not compromising accuracy.
Cocktail leverages ensembling to achieve higher accuracy at lower latency.

Cocktail dynamically adjusts the #models in the ensemble without compromising accuracy.

Cocktail leverages transient instances to reduce the deployment cost.
Thank You

Code: https://github.com/jashwantraj92/cocktail.git
Contact: jashwant.raj92@gmail.com, cyanmishra92@gmail.com