InFi: End-to-End Learnable Input Filter for Resource-Efficient Mobile-Centric Inference

Mu Yuan¹, Lan Zhang¹, Fengxiang He², Xueding Tong¹, Xiang-Yang Li¹

¹University of Science and Technology of China
²JD Explore Academy
Outline

1. Introduction
2. Framework and Analysis
3. Design and Implementation
4. Evaluation
1.1 Mobile-Centric Inference

- Mobile devices’ increasing computing power

![Graph showing Geekbench score comparison between devices]

https://browser.geekbench.com/mobile-benchmarks
1.1 Mobile-Centric Inference

• Mobile devices’ increasing computing power
• Growing demand for real-time sensor data analytics
  • Over 80% of enterprise IoT projects will incorporate AI by 2022

https://www.visualcapitalist.com/aiot-when-ai-meets-iot-technology/
1.1 Mobile-Centric Inference

• Mobile devices’ increasing computing power
• Growing demand for real-time sensor data analytics
  • Over **80%** of enterprise IoT projects will incorporate AI by 2022
    https://www.visualcapitalist.com/aiot-when-ai-meets-iot-technology/

• AI models with the state-of-the-art accuracy are too **computationally intensive**

1.1 Mobile-Centric Inference

• Mobile devices’ increasing computing power
• Growing demand for real-time sensor data analytics
  • Over 80% of enterprise IoT projects will incorporate AI by 2022
    https://www.visualcapitalist.com/aiot-when-ai-meets-iot-technology/
• AI models with the state-of-the-art accuracy are too **computationally intensive**
• **Resource-efficiency** is important for mobile-centric model **inference** workloads

Model Inference: the process of running pre-trained AI models on new inputs
1.2 Input Redundancy

• Widespread redundancy in inputs
  • Type#1: **SKIP** the inputs that do not return valuable results

![Diagram showing input types](image_url)
1.2 Input Redundancy

- Widespread redundancy in inputs
  - Type#1: **SKIP** the inputs that do not return valuable results
  - Type#2: **REUSE** the results that previously computed
1.3 Key Goals

• Robust feature discriminability

SOTA method does not work
1.3 Key Goals

• Robust feature discriminability
• Theoretical filterability for application guidance
  • Tailored solutions bring the cumbersome trial-and-error process, due to the lack of theoretical analysis
Outline

1. Introduction
2. Framework and Analysis
3. Design and Implementation
4. Evaluation
2.1 SKIP as REUSE

- SKIP workflow

\[ x \rightarrow \text{Binary Classification} \rightarrow \text{Skip or not?} \]
2.1 SKIP as REUSE

- SKIP workflow

  \[\text{X} \rightarrow \text{Binary Classification} \rightarrow \text{Skip or not?}\]

- REUSE workflow

  \[\text{X} \rightarrow \text{Similarity Measurement} \rightarrow \text{Hit or miss? If hit, reuse which result?}\]

  \text{Inp:Res Table}
2.1 SKIP as REUSE

- Unify SKIP and REUSE approaches:

  **SKIP equals to REUSE the NONE output of input** $\vec{0}$

\[ \text{Inp:Res Table} \rightarrow 0 : \text{NONE} \rightarrow 0 \]

$\rightarrow \text{Reuse result NONE = Skip}$

\[ \text{Similarity Measurement} \rightarrow x \]
2.2 End-to-End Learnability

- End-to-end learning casts complex processing components into **coherent connections** in neural networks and optimizes itself by applying back-propagation all through the networks.
  - **Robust feature discriminability!**

---

**Filtering Performance**

- **FD**
- **PE**
- **GC**
- **AC**
- **AD**
- **VC**
- **SR**
- **NER**
- **HAR**
- **UI**
- **SC**
- **VC-MP**

*Our method always works.*
2.2 End-to-End Learnability

- **Embedding-Difference-Classification Framework**
- Training Phase:

![Diagram showing the training phase of the Embedding-Difference-Classification Framework](Image)

**End-to-end learnable using metric learning paradigm**
2.2 End-to-End Learnability

- **Embedding-Difference-Classification Framework**

  - Training Phase:

    - **Feature Embedding**

  - Inference Phase:

    - **SKIP**: $x \rightarrow e \rightarrow d \rightarrow z$
    - **REUSE**: $x \rightarrow e \rightarrow d \rightarrow z$
    - **e**: y Table
1.3 Key Goals

- Robust feature discriminability
- Theoretical filterability for application guidance
2.3 Theoretical Analysis

• Learning problem formulation
• Filterability definition, based on hypothesis complexity comparison

Definition 2 (Filterability). Let $\text{Complex}(\cdot)$ denote the complexity measurement of a hypothesis family. We say that the inference workload is filterable, if $\text{Complex}(G) \leq \text{Complex}(H)$, where $h \in H$ and $(f_h \circ h) \in G$.

Intuition: If it is easier to learn the filter than to learn the inference model, the workload is \textit{FILTERABLE}. 
2.3 Theoretical Analysis

- Case 1: low-confidence classification as redundancy
  
  \[ \text{Lemma 4. Let } \mathcal{H} \text{ be a family of binary classifiers taking values in } \{-1, +1\}. \text{ For } \mathcal{G} = \{ \text{sign}(h(h + b)) \} \text{ where } h \in \mathcal{H}, b \in \mathbb{R}: \]

  \[ \hat{R}_S(\mathcal{G}) \leq \hat{R}_S(\mathcal{H}). \]  

  See our paper for detailed formulation, proof and analysis.

- Case 3: regression bound as redundancy
  
  \[ \text{Theorem 6. Let } p \geq 1 \text{ and } \mathcal{H} = \{ x \mapsto |h(x) - c(x)|^p : h \in \mathcal{H} \}. \text{ Assume that } |h(x) - c(x)| \leq M \text{ for all } x \in X \text{ and } h \in \mathcal{H}. \text{ Then the following inequality holds: } \hat{R}_S(\mathcal{H}) \leq pM^{p-1}\hat{R}_S(H). \]
Outline

1. Introduction
2. Framework and Analysis
3. Design and Implementation
4. Evaluation
3.1 Architecture

Modality feature networks

• 5 modalities
  • Text
  • Image
  • Video
  • Audio
  • Sensor signal and feature map (vector)
3.1 Architecture

Modality feature networks
• 5 modalities
  • Text
  • Image
  • Video
  • Audio
  • Sensor signal and feature map (vector)

Task-agnostic classifier
• Fully-connected neural networks

InFi: INput FIlttering

• Audio
• Sensor signal and feature map (vector)
3.2 Inference with InFi

• **SKIP**
  • Confidence threshold

• **REUSE**
  • Cache entry: input embedding – inference result
  • Homogeneity score-based cache miss detection


• K-Nearest Neighbors algorithm for retrieval
3.3 Deployments

- Case#1: on-device
- Case#2: offloading
- Case#3: model partitioning
3.4 Implementation

• Feature networks and classifiers are built with TensorFlow 2.4
• TFLite are used to transform saved checkpoints into Java servable object for Android deployment

• Opensource: https://github.com/yuanmu97/infi
Outline

1. Introduction
2. Framework and Analysis
3. Design and Implementation
4. Evaluation
4.1 Setup

- **12 Workloads**
- **5 datasets**
- **6 modalities**

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Modality</th>
<th>Inference Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hollywood2</td>
<td>Video Clip</td>
<td>Action Classification (AC)</td>
</tr>
<tr>
<td></td>
<td>Image</td>
<td>Face Detection (FD)</td>
</tr>
<tr>
<td></td>
<td>Audio</td>
<td>Pose Estimation (PE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Gender Classification (GC)</td>
</tr>
<tr>
<td></td>
<td>Text</td>
<td>Speech Recognition (SR)</td>
</tr>
<tr>
<td>ESC-10</td>
<td>Audio</td>
<td>Anomaly Detection (AD)</td>
</tr>
<tr>
<td>UCI HAR</td>
<td>Motion Signal</td>
<td>Activity Recognition (HAR)</td>
</tr>
<tr>
<td>MoCap</td>
<td>Motion Signal</td>
<td>User Identification (UI)</td>
</tr>
<tr>
<td>City Traffic</td>
<td>Video Stream</td>
<td>Vehicle Counting (VC)</td>
</tr>
<tr>
<td></td>
<td>Feature Map</td>
<td>Vehicle Counting (VC-MP)</td>
</tr>
</tbody>
</table>

48x10 hours of videos (1FPS) collected from 10 cameras at road intersections
4.1 Setup

• 4 devices
  • GPU server (one NVIDIA 2080Ti)
  • Development board (NVIDIA JETSON TX2)
  • Mobile phone (XIAOMI Mi 5)
  • Smartwatch (HUAWEI WATCH)

• 3 baselines
  • FilterForward \textit{MLSys} ‘19
  • Reducto \textit{SIGCOMM} ‘20
  • FoggyCache \textit{MobiCom} ‘18
4.2 Filtering Performance

- SKIP

Filtering Rate @ 90% Inference Accuracy

- FilterForward
- Reducto
- InFi-Skip
- Optimal
4.2 Filtering Performance

• REUSE

InFi is widely applicable to inference workloads. InFi outperforms state-of-the-art approaches on ALL tasks.
4.2 Filtering Performance

• Sensitivity to training size

**SKIP**

- 

**REUSE**

- 

*Legend*

- **Worst**
- **Optimal**
- **InFi (R=1)**
- **InFi (R=0.1)**
- **Random**
- **InFi (R=1)**
- **InFi (R=0.1)**
4.2 Filtering Performance

- Mobile-centric deployments
  - On-device, offloading, model partitioning
  - Vehicle counting workload

<table>
<thead>
<tr>
<th>Throughput (FPS) / Bandwidth Saving (%)</th>
<th>YOLOv3</th>
<th>YOLOv3 + InFi-Skip</th>
<th>YOLOv3 + InFi-Reuse</th>
<th>YOLOv3-tiny</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference Acc. (%)</td>
<td>100</td>
<td>90.3</td>
<td>90.5</td>
<td>67.9</td>
</tr>
<tr>
<td>On-device</td>
<td>3.2 / -</td>
<td>9.3 / -</td>
<td>27.2 / -</td>
<td>20.4 / -</td>
</tr>
<tr>
<td>Offloading</td>
<td>22.0 / -</td>
<td>55.2 / 66.5</td>
<td>77.2 / 91.1</td>
<td>225.3 / -</td>
</tr>
<tr>
<td>Model partitioning</td>
<td>24.5 / -</td>
<td>39.0 / 70.7</td>
<td>46.0 / 95.0</td>
<td>230.4 / -</td>
</tr>
</tbody>
</table>
4.2 Filtering Performance

- Mobile-centric deployments
  - On-device, offloading, model partitioning
- Pose estimation workload

<table>
<thead>
<tr>
<th>Throughput (FPS) / Bandwidth Saving (%)</th>
<th>OpenPose</th>
<th>OpenPose + InFi-Skip</th>
<th>OpenPose-light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference Acc. (%)</td>
<td>100</td>
<td><strong>90.1</strong></td>
<td>76.5</td>
</tr>
<tr>
<td>On-device</td>
<td>15.4 / -</td>
<td>18.0 / -</td>
<td>28.1 / -</td>
</tr>
<tr>
<td>Offloading</td>
<td>27.7 / -</td>
<td>31.5 / 18.9</td>
<td>98.5 / -</td>
</tr>
<tr>
<td>Model partitioning</td>
<td>29.2 / -</td>
<td>33.1 / <strong>20.2</strong></td>
<td>102.4 / -</td>
</tr>
</tbody>
</table>

*High throughput but low accuracy*

*Throughput boost, Bandwidth saved*

*Flexible resource-accuracy trade-off*
4.3 Filterability

- Filterable vs. non-filterable

*Filtering performance of FILTERABLE workloads is BETTER than NON-FILTERABLE ones.*
4.4 Overhead

• Latency and energy costs on mobile platforms

See our paper for more evaluations of feature discriminability, parameter sensitivity, temporal robustness, and so on.
Take-Home Message

• Much **redundancy** exists in the input of inference workloads.
• We show that some workloads proved to be **filterable**, while some are non-filterable.
• Our ![∞](∞) **InFi** supports **almost all** mobile-centric inference workloads. Try it :)
Thank You!

Opensource: https://github.com/yuanmu97/infi

InFi : End-to-End Learnable Input Filter for Resource-Efficient Mobile-Centric Inference

Mu Yuan¹, Lan Zhang¹, Fengxiang He², Xueling Tong¹, Xiang-Yang Li¹
¹University of Science and Technology of China ²JD Explore Academy