Mistify: Automating DNN Model Porting for On-Device Inference at the Edge

Peizhen Guo, Bo Hu, Wenjun Hu

Yale University
On-device deep learning inference
On-device deep learning inference

Need compact and accurate DNN models
Where do the models come from?
Where do the models come from?

Pre-trained model

TensorFlow Hub
PYTORCH HUB
Keras
GLUON
Where do the models come from?

Pre-trained model → **Tailor** to the deployment setting → Deployed model

- TensorFlow Hub
- PyTorch Hub
- Gluon
- Keras
Tons of DNN tailoring algorithms


ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware

AdaNet: Adaptive Structural Learning of Artificial Neural Networks

Once-For-All: Train One Network and Specialize It for Efficient Deployment

ChamNet: Towards Efficient Network Design through Platform-Aware Model Adaptation

Slimmable Neural Networks

Many others......
However, tailoring a DNN is still not trivial!
However, tailoring a DNN is still not trivial!

**Edge deployment goal:**
- 10 GFLOPs
- 300 MB size

**Implement the tailoring algorithm**

**Laborious manual efforts**
- Annotate the DNNs
- Configure parameters
However, tailoring a DNN is still not trivial!

**Edge deployment goal:**
- 10 GFLOPs
- 300 MB size

**Original model**

**Expensive search process**
- Hundreds of GPU hours
- 10x Gig memory usage

**Tailored model**

1. Implement the tailoring algorithm
2. Execute the DNN adaptation

**Laborious manual efforts**
- Annotate the DNNs
- Configure parameters
Even worse in practice -
Heterogeneous hardware targets
Even worse in practice -
Heterogeneous performance requirements

Autonomous driving

Traffic monitoring

Google Lens

Performance requirements

~1ms  ~30ms  ~1s
Even worse in practice - Model Diversity
Even worse in practice - Model Diversity

Even worse in practice -
Huge tailoring space

DNN model complexity

Hardware / Performance requirements

Huge space for tailoring
Even worse in practice -
Huge tailoring space
Even worse in practice - Runtime dynamics

App requirement dynamics
- Accuracy (critical vs. idle)
- Latency (day vs. night)
- Power (battery vs. charged)
- ...

Device resource dynamics
- Memory space
- CPU quota
- Accelerator availability
- Queuing time
Summary: practical challenges

- Unscalable DNN tailoring needs
- Runtime dynamics
Summary: practical challenges

- Unscalable DNN tailoring needs
- Runtime dynamics
  
Need *system support*
Existing DL ecosystems

Data: TF Dataset

Model: Keras

Training: TF.Distribute.Strategy

Model repo: TF Hub

Analysis: TensorBoard

On-device inference: TensorFlow Lite

Training (design)

Inference (deployment)
Current DNN tailoring practice

Data: TF Dataset
Model: Keras
Training: TF.Distribute.Strategy
Model repo: TF Hub
Analysis: TensorBoard

On-device inference: TensorFlow Lite

Current tailoring practice spills into both sides and relies on human.

Training (design)  Inference (deployment)
Ideal DNN tailoring practice

Data: TF Dataset
Model: Keras
Training: TF.Distribute.Strategy
Model repo: TF Hub
Analysis: TensorBoard

Training (design)

Inference (deployment)

Need system support in between to handle the complexity
On-device inference: TensorFlow Lite
Our solution - *Mistify*

- *Mistify* – framework for automated DNN model porting
- Decoupling and bridging DNN design and deployment
- Reducing manual efforts and computation overhead
Mistify design
How Mistify addresses the challenges

• Unscalable DNN tailoring needs
  • Adaptation executor abstraction
  • Collective adaptation

• Runtime dynamics
  • Switching on multi-branch models
  • Model re-adaptation
Adaptation executor

Diagram:
- Start with **Init**
- Move to **Search**
- Then to **Measure**
- If **satisfied**, go to **END**
- If **not satisfied**, go back to **Adjust**
- **Adjust** connects back to **Search**
Adaptation executor

Init
Embed adaptation logic, configure execution parameters, etc.

Search

Measure

Adjust
Adaptation executor

- **Init**: Embed adaptation logic, configure execution parameters, etc.
- **Search**: Core state: Run the actual DNN structure searching process for ~ iterations
- **Measure**
- **Adjust**
Adaptation executor

**Init**
Embed adaptation logic, configure execution parameters, etc.

**Search**
Core state: Run the actual DNN structure searching process for ~ iterations

**Measure**
Measuring the cost of the current DNN, and decide if ready to terminate.

**Adjust**
**Adaptation executor**

- **Init**: Embed adaptation logic, configure execution parameters, etc.
- **Search**:
  - Core state: Run the actual DNN structure searching process for ~ iterations
  - Adjust the parameters to control the searching algorithm behaviors
- **Measure**: Measuring the cost of the current DNN, and decide if ready to terminate.
How *Mistify* addresses the challenges

- **Unscalable DNN tailoring needs**
  - Adaptation executor abstraction – *minimizes manual efforts*
  - Collective adaptation

- **Runtime dynamics**
  - Switching on multi-branch models
  - Model re-adaptation
Multiple adaptation goals

How to scale to a batch of simultaneous adaptations?
Multiple adaptation goals

X
Original DNN

G1
#Params: 12M
#FLOPs: 7G

G2
#Params: 10M
#FLOPs: 6G

G3
#Params: 7M
#FLOPs: 5G

G4
#Params: 9M
#FLOPs: 2G

G5
#Params: 5M
#FLOPs: 4G

G6
#Params: 6M
#FLOPs: 3G

G7
#Params: 8M
#FLOPs: 1G
Adapt independently

7 adaptation processes, inefficient!
Adapt independently

Repeated efforts!
Adapt independently

- G1: #Params: 12M, #FLOPs: 7G
- G2: #Params: 10M, #FLOPs: 6G
- G3: #Params: 7M, #FLOPs: 5G
- G4: #Params: 9M, #FLOPs: 2G
- G5: #Params: 5M, #FLOPs: 4G
- G6: #Params: 6M, #FLOPs: 3G
- G7: #Params: 8M, #FLOPs: 1G
Instead, collective adaptation
Instead, collective adaptation

Only 3 adaptation processes are needed
How *Mistify* addresses the challenges

- **Unscalable DNN tailoring needs**
  - Adaptation executor abstraction – *simplify manual efforts*
  - Collective adaptation – *scale with multiple adaptation processes*

- **Runtime dynamics**
  - Switching on multi-branch models
  - Model re-adaptation
How to handle runtime dynamics?

Foreground: switching on multi-branch DNNs

Background: on-demand model re-adaptation
Generate multi-branch DNNs

Single-branch model

Branch1: 
#Params: 12M 
#FLOPs: 7G

Branch2 
#Params: 10M 
#FLOPs: 6G

Branch3 
#Params: 7M 
#FLOPs: 5G
Generate multi-branch DNNs

Single-branch model

Now I need the DNN with 8M params and 5G FLOPs
Background: re-adaptation

- **G8**: 
  - #Params: 6.5M
  - #FLOPs: 3.5G

- **G1**: 
  - #Params: 12M
  - #FLOPs: 7G

- **G2**: 
  - #Params: 10M
  - #FLOPs: 6G

- **G3**: 
  - #Params: 7M
  - #FLOPs: 5G

- **G4**: 
  - #Params: 9M
  - #FLOPs: 2G

- **G5**: 
  - #Params: 5M
  - #FLOPs: 4G

- **G6**: 
  - #Params: 6M
  - #FLOPs: 3G

- **G7**: 
  - #Params: 8M
  - #FLOPs: 1G
Background: re-adaptation

- **G1**: #Params: 12M, #FLOPs: 7G
- **G2**: #Params: 10M, #FLOPs: 6G
- **G3**: #Params: 7M, #FLOPs: 5G
- **G4**: #Params: 9M, #FLOPs: 2G
- **G5**: #Params: 5M, #FLOPs: 4G
- **G6**: #Params: 6M, #FLOPs: 3G
- **G7**: #Params: 8M, #FLOPs: 1G
- **G8**: #Params: 6.5M, #FLOPs: 3.5G
How *Mistify* addresses the challenges

- **Unscalable DNN tailoring needs**
  - Adaptation executor abstraction – *simplify manual efforts*
  - Collective adaptation – *scale with multiple adaptation processes*

- **Runtime dynamics**
  - Switching on multi-branch models
  - Model re-adaptation
Mistify system workflow

Original DNN model

On-device DL Inference engine
Mistify system workflow

- Original DNN model
- Adaptation configs

On-device DL Inference engine
**Mistify system workflow**

- **Mistify server**
  - Collective adaptation
    - Parse configs
    - Adaptation executor

- **Parse adaptation requests and generate collective adaptation schedules**

- **On-device DL Inference engine**
Mistify system workflow

Collective adaptation

Parse configs

Adaptation executor

Mistify server

Mistify client

On-device DL Inference engine
**Mistify system workflow**

- **Mistify server**
  - Collective adaptation
  - Adaptation executor
  - Parse configs

- **Performance Monitor**
- **On-device DL Inference**

**Actions**
- **Trigger runtime re-adaptation**
- **Switch model branches**

---

**50**

On-device DL Inference

Collective adaptation

Adaptation executor

Parse configs

Performance Monitor
Mistify performance
## General setup

### Server
- Linux server with RTX 2070 GPU

### Devices
- Samsung S9
- Google Edge TPU
- Nvidia P600 GPU

### Models
- CV: MobileNet, ResNet50, ResNeXt101
- NLP: BiDAF, BERT

### Workloads & datasets
- Image classification (ImageNet, Cifar100)
- Question & Answering (SQuADv1.1)
Scalability

![Graph showing scalability with speedup ratio vs. number of configurations. The ideal speedup ratio is indicated.]
Scalability

![Graph showing scalability with ideal speedup ratio](image)

- MobileNet
- ResNet50
- BERT

Ideal speedup ratio: 10~70x
Minimizing manual efforts

<table>
<thead>
<tr>
<th>Metrics</th>
<th>10 configurations</th>
<th>100 configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>MorphNet</td>
<td><strong>Mistify</strong></td>
</tr>
<tr>
<td>Manual</td>
<td>MorphNet</td>
<td><strong>Mistify</strong></td>
</tr>
</tbody>
</table>

- Implements and executes the algorithm from scratch
- Adapts to each configuration individually
Minimizing manual efforts

<table>
<thead>
<tr>
<th>Metrics</th>
<th>10 configurations</th>
<th>100 configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual</td>
<td>MorphNet</td>
</tr>
</tbody>
</table>

- **Annotates** adaptation logic and termination conditions
- Adapts to each configuration **individually**
## Minimizing manual efforts

<table>
<thead>
<tr>
<th>Metrics</th>
<th>10 configurations</th>
<th>100 configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual</td>
<td>MorphNet</td>
</tr>
</tbody>
</table>

- Fully automated
- Adapt to multiple configurations **collectively**
Minimizing manual efforts

<table>
<thead>
<tr>
<th>Metrics</th>
<th>10 configurations</th>
<th>100 configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Manual</td>
<td>MorphNet</td>
</tr>
<tr>
<td>Lines of Code</td>
<td>&gt;1k</td>
<td>138</td>
</tr>
<tr>
<td>Num of Files</td>
<td>30</td>
<td>12</td>
</tr>
</tbody>
</table>

- **Orders of magnitude** fewer lines of code changed
- **Constant** number of files changed
Minimizing manual efforts

<table>
<thead>
<tr>
<th>Metrics</th>
<th>10 configurations</th>
<th>100 configurations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code</td>
<td>&gt;1k</td>
<td>138</td>
</tr>
<tr>
<td>Num of Files</td>
<td>30</td>
<td>12</td>
</tr>
<tr>
<td>Time (normalized)</td>
<td>10</td>
<td><strong>1.25</strong></td>
</tr>
</tbody>
</table>

**Time:** from linear to *nearly constant*
Conclusion

Mistify – automated and scalable DNN porting service

Decoupling DNN design and deployment and bridging them with an end-to-end framework

Orders of magnitude reduction of computation overhead and manual efforts
Thank you