Dynamic Distributed Computing for Autonomous Vehicles in 5G Infrastructures

Marco Levorato
Professor
CS - University of California, Irvine
levorato@uci.edu

Earth Day Workshop 2024
Marco Levorato
Professor
University of California, Irvine
Computer Science Dept.

Faculty - Computer Science
University of California, Irvine

PostDoc - Electrical Engineering
Stanford University
University of Southern California

Ph.D - Telecommunication Engineering
University’ degli Studi di Padova (Italy)
Dynamic, Distributed and Resilient Computing for Extreme Mobile Applications

Autonomous cars

Autonomous drones

Collaborative robots

Smart manufacturing

Augmented Reality

Mobile HealthCare

Funded by

NSF
DARPA
MTI
Intel
Cisco
Dynamic, Distributed and Resilient Computing for Extreme Mobile Applications

**Challenge**

Mismatch between **computing needs** (e.g., continuous stream of complex computing tasks), **requirements** (accuracy, latency) and **resource** availability (energy, computing power)
Multi-Faceted Challenge

**Algorithm level**
Split, dynamic, multi-branched models designed for efficiency and distributed execution

**Device-level**
Context-aware computing strategies and semantic communication strategies

**Infrastructure-level**
AI for semantic network and edge slicing, generalization and knowledge accumulation

---

**Original Model**

**Modified Model**

**Head Model (Mobile Device)**

**Tail Model (Edge Device)**

**Splitting Point**

**Wireless Channel**

**Inference Output**

**Edge**

**Compressed Representation**

**Client UAV**

**Server UAV**

**Sensor (Task Generation)**

**Analysis**

**Edge Server 1**

**Edge Server 2**

**AR**

**Robots**

**Vehicles**

**Computer Vision**

**NLP**

**Decision Making**

**Local Platform**

**Teacher Block 1**

**Teacher Block 2**

**Teacher Block n**

**Search Block 2**

**RX of STA 1**

**RX of STA 2**

**RX of STA 3**

**Selector**

**Logging Agent**

---

**Offloading point is placed following the execution path for both the processing pipeline with an early-exit model as an additional – within the network to realize an early optimal offloading point –...**

**Modular approach, namely...**

**Blockwise neural architecture search (NAS) whose advantages...**

**For energy efficiency, we propose to supplement the pro-...**

**A predetermined threshold...**

**Modeling fail-safe offloading:**

**Since perception constitutes the bulk of the processing load...**

**performing industry-grade processing pipelines for AS [3],...**

**As how the outputs from perception and localization...**

**To perceive events occurring in the environment, sensing...**

**DNN model design through identifying architecture...**

**Then, we can estimate the maximum allowable cumulative...**

**In terms of the...**

**Min...**

**Loss...**

**(2) subject to a...**

**is performed locally or remotely, each...**

**language:**

**and...**

**and...**

**in which...**

**is derived from the channel frequency response (CFR) matrix...**

**is transmitted from...**

**The Access Point (AP)–integrated with...**

**is normalized relative to the cor-...**

**that...**

**is a different type of the signal are...**

**to allow transmission in...**

**Signal is transmitted from...**

**to the...**

**the...**

**in which...**

**different copies of the signal are...**

**From the...**

**can differ from one candidate partial model to the other within...**

**implementing industry-grade processing pipelines for AS [3],...**

**where...**

**implement the early exit using a modular approach, namely...**

**proposed to divide the search space...**

**the task at hand, allowing optimizations to be targeted towards...**

**blockwise neural architecture search (NAS) whose advantages...**

**through identifying architecture...**

**Then, we can estimate the maximum allowable cumulative...**

**for the stage and the PM's ranking based on the loss defined in (12).**

---

**Head Model**

**Encoder**

**Decoder**

**Modified Model**

**Head Model**

**Tail Model**

**Inference**

**Wireless Channel**

**Client UAV**

**State**

**Application**

**Network**

**Telemetry**

**Analysis**

**Sensor (Task Generation)**

**Edge Server 1**

**Edge Server 2**

**AR**

**Robots**

**Vehicles**

**Computer Vision**

**NLP**

**Decision Making**

**Local Platform**

**Teacher Block 1**

**Teacher Block 2**

**Teacher Block n**

**Search Block 2**

**RX of STA 1**

**RX of STA 2**

**RX of STA 3**

**Selector**

**Logging Agent**

---

**Splitting Block**

**Bottleneck Splitting Point**

**High Resolution Image**

**Head Model**

**Edge**

**Compressed Representation**

**Client UAV**

**State**

**Application**

**Network**

**Telemetry**

**Analysis**

**Sensor (Task Generation)**

**Edge Server 1**

**Edge Server 2**

**AR**

**Robots**

**Vehicles**

**Computer Vision**

**NLP**

**Decision Making**

**Local Platform**

**Teacher Block 1**

**Teacher Block 2**

**Teacher Block n**

**Search Block 2**

**RX of STA 1**

**RX of STA 2**

**RX of STA 3**

**Selector**

**Logging Agent**

---

**Original Model**

**Modified Model**

**Head Model (Mobile Device)**

**Tail Model (Edge Device)**

**Splitting Point**

**Wireless Channel**

**Inference Output**

**Edge**

**Compressed Representation**

**Client UAV**

**State**

**Application**

**Network**

**Telemetry**

**Analysis**

**Sensor (Task Generation)**

**Edge Server 1**

**Edge Server 2**

**AR**

**Robots**

**Vehicles**

**Computer Vision**

**NLP**

**Decision Making**

**Local Platform**

**Teacher Block 1**

**Teacher Block 2**

**Teacher Block n**

**Search Block 2**

**RX of STA 1**

**RX of STA 2**

**RX of STA 3**

**Selector**

**Logging Agent**

---

**Original Model**

**Modified Model**

**Head Model (Mobile Device)**

**Tail Model (Edge Device)**

**Splitting Point**

**Wireless Channel**

**Inference Output**

**Edge**

**Compressed Representation**

**Client UAV**

**State**

**Application**

**Network**

**Telemetry**

**Analysis**

**Sensor (Task Generation)**

**Edge Server 1**

**Edge Server 2**

**AR**

**Robots**

**Vehicles**

**Computer Vision**

**NLP**

**Decision Making**

**Local Platform**

**Teacher Block 1**

**Teacher Block 2**

**Teacher Block n**

**Search Block 2**

**RX of STA 1**

**RX of STA 2**

**RX of STA 3**

**Selector**

**Logging Agent**

---

**Original Model**

**Modified Model**

**Head Model (Mobile Device)**

**Tail Model (Edge Device)**

**Splitting Point**

**Wireless Channel**

**Inference Output**

**Edge**

**Compressed Representation**

**Client UAV**

**State**

**Application**

**Network**

**Telemetry**

**Analysis**

**Sensor (Task Generation)**

**Edge Server 1**

**Edge Server 2**

**AR**

**Robots**

**Vehicles**

**Computer Vision**

**NLP**

**Decision Making**

**Local Platform**

**Teacher Block 1**

**Teacher Block 2**

**Teacher Block n**

**Search Block 2**

**RX of STA 1**

**RX of STA 2**

**RX of STA 3**

**Selector**

**Logging Agent**

---

**Original Model**

**Modified Model**

**Head Model (Mobile Device)**

**Tail Model (Edge Device)**

**Splitting Point**

**Wireless Channel**

**Inference Output**

**Edge**

**Compressed Representation**

**Client UAV**

**State**

**Application**

**Network**

**Telemetry**

**Analysis**

**Sensor (Task Generation)**

**Edge Server 1**

**Edge Server 2**

**AR**

**Robots**

**Vehicles**

**Computer Vision**

**NLP**

**Decision Making**

**Local Platform**

**Teacher Block 1**

**Teacher Block 2**

**Teacher Block n**

**Search Block 2**

**RX of STA 1**

**RX of STA 2**

**RX of STA 3**

**Selector**

**Logging Agent**
Autonomy
Autonomy Stack

Perception
- Object Detection
- Semantic Segmentation
- Image Classification

Reasoning
- Semantic Extraction
- Motion Prediction
- Mapping and Localization

Control
- Path Planning
- Vehicle Control

Concatenation of complex Modules (often neural networks)
Autonomous cars and other large vehicles can support the stack, at the price of an increased cost and power expense.
Autonomous cars and other large vehicles can support the stack, at the price of an increased cost and power expense.
What if we want to empower small scale vehicles with advanced autonomy functionalities?

Defense

Logistics

Delivery
Or even just advanced computer vision…

Traffic/City Monitoring

Low-orbit monitoring
Objectives and Approach
Dynamic-Distributed AI pipelines that adapt to data and system context
Objectives and Approach
Dynamic-Distributed Composable AI pipelines that adapt to data and system context
Objectives and Approach

Dynamic-Distributed Composable AI pipelines that adapt to data and system context
Hydra - Testbed

• Middleware for AI-controlled dynamic edge computing
• Airborne and ground autonomous vehicles connected to multiple edge servers
• Real-time telemetry, network and application logging for decision making
• Control of how many and which edge servers
Advanced comms

- MU-MIMO comms
- Semantic control of comms parameters (bandwidth, number of antennas, connectivity) and packetization (replication vs parallelization)

Indoor-Outdoor Navigation

- Dynamic reconfiguration of autonomy stack across “layers”
- Dynamic neural sensor fusion
Preliminaries
Stream of data from multiple sensors to be analyzed to support mission functions and autonomous navigation.
“Compressed” models deployed on the vehicle/robot
Quantization, distillation, pruning
Onboard Computing

- Hardware cost
Onboard Computing

- **Hardware cost**
- **Task performance**
  - Model compression
  - Tradeoff between MAP/latency
Onboard Computing

- **Hardware cost**
- **Task performance**
  - Model compression
  - Tradeoff between MAP/latency
- **Energy**
Onboard Computing

- Hardware cost
- Task performance
  - Model compression
  - Tradeoff between MAP/latency
- Energy
- Hardware Degradation
Onboard Computing

- **Hardware cost**
- **Task performance**
  - Model compression
  - Tradeoff between MAP/latency
- **Energy**
- **Hardware Degradation**
- **Limited functionalities**
Data sent to a compute-capable device taking over the tasks
• Latency/Latency variance
Edge Computing

[Diagram showing the flow from Mobile Device to Edge Server through Wireless Channel]

[Graph showing losses over time with different settings]
Edge Computing

- Latency/Latency variance
- Uncertainty
Edge Computing

- Latency/Latency variance
- Uncertainty
- Bandwidth usage
  - Sharing with other users and services
Edge Computing

- Latency/Latency variance
- Uncertainty
- Bandwidth usage
  - Sharing with other users and services
- Hardware degradation
  - Servers are more resilient
Distributed AI
**JPEG**
- Low complexity
- Bad rate-distortion curve (high compression gain)
- Designed for human perception
Neural Encoders
• High complexity (mobile device and server)
• High performance
• Designed to reconstruct the input image
Split Deep Neural Networks

Trivial Split DNN
- Distribution of computing load
- Compression only if split point is toward the end of the model
- Optimal latency often at extreme point
Split Deep Neural Networks

“Artificial” Bottleneck
- Architecture altered to incorporate a bottleneck (in-model compression)
- Objective: minimal complexity - maximum compression - maximum task performance
- Specialized training
Supervised Compression

Input Image \( x \)

Bottleneck \( h_{k^*} = o_{k^*} \)

Encoder \( f_{enc}(x) \)

Decoder \( f_{dec}(h_{k^*}) \)

Classifier \((k^* + d)^{th} - n^{th}\) layers

Head Model \( \mathcal{H} \)

Tail Model \( \mathcal{T} \)

Prediction “bird”

- Encoder/decoder-like structure within the model
- Semantic in-model compression obtained at the splitting point
Multistage Training

- Encoder-decoder trained to reproduce an intermediate layer of the original model
- Supervised task-oriented compression: representation if trained to shed irrelevant bits
Performance

Rate-Distortion-Complexity Curve
Bit Allocation

Visualization: bit allocation with respect to a variational autoencoder

Input images

SC vs. IC
Multi-Branched Split Architecture
Local and Edge pipelines dynamically selected based on sample and system parameters

- Computing path dynamically controlled by a lightweight AI agent
- Predictive logics based on DRL
### Dynamic Neural Network for Autonomous Vehicles

#### Blockwise NAS

- **Teacher Block 1**: Bottleneck in the first student block
- **Search Block 2**: Input for search block 2
- **Teacher Block n**: Knowledge Distillation for loss estimation

#### Traversal Search

<table>
<thead>
<tr>
<th>Round 1</th>
<th>PM11</th>
<th>PM12</th>
<th>PM13</th>
<th>PM14</th>
<th>C₁</th>
<th>C₂</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) PM13 is the first to satisfy C₁</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Round 2</th>
<th>PM11</th>
<th>PM12</th>
<th>PM13</th>
<th>PM14</th>
<th>C₁</th>
<th>C₂</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>4) Still exploring models derived from PM13</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

- **PM21, PM22, PM23, PM24**
- **PM12, PM22, PM23**
- **PMn1, PMn2, PMn3, PMn4**

#### Knowledge Distillation

- **Loss**

#### Architectural Benefits

- Architecture automatically generated using Neural Architecture Search based on system parameters
- Optimized position and shape of the bottleneck
Split Self-Adaptive AI for Navigation
Navigation problem: nano/microdrone autonomously determines path to reach point B from point A in an unknown environment
Input: (a) GPS, (b) RGB image (no depth)
Two neural models:
- **Depth estimation**: neural model transforms the RGB image into a depth map
- **Navigation**: neural model transforms the depth map into motion commands
Supervised training based on knowledge distillation to **split** the depth estimation model.
Our solution:

- **Split depth estimation**: depth model is split to minimize data transmission and computing effort at the drone
- **Navigation**: neural model transforms the depth map into motion commands
- **Adaptation**: auxiliary neural model takes mission parameters and depth map as input to determine the optimal representation
Performance

Mean absolute percent error vs data size

Fig. 4. Comparing the compressed size of the data at the split point against depth mean absolute percent error (MAPE). Each marker (from left to right) corresponds to a number of channels at the split point equal to: 2, 4, 8, 16, and 32, respectively.

We then compare: the baseline student models, the bottleneck student models, and edge computing that completely offloads the RGB image by first running a JPEG compression – where we experiment with various qualities of compression between 5 and 95. See Figure 5. We see that the baseline models perform with equal memory consumption as both the bottleneck and JPEG models, however does not result in a lower error than JPEG. Alternatively, the bottleneck models perform with better error than that of the JPEG models - showing that the bottleneck methodology is more robust.

We then select the model with lowest testing error for each student model configuration. Figure 3 illustrates how each of the baseline student models compare in depth error to each other and to the teacher model. It is clear that the largest student model, with 32 channels, either has lower or equal error to the teacher at every distance. This illustrates the robustness of the split computing training methods. The 8-channel student model at some point switches between higher and lower error than the teacher model, and always receives higher error than the 32-channel student model. The 2-channel student model receives lower error than all other models when inferring depths below 10 meters, but otherwise receives the highest error for all other depths. Thus there is a subtle gradient in error between the models, for which we aim for the auxiliary model to select from given the context.

C. Navigation Model

The three baseline student models illustrated in Figure 3 are each used to infer depth maps to be sent downstream to a navigation model. The navigation model is trained and initially evaluated in the Blocks map. Each branch is trained independently, with no knowledge of the others. Figure 6 shows the number of evaluation paths that successfully reached the target location versus initial starting distance to target, where we can see that the paths struggle the most in mid range (50-100 meters) and each student model varies in performance. Figure 7 shows the different path lengths found from each model, of which we see what appears to be a linear relationship that is independent of student model. Figure 8 shows the total size of encoded data for each model throughout the path, where we see the aggregating effects of the larger models – especially Student-32. Also shown in these figures are that of the auxiliary model, which we discuss next.
Performance

AirSimNH Map

Gate Control C

x-position [meters]

y-position [meters]

Adaptation
Self-Adaptive Low-Complexity AI
Slimmable Neural Networks

Networks whose width can be reduced at runtime

\[
\frac{\partial}{\partial \theta} \left[ \frac{1}{2} (y - \hat{y})^2 \right]
\]

Knowledge distillation:
sub networks learn to mimic the super network
Dynamic Neural Navigation for Microdrones

New architecture realizes a **gated dynamic slimmable network** for navigation

Auxiliary neural gate controls the slices of a main navigation model decision by decision
- Number of operations
- Sensor selection and resolution

Specialized multi-stage training uses
- Knowledge distillation
- Curriculum learning
- Deep reinforcement learning
and downward facing depth sensor, respectively. Table I lists

Interestingly, as the values returned from the forward depth

sensor observations – which is most intuitive. 

depth sensor observations, but has a clear dependence on the

horizontal motion, the downward facing depth sensor is almost

for any navigation networks larger than the one considered.

amount of time needed to train the auxiliary network. We

learned values of 

increases with size of the navigation network. Note that the

navigation error, by using

with larger neural networks, which is needed to achieve lower

is characterized by a positive speedup and also overlaps with

larger networks result in decreased run times - as indicated by

Fig. 12 shows that smaller networks can actually result in

increased execution time when using

size – while using a fixed auxiliary network that has 2 hidden layers with

between using and not using

Fig. 12. Test set results ran on a Jetson Nano to measure the relative speedups

for the forward facing depth sensor or

auxiliary hidden layers was fixed at [32, 32], while the size of

NaviSlim

• Complexity slimming factors

• Sensing slimming factors
Dynamic Slimmable Networks

- Runtime adaptation
- Context AND system-aware
- Designed to be distributed (slimmable encoders are a component of it)
Vision
Layered Collaboration
FlexAI - Lightweight Swarm Intelligence

Wireless Links

Lightweight Adaptive Compression and Fusion

AI-Agent

3D Mapping and Localization
Object Detection
Semantic Segmentation

Camera Input

Lightweight Adaptive Compression and Fusion

AI-Agent

Lightweight Adaptive Compression and Fusion

AI-Agent

Lightweight Adaptive Compression and Fusion

AI-Agent

Camera Input

LiDAR Input

Wireless Links

Adaptive Decoder and Multi-Task Models

AI-Agent

AI-Agent

AI-Agent
Dynamic Distributed Computing for Autonomous Vehicles in 5G Infrastructures

Marco Levorato
Professor
CS - University of California, Irvine
levorato@uci.edu