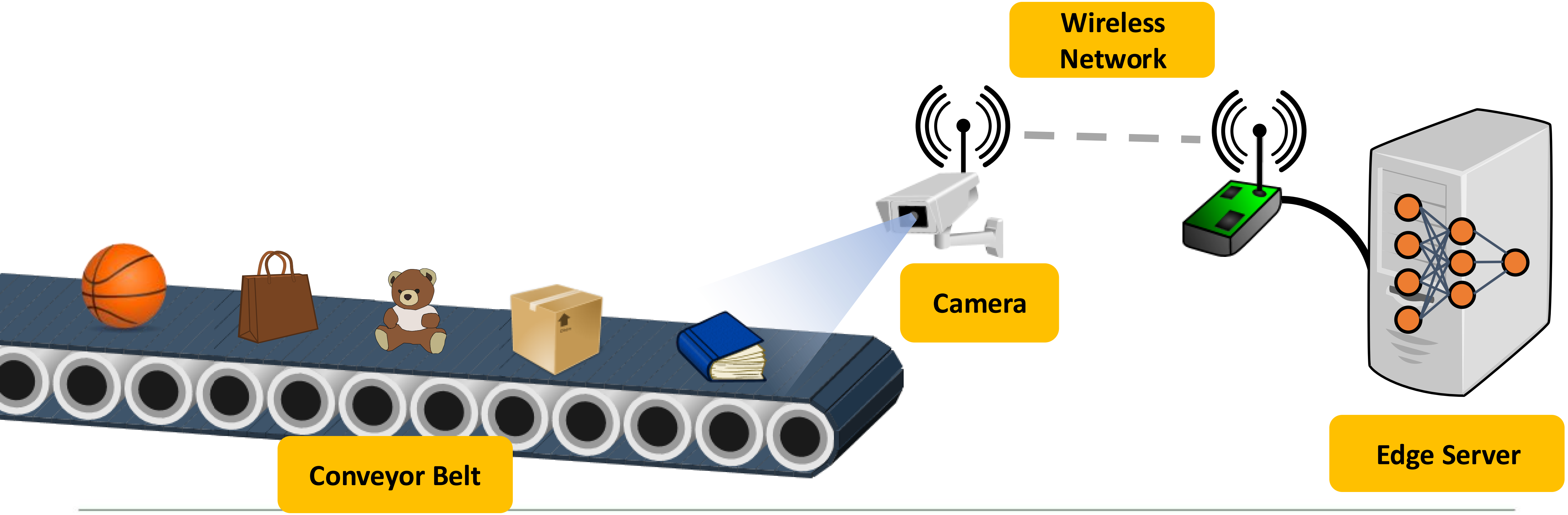
Progressive Neural Compression for Adaptive Image Offloading under Timing Constraints

Ruiqi Wang, Hanyang Liu, Jiaming Qiu, Moran Xu, Roch Guérin, Chenyang Lu
Department of Computer Science & Engineering

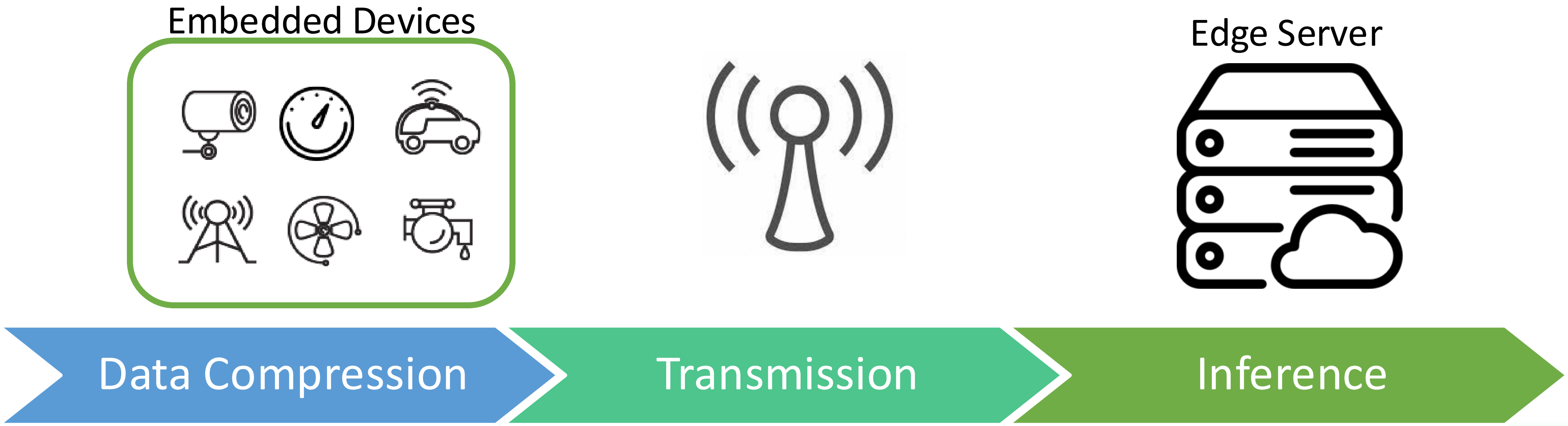
Real-Time Image Classification

- A camera captures, compresses, and offloads the image to an edge server.
- Edge server decodes and classifies the image.
- **Deadline** for offloading the image: the arrival of the next object.



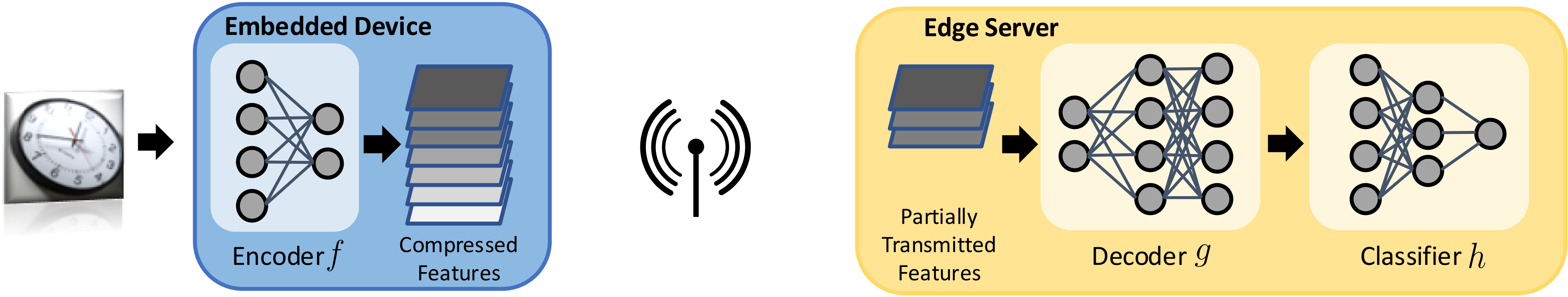
Progressive Compression

- Maximize **classification accuracy** under **varying bandwidth and deadlines**
  - Varying amount of data received by the deadline
- **Progressive** compression
  - Classification can be performed at any time on partially received data
  - Fewer data → graceful degradation in classification accuracy
  - More data → more accurate classification



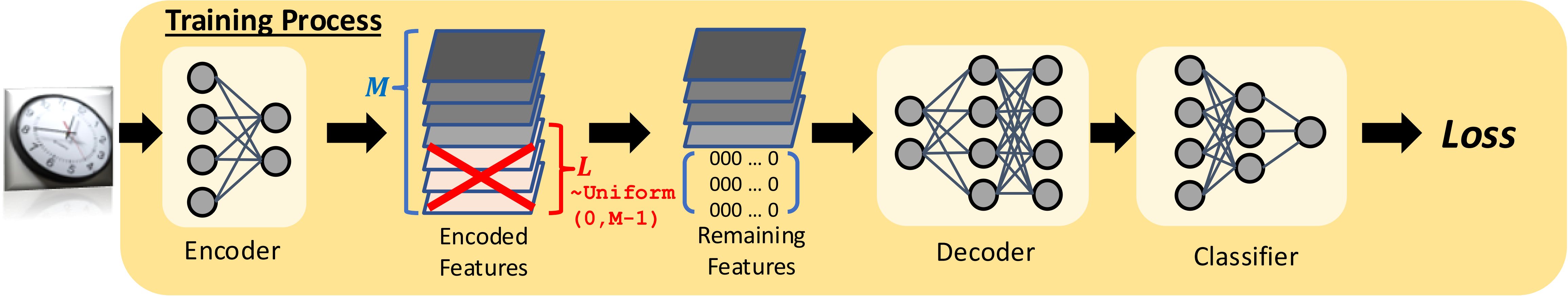
Neural Compression with Autoencoder

- The device compresses an image into features using an encoder network.
- The server reconstructs the image using a decoder network.
- Need to **make an autoencoder progressive.**
  - Maximize image classification accuracy with **partially** received features.



Make Autoencoder Progressive

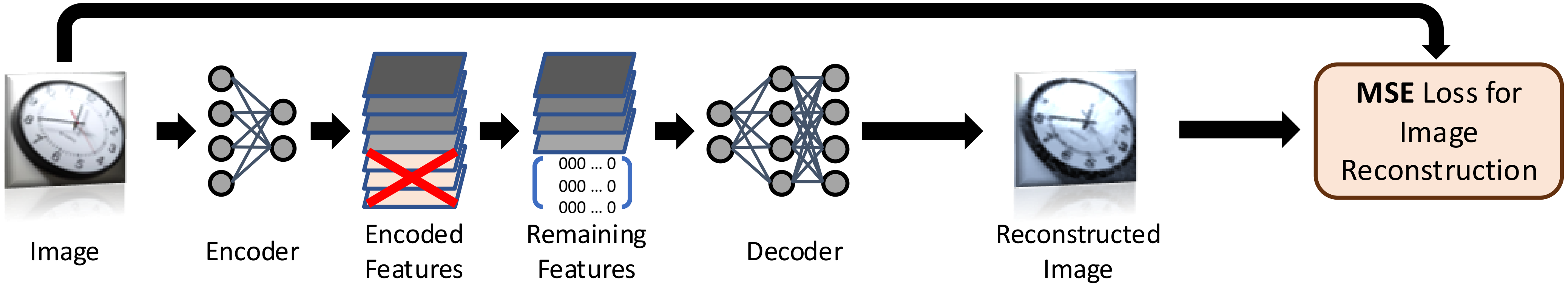
- "Stochastic tail-drop" (Koike-Akino et al., 2020)
  - During each training iteration: **Randomly** zero-out the last ***L*** **features** (**out of** ***M***).
  - Optimize the training loss as normal.



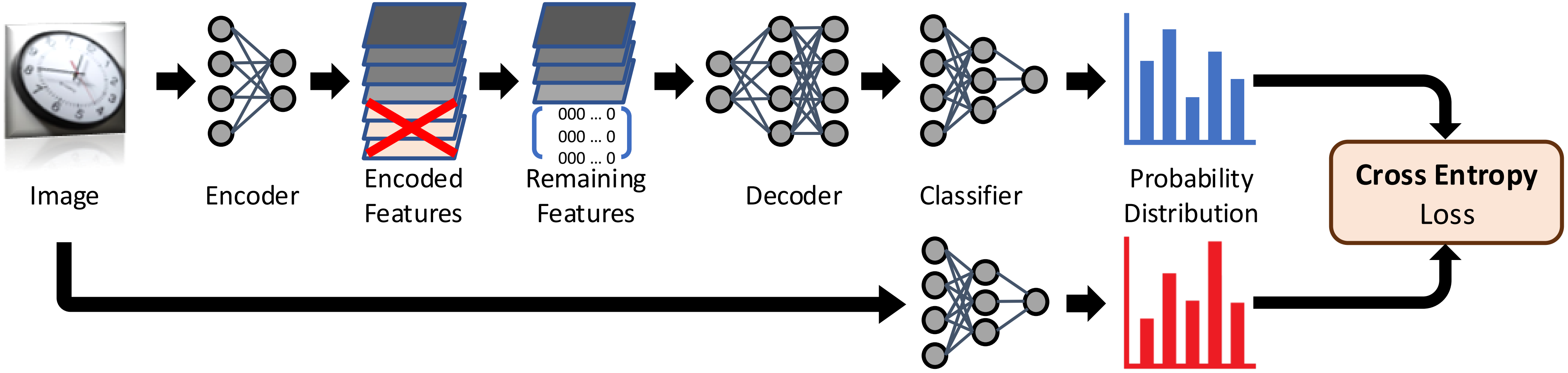
- By applying various tail-drop length in different iterations:
  - The decoder is trained to deal with **incomplete set of features**.
  - Top features are trained more often → higher importance.

Optimize for Classification

- Stage 1: Unsupervised Pretraining



- Stage 2: Knowledge Distillation for Inference

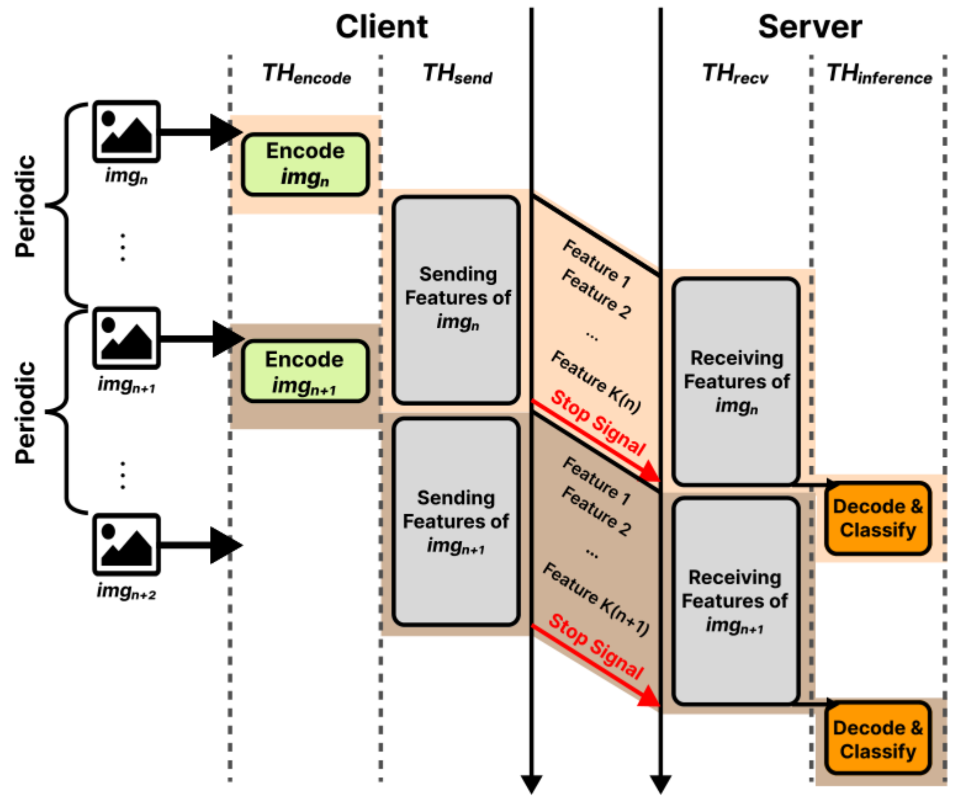


Progressive Image Offloading Pipeline

- Distributed architecture
  - ❑ encoding on the device (client)
  - ❑ offloading
  - ❑ decoding and classification (edge server).

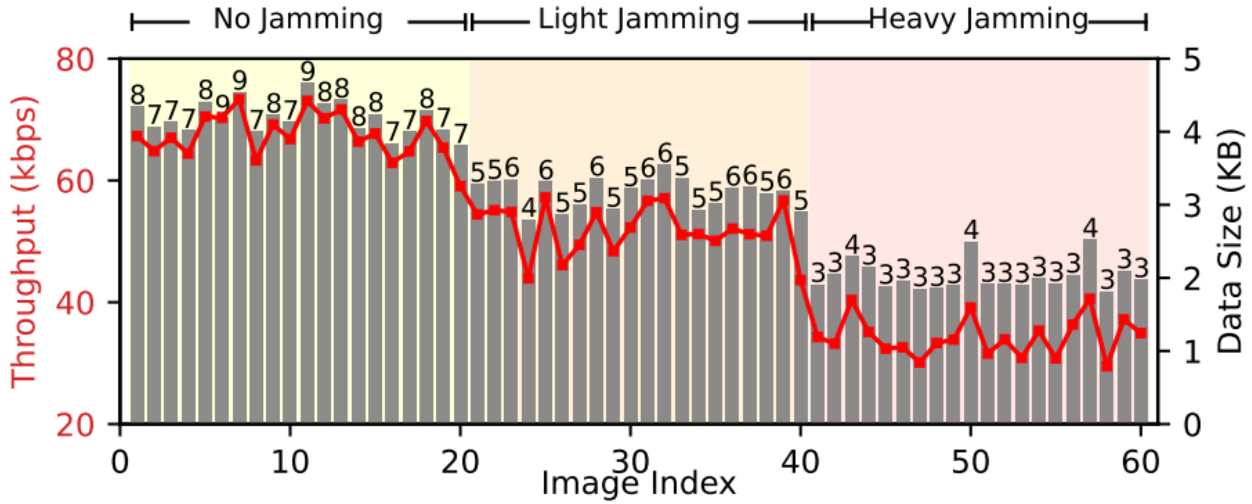
- Features are sent in 64-byte data blocks

- Terminate offloading upon the the next image arrival (deadline).



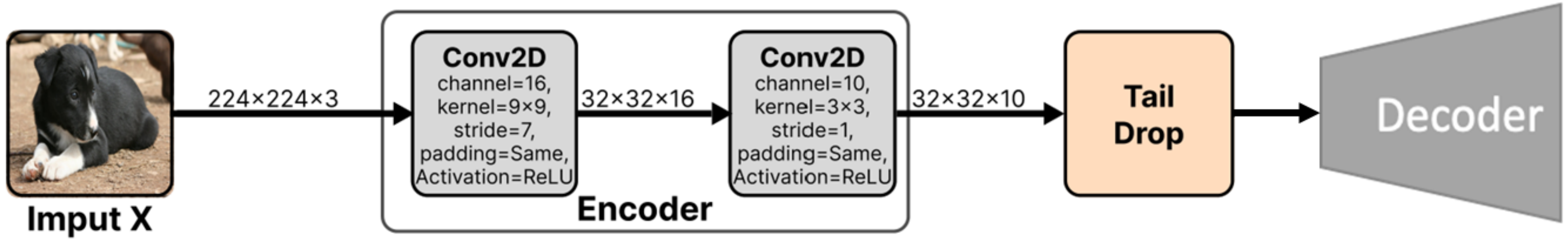
Adapting to Network Variations

- A typical run of the offloading process under varying network bandwidth.
  - Red curve: throughput of offloading traffic.
  - Gray bars: data size of the fully offloaded features.
  - Numbers: the number of features fully offloaded to the edge server.



Model Implementation

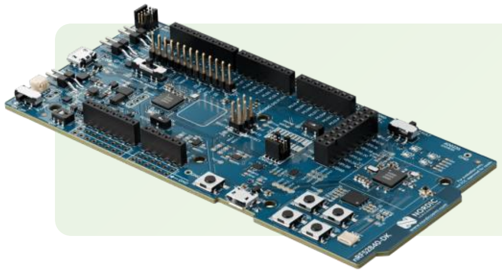
- Autoencoder with an asymmetric design
  - Encoder: **2-layer** convolutional network to match the capacity of the device
  - Decoder: **5-layer** convolutional network with higher dimension and complexity



- Additional training data from the ImageNet validation set
  - 35,000/5,000 for training and validation
  - 2,000 for testbed experiments.

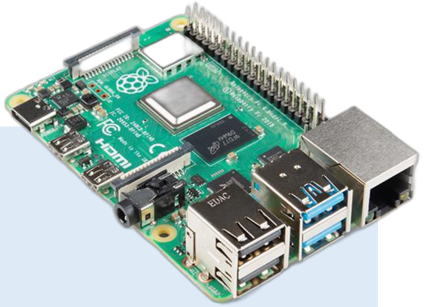
Experimental Setup

➢ Floor plan and hardware details for the experiments



IEEE 802.15.4 (2.4GHz) radio module:

- A pair of nRF52840 development kits (Nordic Semiconductor).

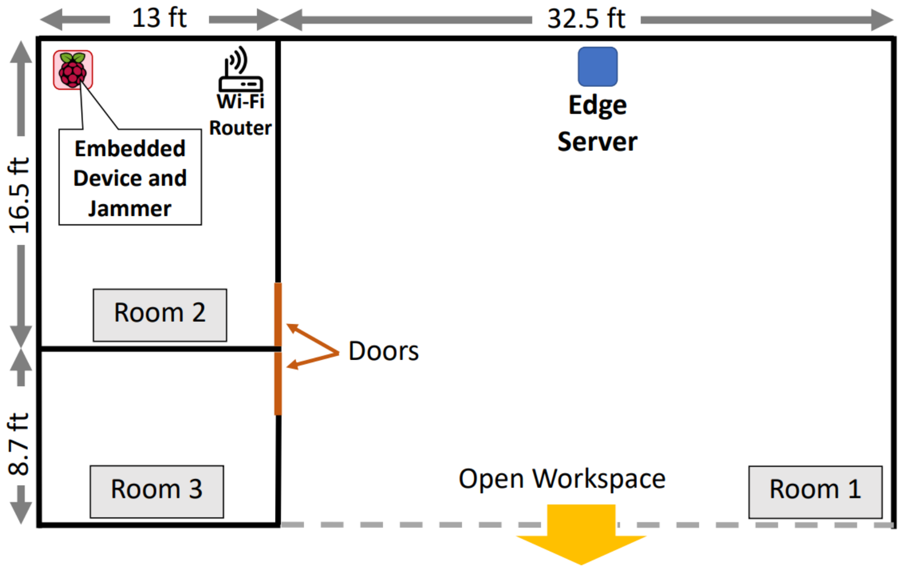


Embedded IoT device:

- Raspberry Pi 4B

Jammer:

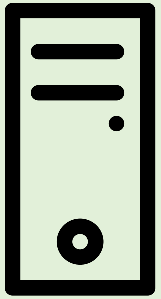
- Raspberry Pi 3B



Layout of the indoor lab space.

Edge Server:

- CPU: Intel Core i7-10700K
- GPU: Nvidia GeForce RTX 3090



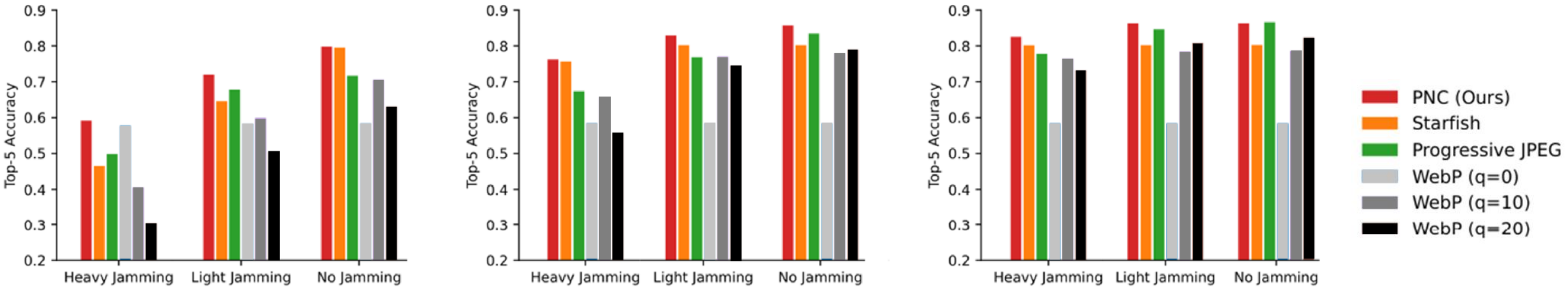
Experimental Setup

- Three levels of network jamming (None, Light, and Heavy).
  - A Raspberry Pi "Jammer" generates 2.4GHz WiFi traffic with `iperf`.
- Baselines

	Traditional Image Encoding	Neural Compression
Non-progressive	JPEG, **WebP**	DeepCOD (Yao et al., 2020), **Starfish** (Hu et al., 2020)
Progressive	**Progressive JPEG**	**RNN-based** (Toderici et al., 2017) **PNC (Our work)**

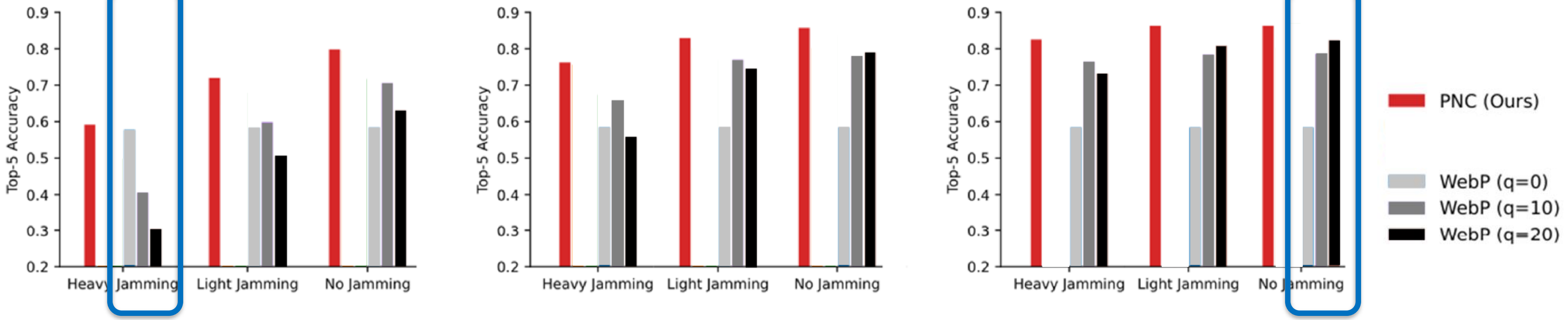
- Metric: Top-5 Accuracy

Classification Accuracy



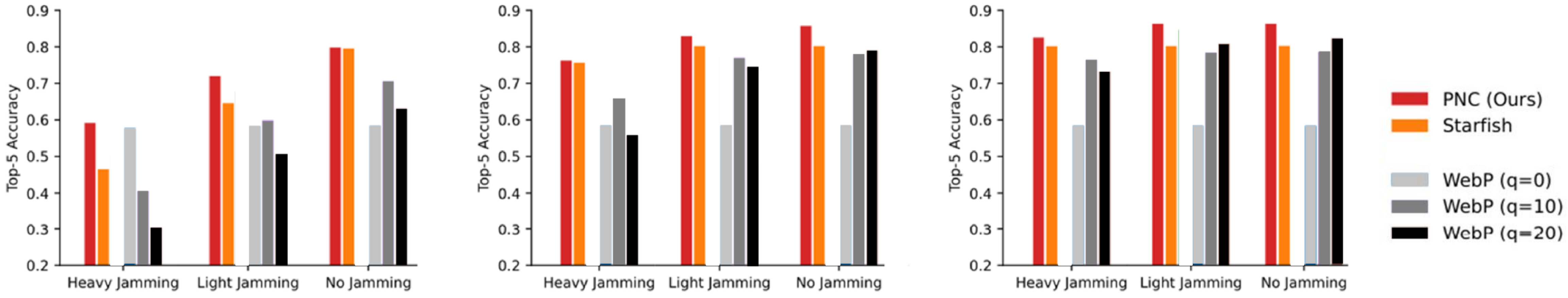
(a) T = 300ms (b) T = 500ms (c) T = 700ms

Classification Accuracy



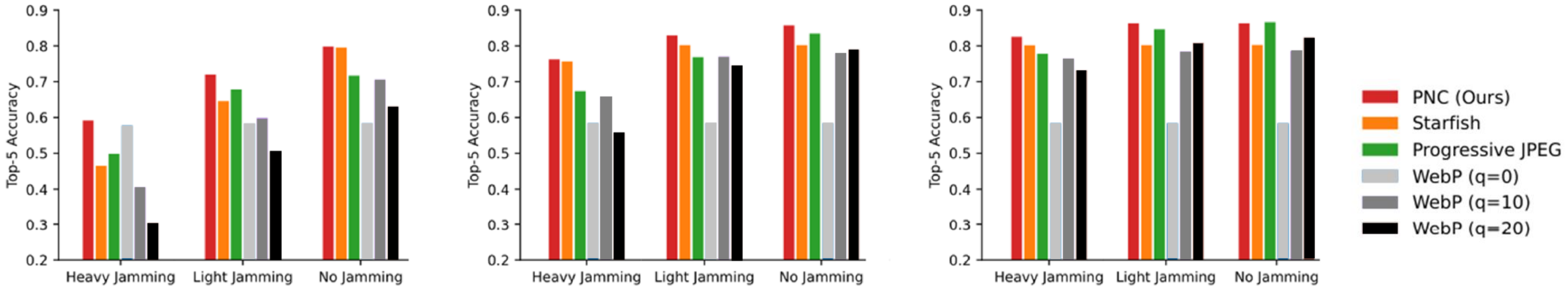
(a) T = 300ms (b) T = 500ms (c) T = 700ms

Classification Accuracy



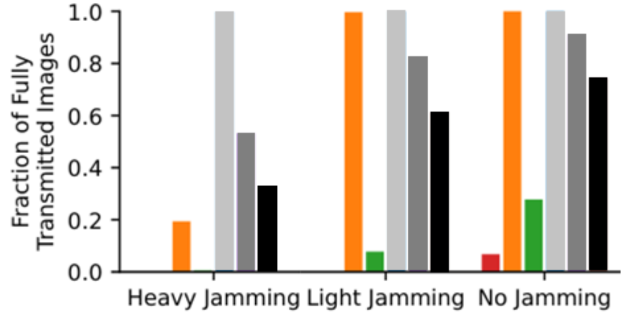
(a) T = 300ms (b) T = 500ms (c) T = 700ms

Classification Accuracy



(a) T = 300ms (b) T = 500ms (c) T = 700ms

❑ Ratio of images fully offloaded.



(b) T = 500ms

Encoding Efficiency

- PNC shows a comparable encoding overhead as traditional image encoders.
  - q represents the quality factor.

	Method	Configuration	Average Encoding Overhead (ms)
	PNC	#thread $= 1$	11.8
Traditional Image Encoding	WebP	$q = 0$	32.5
		$q = 20$	48.6
	Prog. JPEG	$q = 30$	5.8
Neural-network-based Encoding	Starfish	One Patch	62.9
	RNN-TFLite	One Iteration	1900

Progressive Neural Compression (PNC)

- PNC is designed for adaptive image offloading under timing constraints.
  - Training for progressive behavior through stochastic **tail-drop**
  - Optimized classification through **knowledge distillation** for inference
  - **Asymmetric** autoencoder design for encoding efficiency

- PNC has been implemented in a distributed real-time image classification testbed.

- PNC's characteristics on an edge computing testbed
  - **Classification accuracy**
  - **Encoding efficiency**
  - **Adaptability** to different deadlines and varying bandwidth

The End of Presentation