Deep Compressive Offloading: Speeding Up Neural Network Inference by Trading Edge Computation for Network Latency

Shuochao Yao[1], Jinyang Li, Dongxin Liu, Tianshi Wang,
Shengzhong Liu, Huajie Shao, Tarek Abdelzaher
[1]George Mason University, University of Illinois at Urbana-Champaign
[1]shuochao@gmu.edu, {jinyang7, dongxin3, tianshi3, sl29, hshao5, sl29, zaher}@illinois.edu

SenSys 2020 Award

Best Paper Award

- sensys-s3-paper1: Zero-Wire: A Deterministic and Low-Latency Wireless Bus through Symbol-Synchronous Transmission of Optical Signals
  Jonathan Oostvogels, Fan Yang (imec-DistriNet, KU Leuven); Sam Michiels, Danny Hughes (imec-DistriNet KU Leuven)
- sensys-s7-paper2: Deep Compressive Offloading: Speeding Up Neural Network Inference by Trading Edge Computation for Network Latency
  Shuochao Yao (George Mason University); Jinyang Li, Dongxin Liu, Tianshi Wang, Shengzhong Liu, Huajie Shao, Tarek Abdelzaher (UIUC)

Outline

- Background: deep learning on the edge
- Problems and Previous Work
- Compressive Sensing
- DeepCOD Framework
- Evaluation and Results
- Conclusions

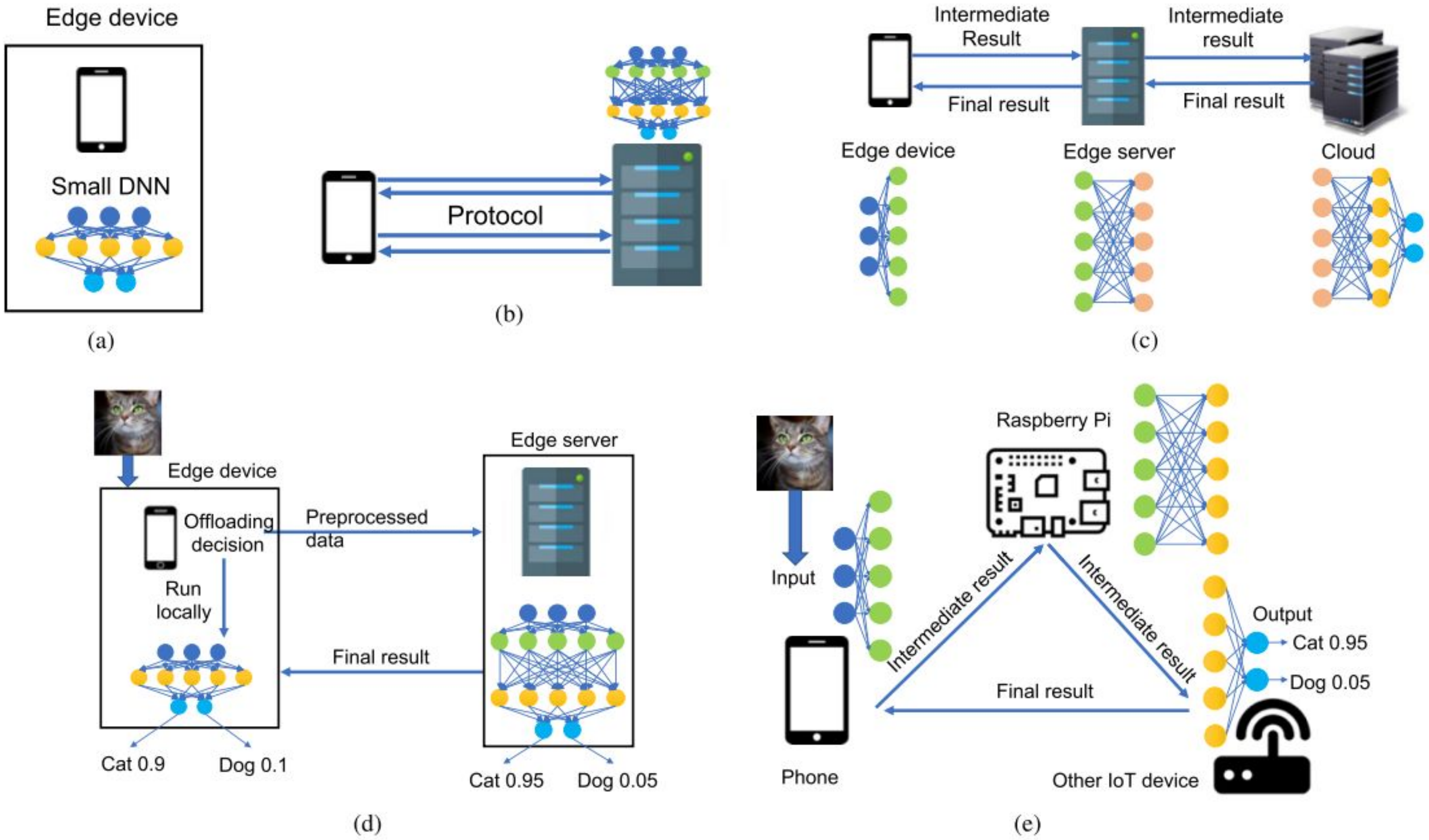


Fig. 5. ***Architectures for deep learning inference with edge computing. (a) On-device computation. (b) Secure two-party communication. (c) Computing across edge devices with DNN model partitioning. (d) Offloading with model selection. (e) Distributed computing with DNN model partitioning.***

Fast Inference Methods

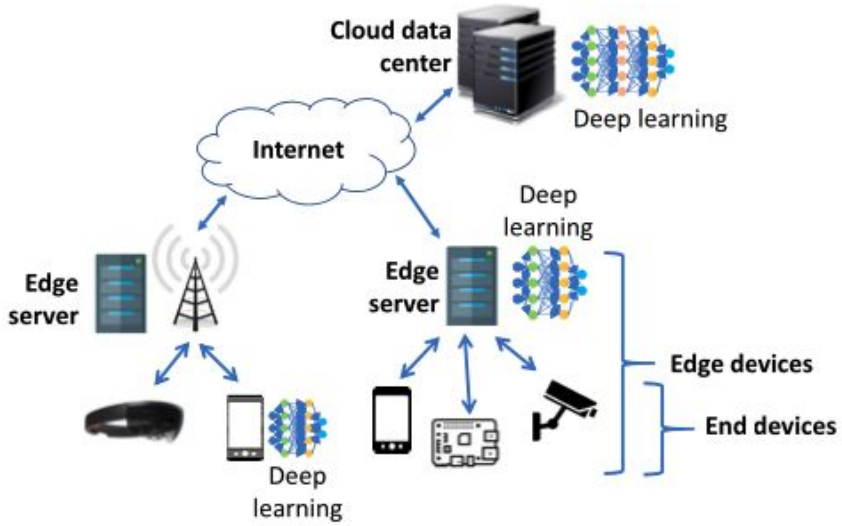
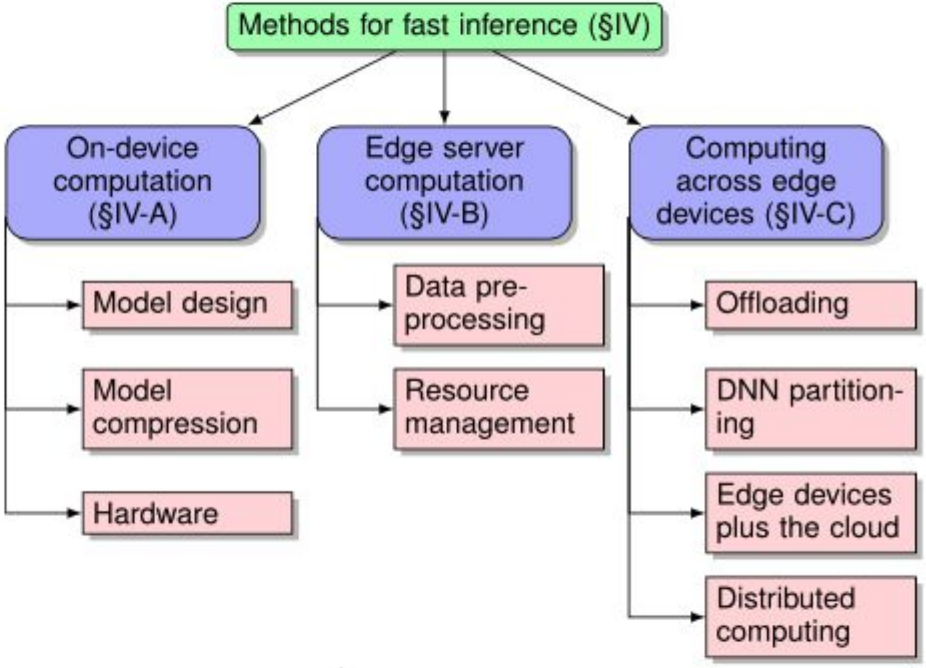


Fig. 1. ***Deep learning can execute on edge devices (i.e., end devices and edge servers) and on cloud data centers.***



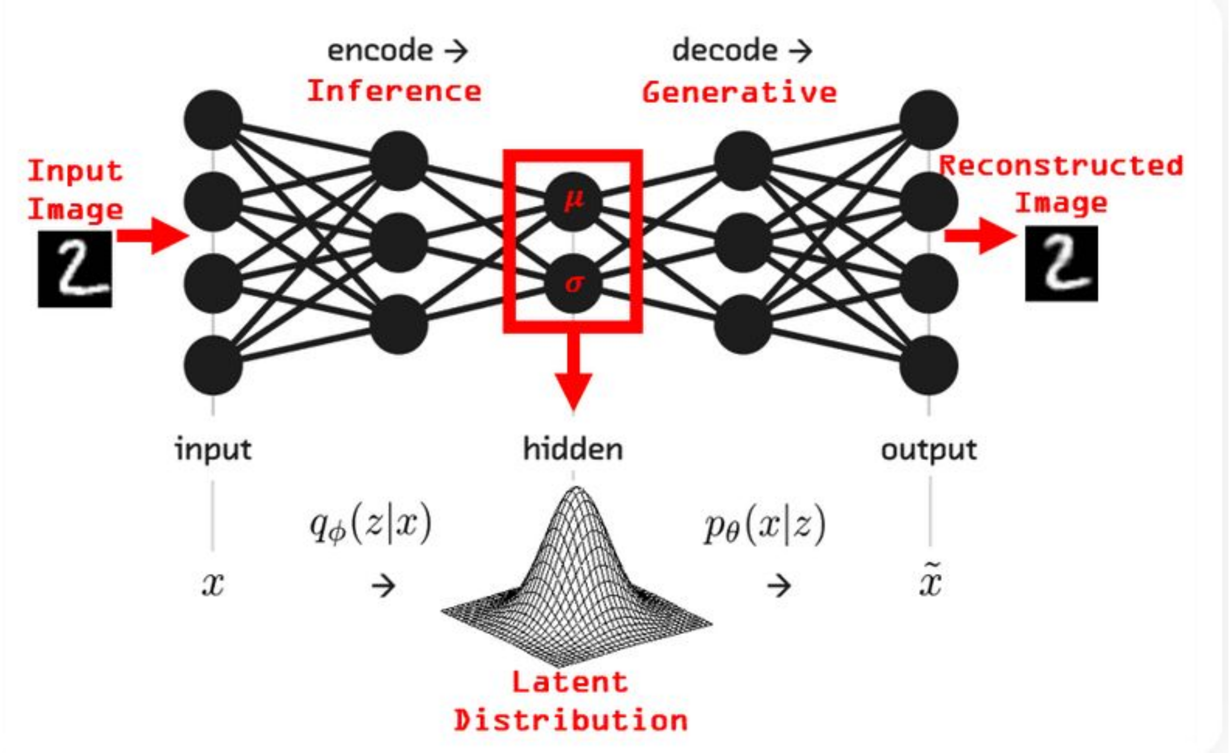
Demo


Could we deploy Object Detection App (YoLo) on a Raspberry Pi?
✓ we can deploy the object detection on a Raspberry Pi with edge offloading.
✗ But the network latency is not neglectable. The averaged delay is around 800 ms.
? How can we speed up

Problems and Previous Work

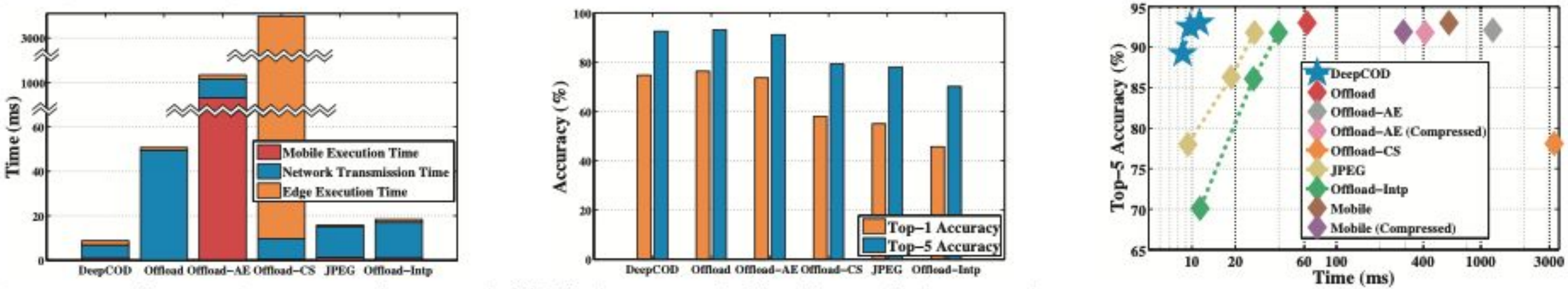
- Problem: high latency caused by transferring data to edge server
- Previous work
  - Decide optimal offloading point in NN based on current computing resources and network conditions
    - Intermediate data sizes of first several layers are still large
  - Use inferior but efficient local model to cut down freq of offloading requests
    - Up to 10% accuracy loss
  - Learning-based data compression: **auto-encoder**, compress data locally for offloading then reconstruct on the server side
    - symmetric processing burden on encoder and decoder side
- DeepCOD: asymmetric encoder/decoder framework
  - Imbalanced “autoencoder” using compressing sensing theory
  - Much less overhead on end-device, most burden on server side
  - Improves latency significantly with no degradation in inference accuracy

AutoEncoder



Performance of State-of-the-Art

Deep Compressive Offloading   SenSys '20, November 16–19, 2020, Virtual Event, Japan



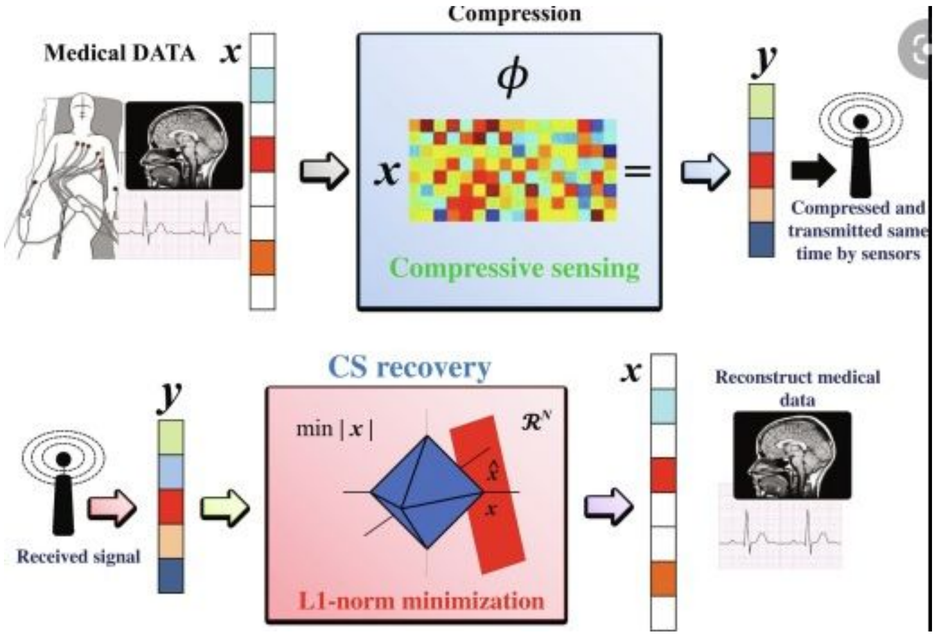
(a) Decompose latency into network transmission, mobile and edge execution time.

(b) The impacts of offloading techniques on inference accuracy.

(c) Deep compressive offloading achieves Pareto optimality (the time axis in log scale).

Figure 1: A case study of image recognition application with ResNet-50 model. Google Pixel is connected to an edge with Titan V through 450Mbps WiFi. Images on the mobile device are offloaded to the edge server with various offloading techniques.

DeepCOD = Compressive Sensing + Deep Learning



$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\mathbf{y} - \mathbf{E}\mathbf{x}\|^2$$ DCT, Fourier, wavelet

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \|\mathbf{y} - \mathbf{E}G_\theta(\mathbf{z})\|^2$$ Pretrained generative neural networks

- **Slow reconstruction** due to iterative optimization / backpropagation
- **Loss in accuracy**
  - Measurement matrix E and pretrained GNN cannot perfectly fit application-specific data

- **Deep compressive offloading**
  - Moving the computation load from online iterations steps to offline training
  - Reconstruct using one-shot inference on decoder

$$\arg\min_{\theta,\phi} \left\|\mathbf{x} - G_\theta(\mathbf{E}_\phi \circledast \mathbf{x})\right\|^2 \quad (5)$$

where $\circledast$ denotes the convolution operation, θ, and ϕ are sets of learnable parameters for decoder and encoder.

More details on regularization and knowledge distillation

DeepCOD Framework

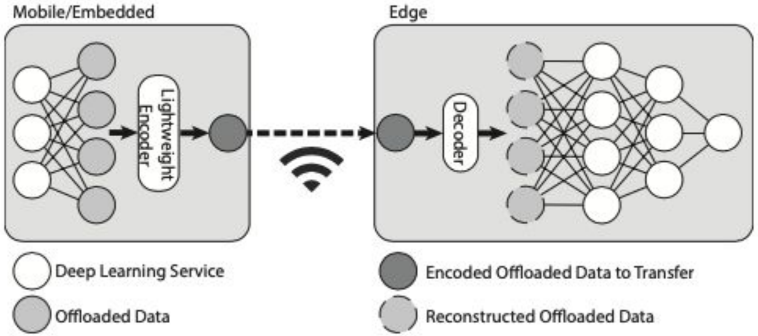


Figure 2: The Deep Compressive Offloading designs with a lightweight encoder on the local device to compress data and a decoder on the edge server to reconstruct.

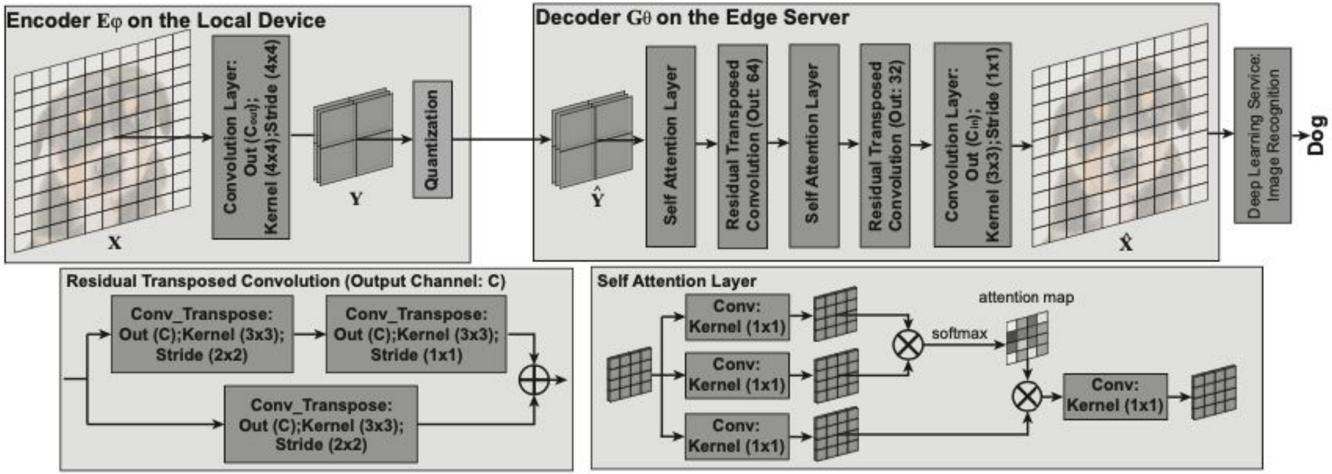
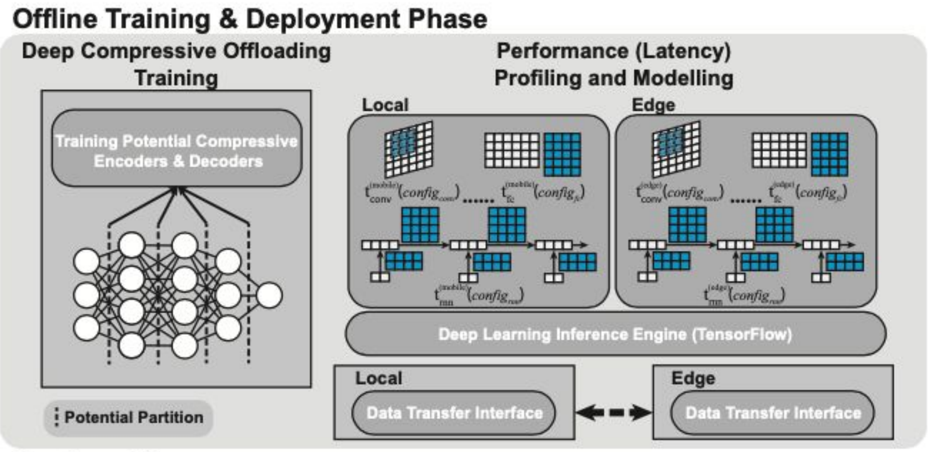


Figure 3: The default designs and configurations of decoder and decoder structures that used in all our experiments.

DeepCOD System



FastDeepIoT

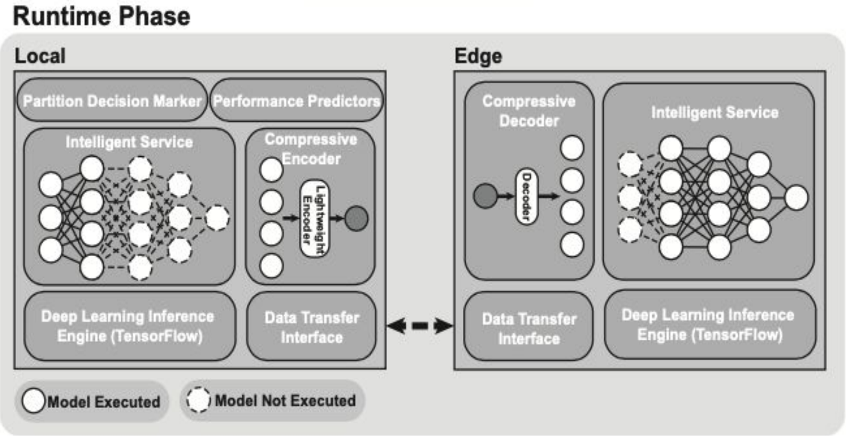


Figure 5: System overview of DeepCOD.

$$\underset{p\in\{1,\cdots,P\}}{\arg\min}\ t_p^{(edge)} + t_p^{(local)} + d_p/B.$$

Implementation and Experiments Setup

- Two phones
  - Google Pixel and Nexus 6
- Two type of connections
  - WiFi (450Mbps) and LTE
- Two types of GPU on linux edge server
  - Nvidia Titan V
  - Nvidia GeForce GTX Titan X
- Three DL tasks
  - Image recognition using ResNet-50 on ImageNet dataset
  - Speech recognition using DeepSpeech
  - Object detection using YOLOv3 (demo with raspberry pi offloading)
- Five baselines
  - Offload-Intp
  - Offload-CS: DCT and wavelet basis, ISTA for reconstruction
  - Offload-Lossy: JPEG, Huffman-coding
  - Offload-AE+: enhanced auto-encoders with state-of-the-art model compression
  - Offload: no processing on offloaded data

Trade-off between Inference Acc and Compression Ratio

Table 1: Tradeoff between model inference accuracy (Top-5 classification accuracy) and compression ratio of offloaded data for image recognition service with ResNet-50 model through WiFi connection with 450Mbps bandwidth.

	Input			Block1			Block2		
	Size	t_{net}	Acc	Size	t_{net}	Acc	Size	t_{net}	Acc
DeepCOD	**5.7KB (0.97%)**	**4.3ms (7.1%)**	**92.0% (-1.1%)**	**980B (0.12%)**	**2.8ms (3.5%)**	**92.0% (-1.1%)**	**245B (0.02%)**	**2.4ms (1.5%)**	**92.1% (-1.0%)**
Offload-CS	18.5KB (3.1%)	10.9ms (18.0%)	91.7% (-1.4%)	94.6KB (12.1%)	16.0ms (20.1%)	80.1% (-13.1%)	177KB (11.3%)	26.8ms (16.8%)	79.0% (-14.1%)
Offload-Intp	24.8KB (4.2%)	12.8ms (21.2%)	91.8% (-1.3%)	95.2KB (12.1%)	16.1ms (20.2%)	75.7% (-17.4%)	178KB (11.4%)	26.8ms (16.8%)	78.8% (-14.3%)
Offload-Lossy	19.5KB (3.3%)	12.3ms (20.3%)	91.7% (-1.4%)	94.7KB (12.1%)	16.0ms (20.1%)	77.1% (-16.0%)	178KB (11.4%)	26.8ms (16.8%)	79.0% (-14.1%)
Offload-AE+	12.2KB (2.1%)	7.8ms (12.9%)	92.0% (-1.1%)	14.7KB (1.9%)	8.5ms (10.7%)	92.0% (-1.1%)	17.2KB (1.1%)	8.6ms (5.4%)	92.1% (-1.0%)
Offload	588KB	60.5ms	93.1%	784KB	79.6ms	93.1%	1568KB	159.2ms	93.1%

	Block3			Block4		
	Size	t_{net}	Acc	Size	t_{net}	Acc
DeepCOD	**184B (0.02%)**	**2.3ms (2.9%)**	**92.0% (-1.1%)**	**123B (0.03%)**	**2.2ms (5.1%)**	**92.1% (-1.0%)**
Offload-CS	87.4KB (11.1%)	15.3ms (19.2%)	86.5% (-6.6%)	80.4KB (20.5%)	15.2ms (35.1%)	89.0% (-4.1%)
Offload-Intp	87.5KB (11.2%)	15.3ms (19.2%)	86.9% (-6.2%)	80.6KB (20.6%)	15.2ms (35.1%)	89.5% (-3.6%)
Offload-Lossy	87.4KB (11.1%)	15.3ms (19.2%)	86.6% (-6.5%)	80.8KB (20.6%)	15.2ms (35.1%)	89.1% (-4.0%)
Offload-AE+	12.3KB (1.6%)	7.7ms (9.7%)	92.0% (-1.1%)	6.7KB (1.7%)	5.3ms (12.2%)	92.1% (-1.0%)
Offload	784KB	79.6ms	93.1%	392KB	43.3ms	93.1%

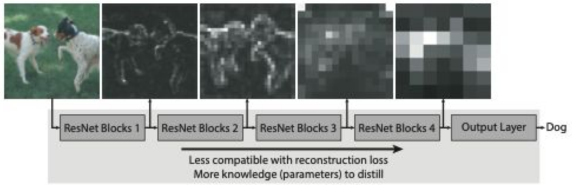


Figure 4: An illustration of intermediate representations in ResNet-50 image recognition service.

Trade-off between Inference Acc and Compression Ratio

Table 2: Tradeoff between Word Error Rate (WER) and compression ratio of offloaded data for speech recognition service with DeepSpeech model through WiFi connection with 450Mbps bandwidth.

	Input			Layer1			Layer2		
	Size	t_{net}	WER	Size	t_{net}	WER	Size	t_{net}	WER
DeepCOD	**17.9KB (1.5%)**	**8.8ms (8.2%)**	**0.085 (+0.003)**	**7.3KB (0.2%)**	**6.9ms (1.8%)**	**0.087 (+0.005)**	**5.5KB (0.1%)**	**4.3ms (1.1%)**	**0.085 (+0.003)**
Offload-CS	140KB (12.1%)	25.8ms (24.1%)	0.231 (+0.149)	551KB (11.5%)	50.6ms (13.4%)	0.144 (+0.062)	550KB (11.5%)	50.5ms (13.4%)	0.128 (+0.046)
Offload-Intp	142KB (12.3%)	25.9ms (24.2%)	0.262 (+0.18)	550KB (11.5%)	50.6ms (13.4%)	0.148 (+0.066)	550KB (11.5%)	50.6ms (13.4%)	0.313 (+0.231)
Offload-Lossy	144KB (12.4%)	25.9ms (24.2%)	0.264 (+0.182)	551KB (11.5%)	50.6ms (13.4%)	0.145 (+0.063)	551KB (22.4%)	50.6ms (13.4%)	0.135 (+0.053)
Offload-AE+	21.7KB (1.9%)	8.9ms (8.3%)	0.088 (+0.006)	45KB (0.9%)	23.3ms (6.2%)	0.09 (+0.008)	30KB (0.6%)	20.3ms (5.4%)	0.087 (+0.005)
Offload	1158KB	107.2ms	0.082	4800KB	377.9ms	0.082	4800KB	377.9ms	0.082

	Layer3			Layer4			Layer5		
	Size	t_{net}	WER	Size	t_{net}	WER	Size	t_{net}	WER
DeepCOD	**4.4KB (0.1%)**	**4.0ms (1.1%)**	**0.085 (+0.003)**	**3.7KB (0.08%)**	**3.8ms (1.0%)**	**0.084 (+0.002)**	**2.9KB (0.06%)**	**3.7ms (1.0%)**	**0.084 (+0.002)**
Offload-CS	552KB (11.5%)	50.7ms (13.4%)	0.145 (+0.063)	550KB (11.5%)	50.5ms (13.4%)	0.126 (+0.044)	550KB (11.5%)	50.5ms (13.4%)	0.131 (+0.049)
Offload-Intp	551KB (11.5%)	50.6ms (13.4%)	0.099 (+0.017)	550KB (11.5%)	50.5ms (13.4%)	0.098 (+0.016)	550KB (11.5%)	50.5ms (13.4%)	0.191 (+0.109)
Offload-Lossy	551KB (11.5%)	50.6ms (13.4%)	0.159 (+0.077)	551KB (11.5%)	50.6ms (13.4%)	0.119 (+0.037)	551KB (11.5%)	50.6ms (13.4%)	0.133 (+0.051)
Offload-AE+	25.5KB (0.5%)	16.3ms (4.3%)	0.087 (+0.005)	21KB (0.4%)	15.4ms (4.1%)	0.087 (+0.005)	16.5KB (0.3%)	8.4ms (2.2%)	0.086 (+0.004)
Offload	4800KB	377.9ms	0.082	4800KB	377.9ms	0.082	4800KB	377.9ms	0.082

End-to-End Latency

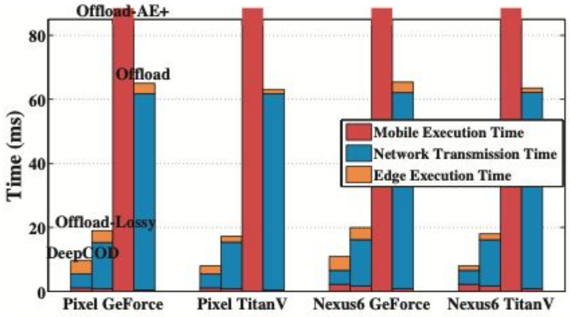


Figure 6: End-to-end offloading latency of image recognition through WiFi with 450Mbps bandwidth.

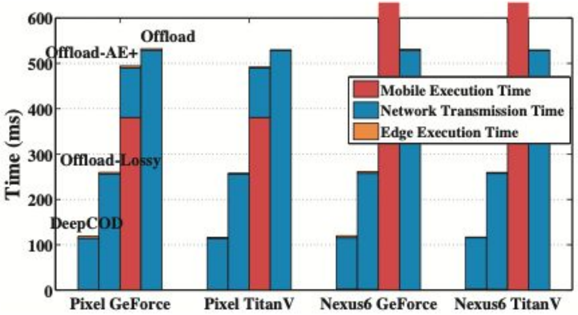


Figure 8: End-to-end offloading latency of image recognition through LTE.

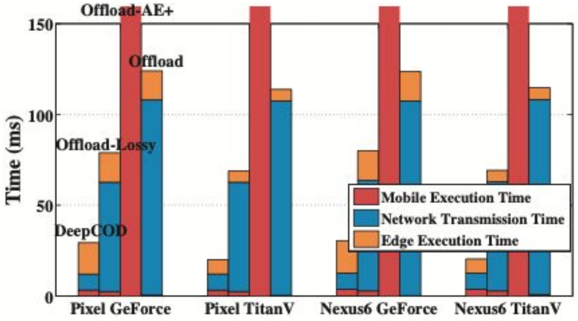


Figure 7: End-to-end offloading latency of speech recognition through WiFi with 450Mbps bandwidth.

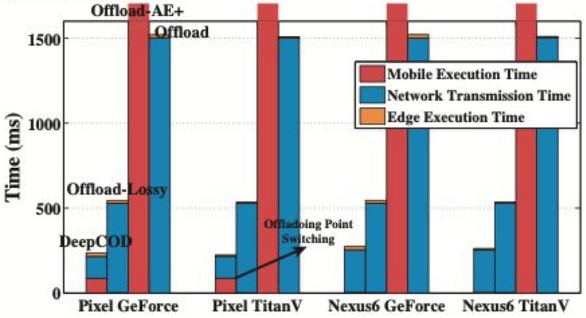
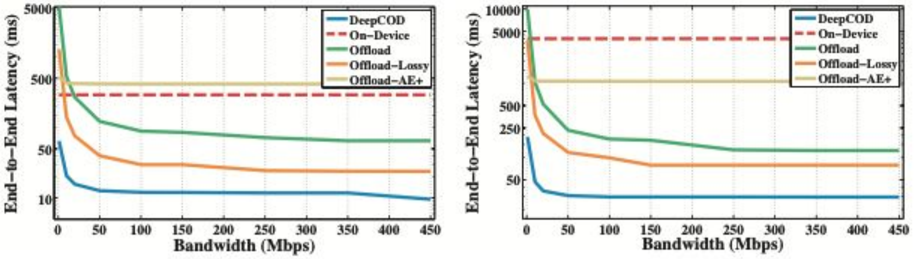


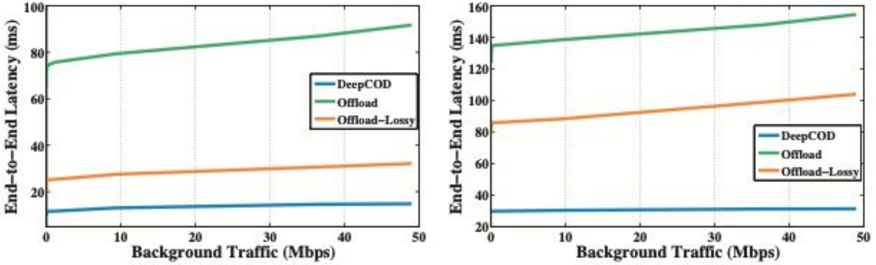
Figure 9: End-to-end offloading latency of speech recognition through LTE.

Impact of Bandwidth & Background Traffic



(a) Image recognition (b) Speech recognition

Figure 10: End-to-end offloading latency under various bandwidth conditions (y-axis log scale)



(a) Image recognition (b) Speech recognition

Figure 11: End-to-end offloading latency under various background network traffic.

Offline Training Overhead

Table 4: Training overhead of DeepCOD.

	DeepCOD	Original
ImageNet (1.3M Pictures)	4.8 ± 0.8h	134h
LibriSpeech (300h Speech)	1.6 ± 0.3h	23h

Conclusions

- DeepCOD: deep compressive offloading
- Asymmetric encoder-decoder by merging compressive sensing and deep learning
- Reduce end-to-end latency by a factor of 2-35
- At most 1% accuracy loss
- Application-agnostic, domain-agnostic