ACM multimedia
Chengdu, China OCT 20-24 2021

Boosting Mobile CNN Inference through Semantic Memory

Yun Li[1,4], Chen Zhang[2,*], Shihao Han[3,4], Li Lyna Zhang[4], Baoqun Yin[1,*],

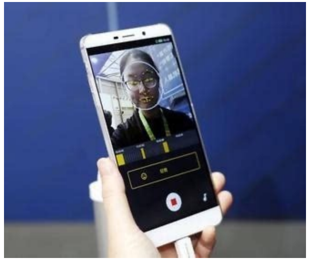
Yunxin Liu[5], Mengwei Xu[6]

[1]University of Science and Technology of China, [2]Alibaba Group, [3]Rose-Hulman Institute of Technology,

[4]Microsoft Research Asia, [5]Tsinghua University, [6]Beijing University of Posts and Telecommunications

(* indicates the corresponding author)

CNNs have catalyzed many emerging mobile vision tasks



Face Recognition

Classification

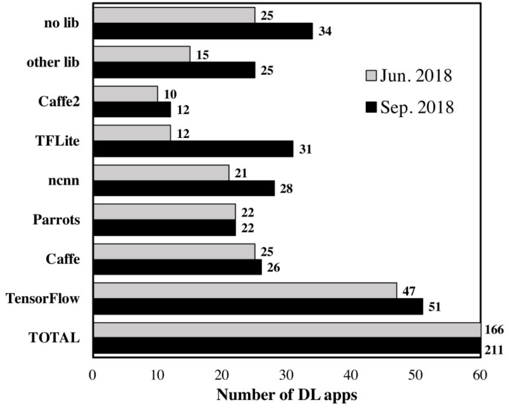
Action Recognition

- DNNs have achieved great success in many continuous mobile vision applications.
- The mobile/wearable devices need to perform CNN inference in real time on these video images.

Fast inference on mobile devices is urgent

- The CNN executions are costly.
  - high time complexity and energy-consuming
- Offloading to the cloud?
  - tight delay constraint and data privacy concerns
- A notable trend is on-device CNN inference

Cloud? Privacy concerns

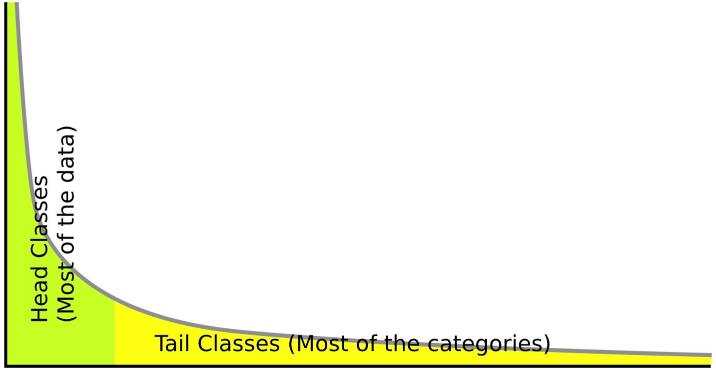


increased by 27% with 3 month

Two critical observations

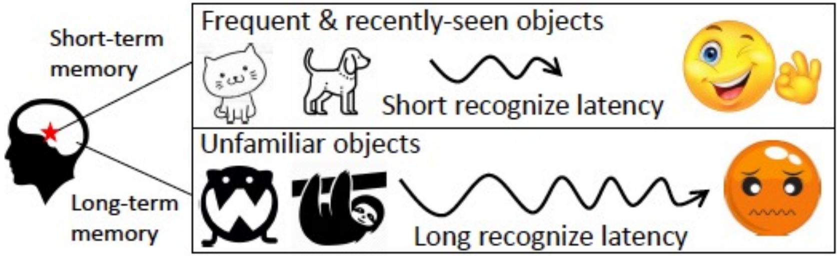
- The temporal locality in mobile video streams
  - Recently seen objects are more likely to appear again in the next few frames

- Long-tail distribution
  - The frequency of object occurrence in the mobile video streams typically follows a long-tail distribution



How does the human brain solve it?

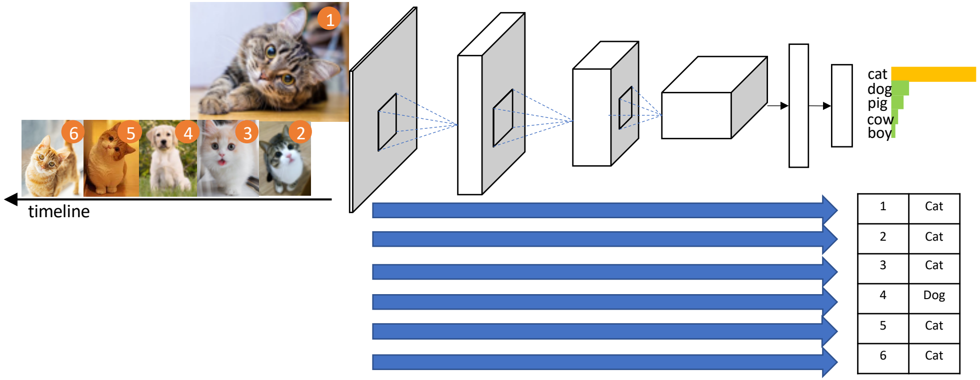
- Human brain leverages temporal redundancy with priming effect
- Priming effect : a psychology phenomenon whereby exposure to one stimulus improves a response to a subsequent stimulus, without conscious guidance or intention.
- Priming effect is related to the long- and short-term memory of human brains



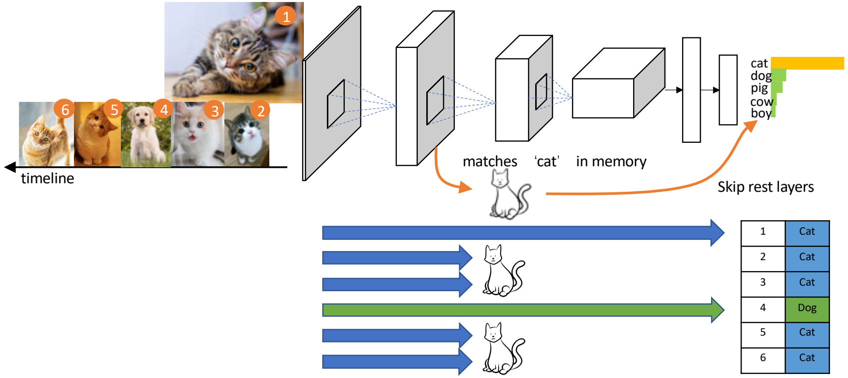
(a) The priming effect

Motivation

- Infuse the priming effect with CNN inference

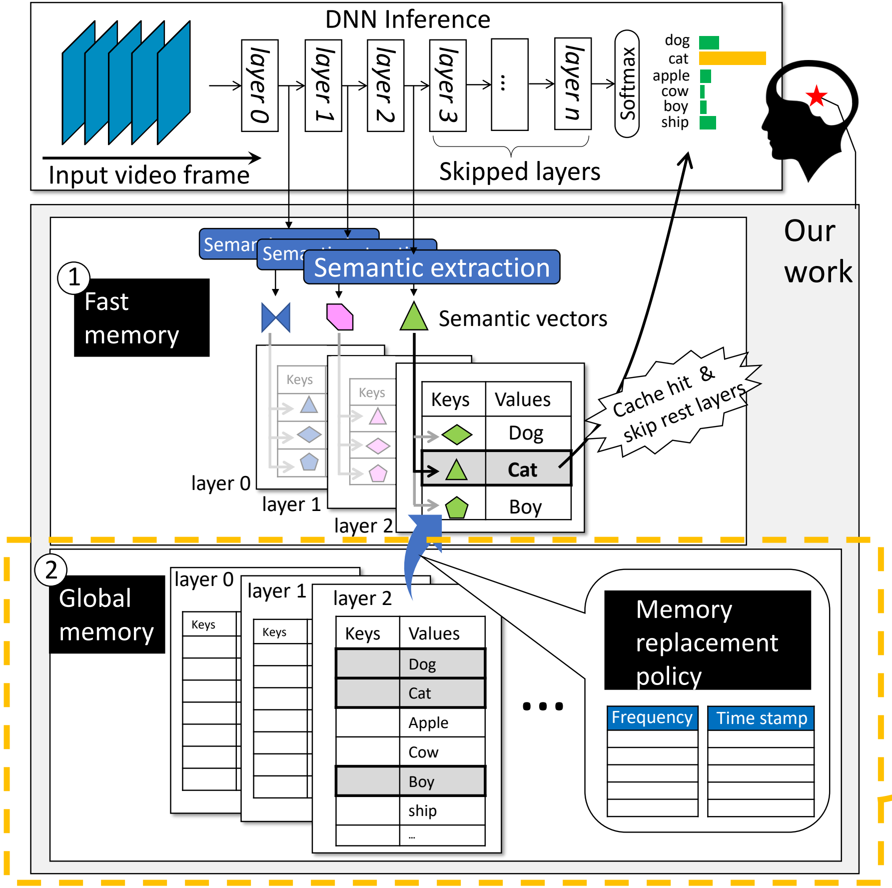


Traditional CNN model: carry out fully inference every time



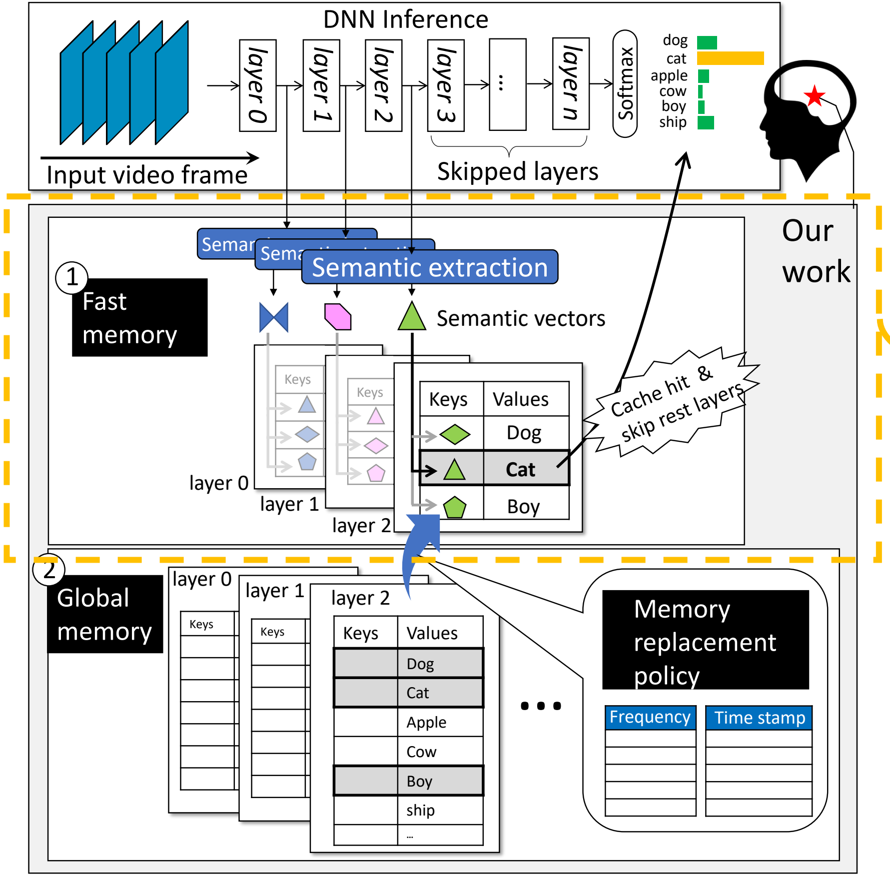
Rarely seen objects: fully inference
Recently seen objects: early exit

Our proposal: semantic memory (SMTM)



- store the semantic centers of all classes
- Memory Replacement Policy: cache the frequency and time stamp of recently seen classes
- select few classes with the greatest probability of recurring

Our proposal: semantic memory (SMTM)



- extracts the intermediate feature of layer by layer
- match the extracted feature with the features of selected classes
- if matched, skip the rest layers and output the final results directly

Challenges

- Efficient memory encoding against CNN models' over-parameterization
  - extremely large volume of intermediate data
  - directly look up feature maps is cumbersome
  - take about 10ms even with GPU acceleration
- Obtaining speedup by high-level vision semantics
  - previous methods: low-level vision information
  - human brain: makes recognition by high-level features
  - traditional execution flow can not reuse semantics
- Battling dynamics on scenario variation
  - the scene change drastically
  - the scene complexity is not known in advance
  - real scenario data ≠ training data, more complicated

SMTM tech#1: Semantic Memory Encoding

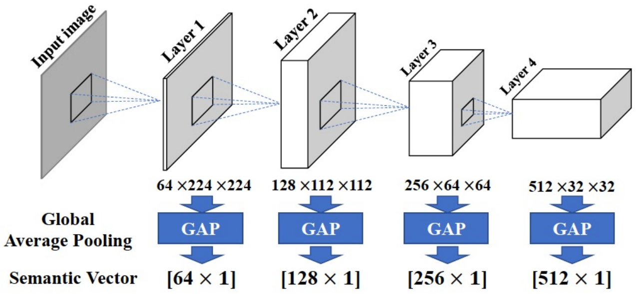


Figure 3: Semantic vectors extraction.

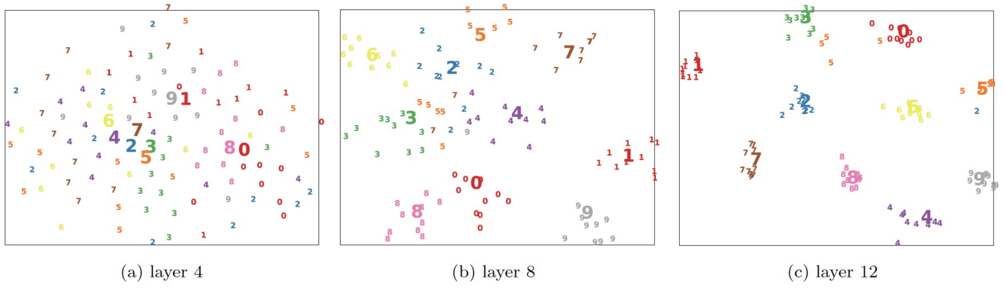


Figure 4: Visualized separability of semantic vectors for different VGG16 layers, showing that going deeper the semantic vectors can be more accurately separated.

- adopt global average pooling (GAP) to perform dimensionality reduction
  - much more light-weight
  - an effective indicator: clear separability in hidden layers
- adopt the cosine distance to evaluate the distance of different objects
  - $s_j^l = \xi(SV^l, SC_j^l) \in [-1,1], j \in [1,n]$
  - $sep^l = \frac{s_H^l - s_{SH}^l}{s_{SH}^l}$

SMTM tech#2: Early Exit

- The separability in shallow layers is not as strong or stable as the deeper layers

- The cross-layer cumulative similarity
  - $SA_j^l = \sum_{l_0}^{l} s_j^{l_0} \times weight_{l_0}, j \in [1, n]$
  - $weight_{l_0} = 2^{l_0 - 1}, 1 + 2^1 + 2^2 + \cdots + 2^{n-1} = 2^n - 1$

- the accumulated confidence (AC)
  - SA_H^l : the highest similarity accumulation result
  - SA_{SH}^l : the second-highest result
  - $AC^l = \frac{SA_H^l - SA_{SH}^l}{SA_{SH}^l}$
  - If $AC^l > global\ threshold, exit!$

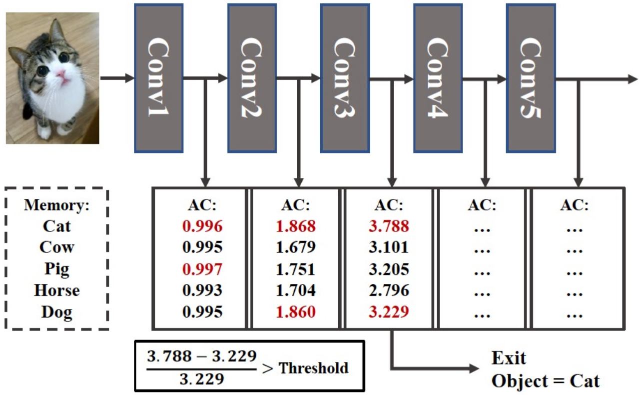


Figure 5: Memory look up by accumulated confidence (AC) metric. Memory: objects in memory.

SMTM tech#3: Adaptive Priming Memory

1) cache replacement policy 2) adaptive cache size 3) adaptive semantic centers

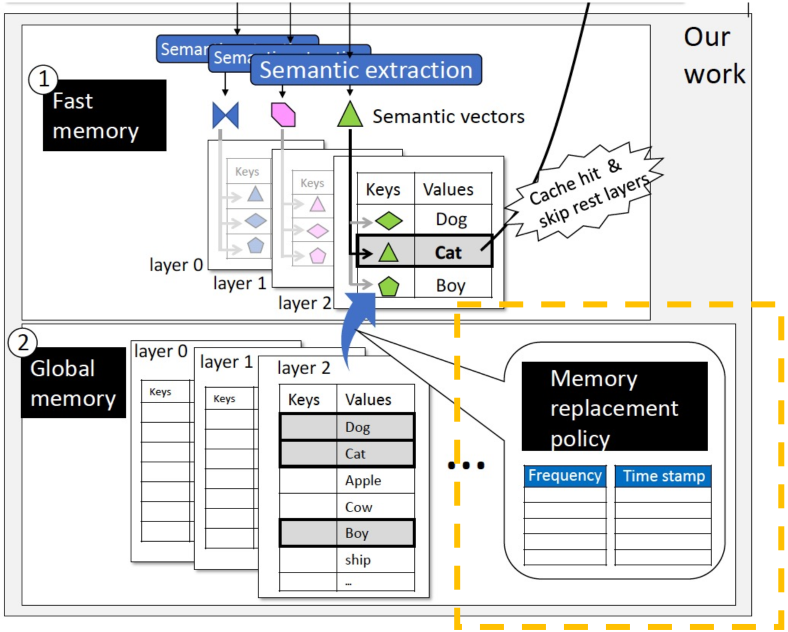
- Frequency table:
  - keeps a record of the number of times that each object class presented in history
- Time-stamp table
  - keeps a record of the recency of each object class
  - the forgetting mechanism

$$\psi_i = \psi_i \times (0.25)^{\left\lfloor \frac{TS_i}{W} \right\rfloor}$$

- The replacement policy
  - takes the Top-k highest score

$$Score_i = Score_i \times (0.25)^{\left\lfloor \frac{TS_i}{W} \right\rfloor}$$

  - cache the Top-k objects in the fast memory



SMTM tech#3: Adaptive Priming Memory

1) cache replacement policy **2) adaptive cache size** **3) adaptive semantic centers**

✓ Adaptive cache size

- probability estimation method
- $P(\theta \in \Psi) = \sum_{i=1}^{k} \frac{score_i}{\sum_{i=1}^{n} score_i}$
  - Confidence Level (CL) 95%
  - adjust the k, $P(\theta \in \Psi) > \mathrm{CL}$
- The experiments show a 21.6% hit ratio improvement

✓ Adaptive semantic centers

- warms up using the training data
- update in weighted average manner
  - $\widehat{SC}_{l_0}^{j} = \frac{SC_{l_0}^{j} \cdot m_{l_0}^{j} + SV_{l_0}^{j}}{m_{l_0}^{j} + 1}$
- The experiments show a 16.9% accuracy improvement

SMTM Implementation

- Computing framework:
  - ncnn
- Test Platform:
  - Google Pixel 4XL (Qualcomm Snapdragon 855 Processor)
- Datasets:
  - Action Recognition (UCF-101), Classification (long-tail Cifar-100)
- Five popular CNN models:
  - AlexNet, GoogleNet, ResNet50, MobileNet V2, VGG16
- Five evaluation metrics:
  - latency improvement
  - accuracy loss
  - energy saving
  - memory overhead
  - early exit ratio

SMTM Evaluation

- latency improvement

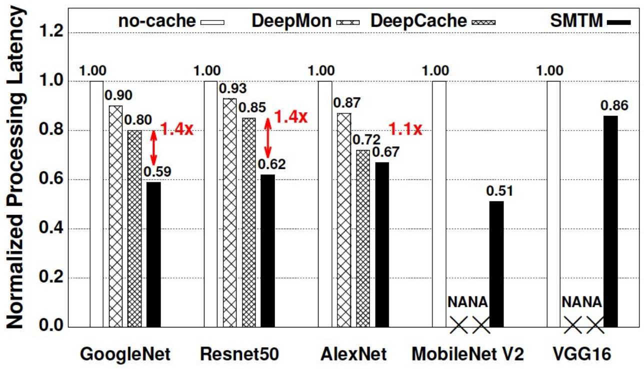


Figure 6: Average processing latency with CPU (w/o SIMD) on action recognition (AlexNet, GoogleNet, ResNet50, MobileNet V2) and classification (VGG16). 'NA': 'not applicable'. SMTM speedup the processing time by 1.1×-1.4× comparing to DeepCache [52], and 1.3×-1.5× comparing to DeepMon [24]. DeepCache's and DeepMon's implementation is not compatible with the two models MobileNet V2 and VGG16, so we are not able to reproduce some results. 'NA': 'not applicable'.

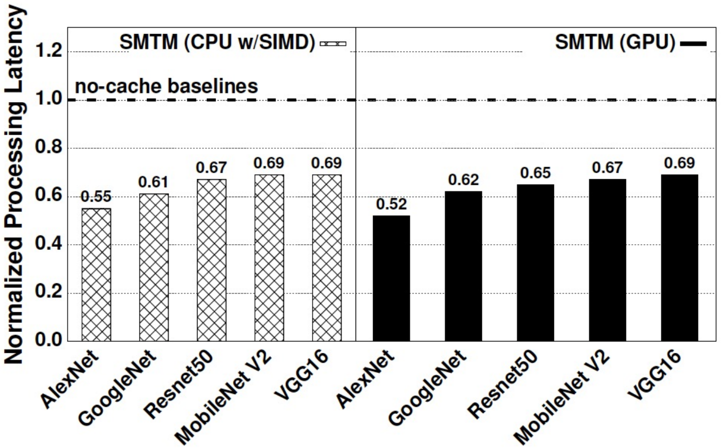


Figure 7: Average processing latency of SMTM with mobile CPU (w/SIMD) and mobile GPU on action recognition (AlexNet, GoogleNet, ResNet50, MobileNet V2) and classification (VGG16).

Mobile CPU and GPU: 30%-50% latency reduction

Compare with SOTA: 1.1X – 1.5X

SMTM Implementation

- Accuracy Drop, Memory Overhead

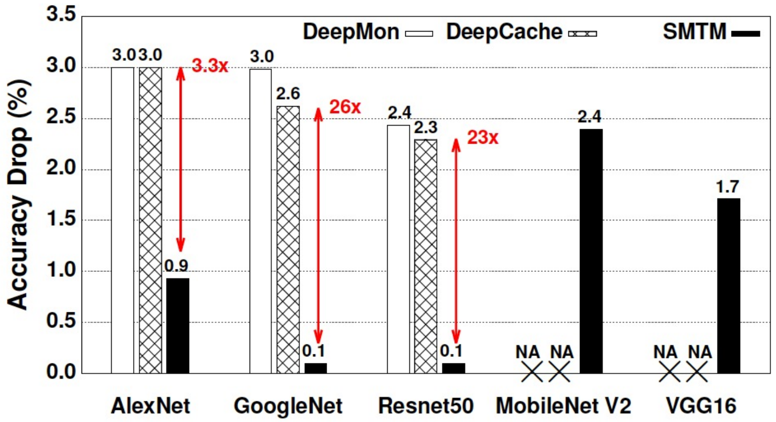


Figure 8: Top-1 accuracy drop of SMTM on action recognition (AlexNet, GoogleNet, ResNet50, MobileNet V2) and classification (VGG16). 'NA': 'not applicable'.

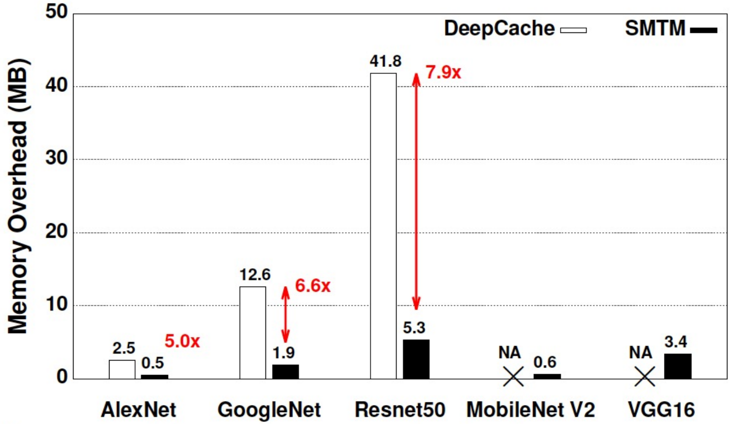


Figure 9: The memory overhead of SMTM on action recognition (AlexNet, GoogleNet, ResNet50, MobileNet V2) and classification (VGG16). 'NA': 'not applicable'.

Accuracy drop: 1% on average

Memory overhead:

20% of SOTA, less than 5% of original models

SMTM Implementation

- Energy Saving, Early Exit Ratio, the Effect of the Threshold

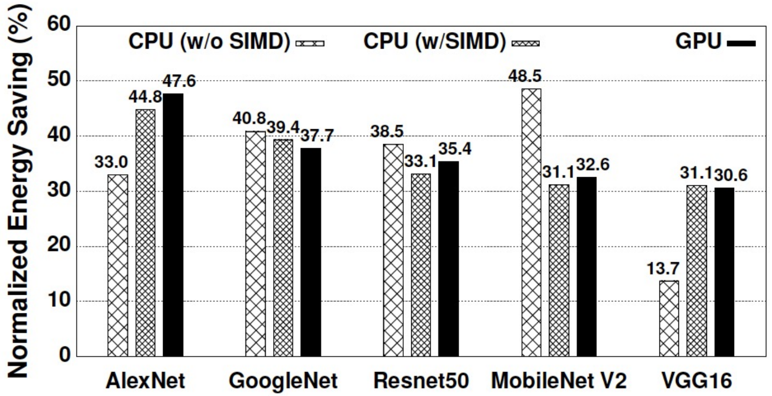
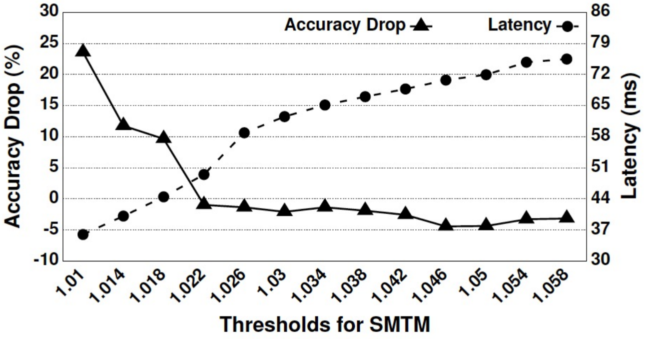
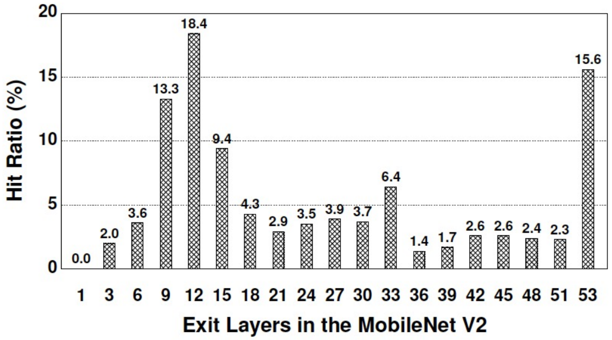


Figure 10: The energy saving ratio with different devices on action recognition (AlexNet, GoogleNet, ResNet50, MobileNet V2) and classification (VGG16).

Energy saving: 36% on average





SMTM Implementation

- The performance of adaptive semantic memory

	Hit ratio	Latency reduction
SMTM (Constant)	65.39%	25.21%
SMTM (Adaptive)	87.00%	38.46%

Table 1: The impact of adaptive cache size. Tested on ResNet50 model.

21.61% hit ratio improvement

13.25% acceleration

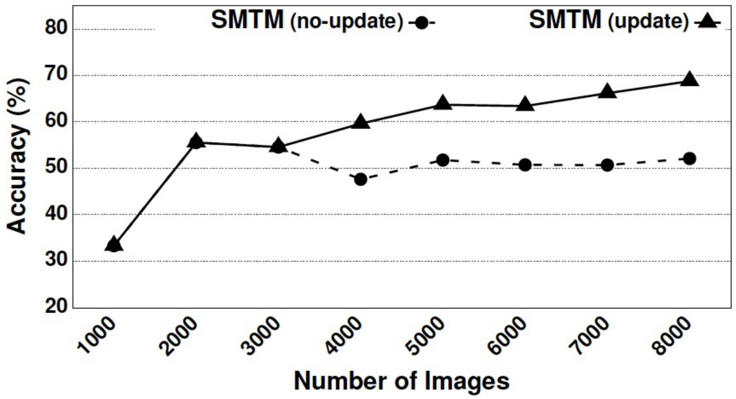


Figure 13: The impact of adaptive semantics center on the prediction accuracy on ResNet50.

16.9% accuracy improvement

Summary

- SMTM: a novel memory mechanism to accelerate CNN-powered mobile vision by infusing the priming effect with CNN inference.
  - speeds up CNN inference for the frequently and recently-seen objects
  - an accurate yet low-cost memory encoder
  - an early exit method
  - an adaptive priming memory policy
- prototype on commodity engine, evaluate on 5 CNN architectures, 2 datasets, on both mobile CPU/GPU
  - Mobile CPU and GPU: 30%-50% latency reduction
  - Only 5% memory overhead

Thank you for watching!