Cost Aware Inference With Early Exits

Vishal Keshav, Berett Babrich, Paige Calisi

Introduction: Prediction At The Edge

- A growing need for using ML Inference on Edge Devices
  - Four categories (right)
- Reduced Latency, but with costs
  - Power, performance
  - Most importantly: **variability**
- Current solutions do not directly address variability
- Our solution: **dynamic real-time inference**

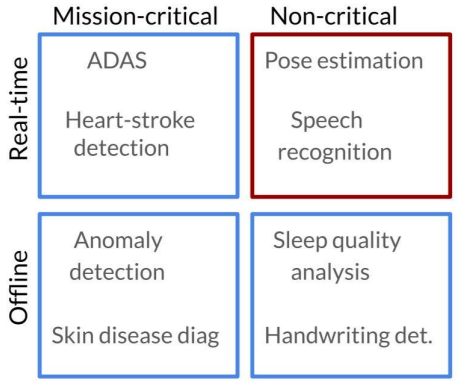
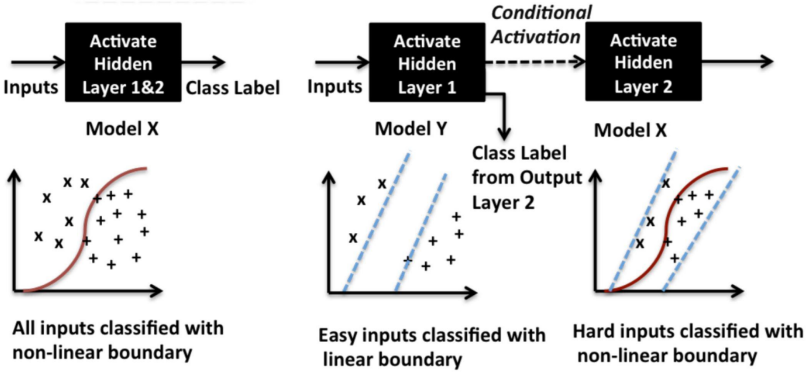


Figure 1: Application categorization.

Background Work: Dynamic Inference

- How to choose when to exit?
- Conditional DeepLearning
  - Prioritizes Accuracy
- Throttling
  - Two stages of training
  - Cannot use existing architectures



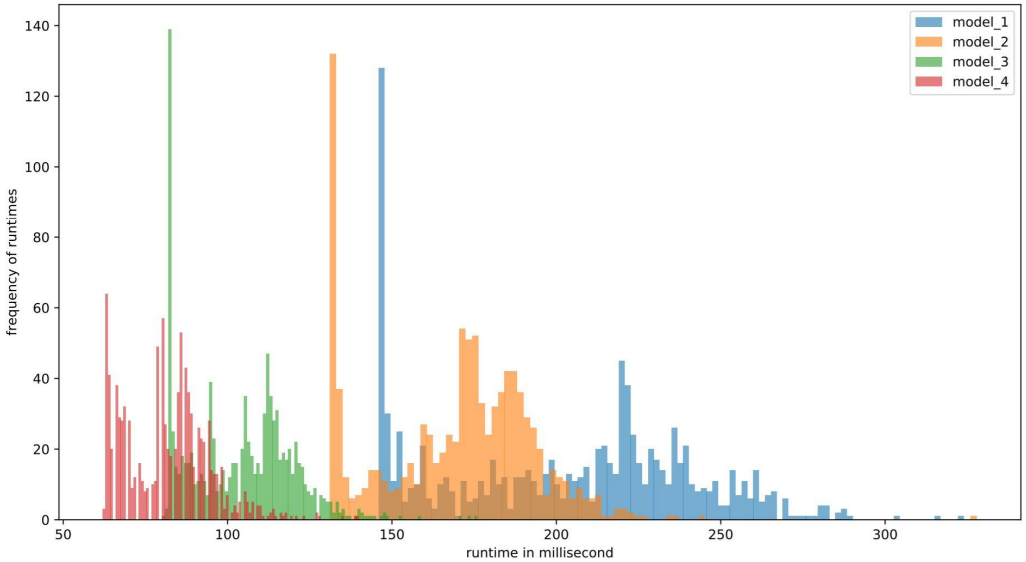
- None of the methods talks about continuous inference, the focus is on improving a single run.

Problem Statement: Evidence

- Runtime varies according to number of MACs in a model
- Experiments were ran with MobileNet Models on an android device

MobileNet version	MACs (Millions)	Paramerers (Millions)	Top 1 accuracy	Lower bound runtime (ms)	Upper bound runtime (ms)
mobilenet v2 1.4 224	582	6.06	75.0	150	280
mobilenet v2 1.3 224	509	5.34	74.4	120	210
mobilenet v2 1.0 224	300	3.47	71.8	70	150
mobilenet v2 0.75 224	209	2.61	69.8	50	110
mobilenet v2 0.5 224	97	1.95	65.4	30	55
mobilenet v2 0.35 224	59	1.66	60.3	20	45

Problem Statement: Evidence



Problem Statement: Formalization

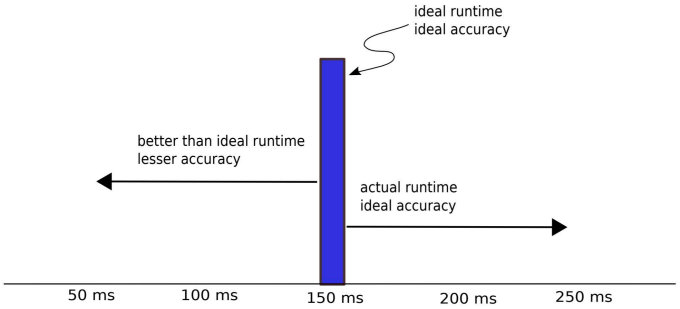
- Thus, there is a tradeoff between accuracy and runtime
- We formalize this problem in the following way:

Choose

$$min_{k \in K} T_p^{(N)}$$

Such That

$$T_p^{(N)} = \sum_{i=1}^{N} (\alpha * R_p^{(i)} + (1 - \alpha) * A_p^{(i)})$$



Continued from previous slide

$$R_p = max(0, r - \hat{r})^2 \tag{1}$$

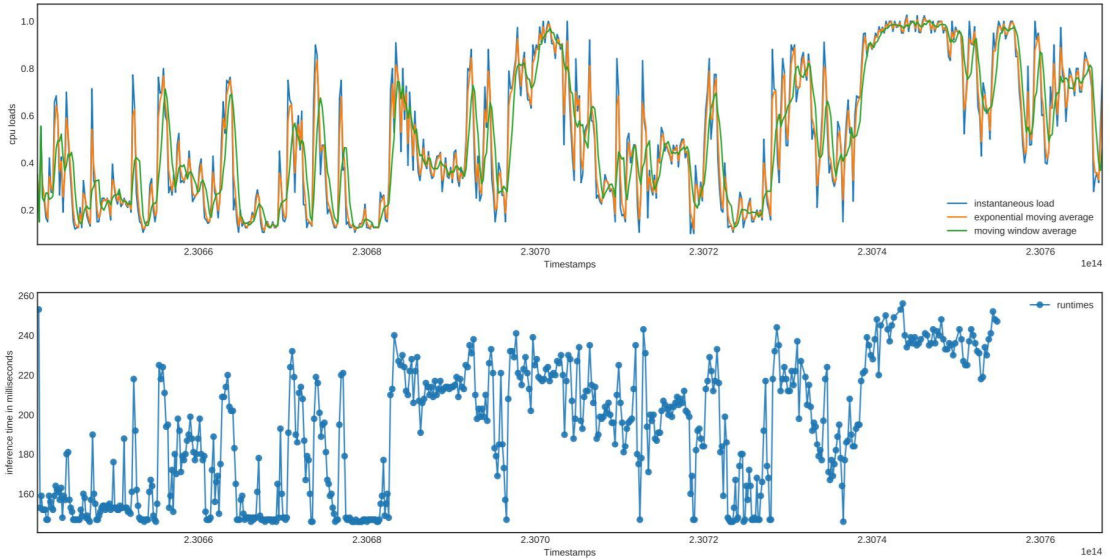
where R_p is the runtime penalty, r is the actual inference runtime and $\hat{r}$ is the ideal runtime, which is the lowest runtime achieved from the best performing model/exit-path.

$$A_p = (\hat{a} - a) \tag{2}$$

where A_p is the accuracy penalty, $\hat{a}$ is the ideal accuracy (highest accuracy achieved from a model/exit-path) and a is the accuracy of the model chosen to run a single inference.

Approach: Estimating CPU Load

- Minimizing the total penalty is done by reliably estimating runtime execution
- Experimentally, we saw that inference time and CPU Load are correlated (right)



Approach: Exit Path Selection Algorithm

- Estimated runtime was calculated by

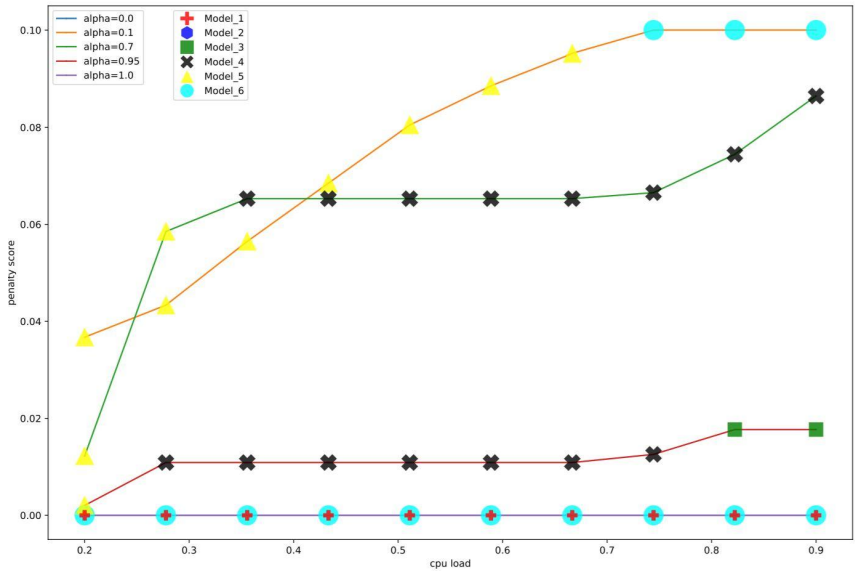
$$\tilde{r_t} = (U - L) * avg_t$$

Algorithm 1 Exit-path selection algorithm

Input: exp_t, **L**, **U**, **A** α
Output: Exit path index
for $i = 1, 2, \ldots |L|$ **do**
    Set $\tilde{r^{(i)}} = L_i + (U_i - L_i) * exp_t$
end for
Set $\hat{r} = L_{|L|}$
Set $\hat{a} = A_{|L|}$
for $i = 1, 2, \ldots |L|$ **do**
    Set $R_p^{(i)} = max(0, \tilde{r^{(i)}} - \hat{r})^2$
    Set $A_p^{(i)} = \hat{a}$ - A_i
end for
Normalize R_p and A_p
$T_p = \alpha * R_p + (1 - \alpha) * A_p$
$k = argmin\ T_p$
Return k

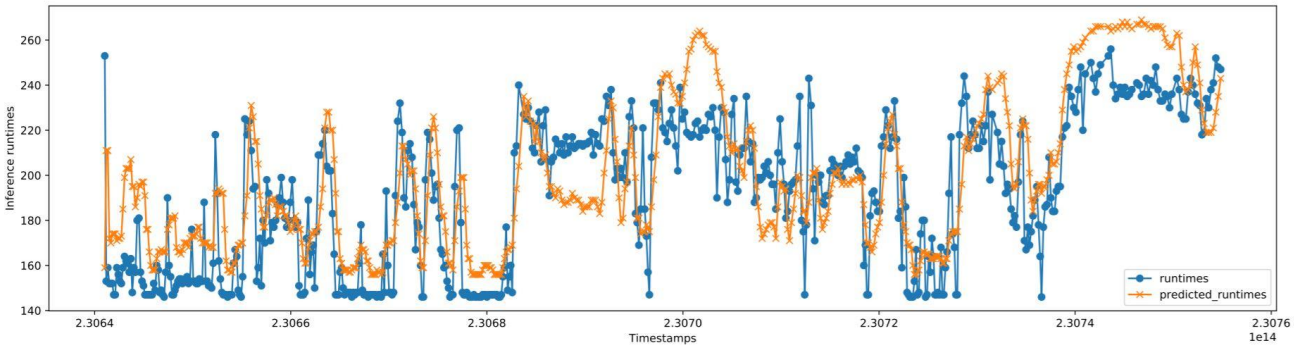
Approach: The Effect of Alpha On Penalty

- Using our algorithm, different choices of alpha lead to different penalties



Experiments: MobileNet Architectures

- Results indicated that CPU load was indeed a good estimate of actual runtime
- Furthermore, our algorithm yielded relatively high accuracies while keeping runtimes low



Experiments: MobileNet Architectures

MobileNet version	Average runtime	Runtime variation	Average accuracy
mobilenet v2 1.4 224	202.02	37.94	75.0
mobilenet v2 1.3 224	169.26	19.05	74.4
mobilenet v2 1.0 224	103.77	1.32	71.8
mobilenet v2 0.75 224	81.49	0.0	69.8
mobilenet v2 0.5 224	40.326	0.0	65.4
mobilenet all ($\alpha = 0.5$)	**170.10**	**31.84**	**74.82**

Experiments: Augmented MobileNets

- A modified TFLite interpreter allowed a mobilenet architecture to have multiple exit points
- Average number of image buffers dropped, as well as variation in number of image buffers

MobileNet version	Average frames dropped	Variation in frame dropped
mobilenet v2 1.4 224	36	15.2
mobilenet all ($\alpha = 0.5$)	**28**	**11.8**

Pros and cons

- Pros
  - A deterministic algorithm, not a black box policy
  - Highly configurable algorithm
  - Reactive/proactive to background cpu load (which is a single proxy for the workload context).
- Cons
  - The hyper-parameter alpha is application specific, and device dependent.
  - A discrete number of exit points/models can limit the algorithm performance.

Thank you for listening

Source Code: https://github.com/vishal-keshav/cost-aware-inference-source-code