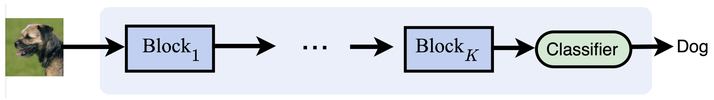
# Learning to Weight Samples for Dynamic Early-exiting Networks

Yizeng Han[1*], Yifan Pu[1*], Zihang Lai[2†], Chaofei Wang[1], Shiji Song[1], Junfeng Cao[3], Wenhui Huang[3], Chao Deng[3], and Gao Huang[1]✉
[1]Tsinghua University, Beijing, China; [2]Carnegie Mellon University, Pennsylvania, United States; [3]China Mobile Research Ins. Beijing, China
*Equal contribution. †Work done during an internship at Tsinghua University. ✉Corresponding author.

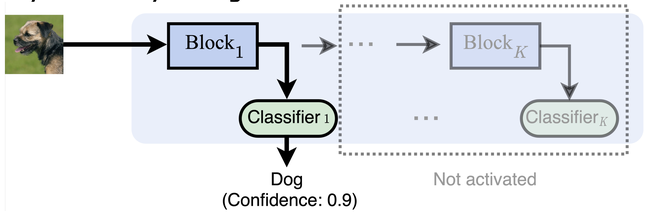
ECCV 2022 EUROPEAN CONFERENCE ON COMPUTER VISION TEL AVIV 2022 — October 23-27, 2022, Tel Aviv

## Background

**Conventional deep network:**

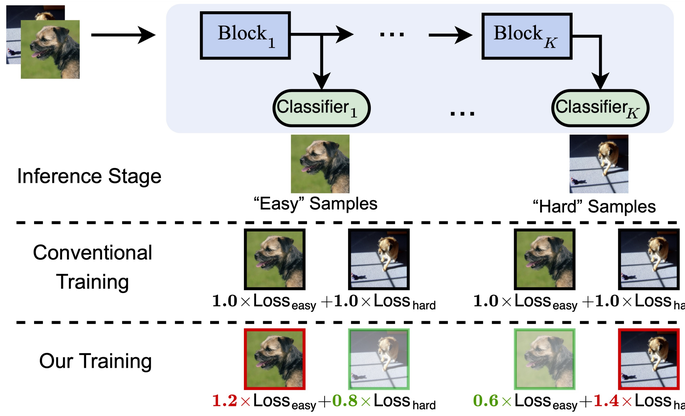



**Dynamic early exiting:**




[Observation]
- "Easy" samples can be output at early exits (classifiers) at test time
- Different exits are responsible with samples with different complexity

## Motivation




- [Intuition] Samples with varying complexity should **contribute unequally** to exits (classifiers) with varying capacity in training.
- [Solution] Sample **weighting**.
- $\mathbf{x}_i, y_i$ : the $i$-th sample and its ground-truth label
- $f(\cdot;\boldsymbol{\Theta}_f)$: multi-exit classification network.
- $f^{(k)}\left(\cdot;\boldsymbol{\Theta}_f^{(k)}\right)$ : the $k$-th sub-network contained in $f$. $k = 1,2,\dots,K$.
- Conventional training objective:

$$\mathcal{L} = \sum_{k=1}^{K}\frac{1}{N}\sum_{i=1}^{N} l_i^{(k)}, \qquad \text{where } l_i^{(k)} = \mathrm{CE}\left(f^{(k)}\left(\mathbf{x}_i;\boldsymbol{\Theta}_f^{(k)}\right), y_i\right)$$

- Our training objective with **sample weighting**:

$$\mathcal{L} = \sum_{k=1}^{K}\frac{1}{N}\sum_{i=1}^{N} w_i^{(k)} l_i^{(k)}$$

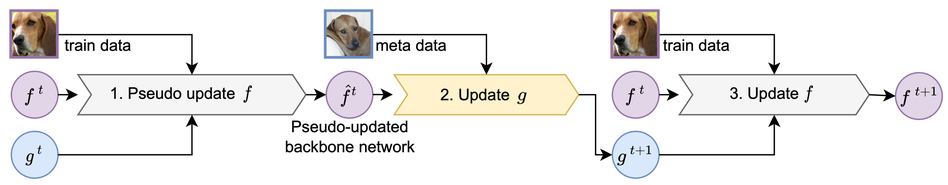
- Weight prediction network $g$: $\boldsymbol{w}_i = g(\boldsymbol{l}_i;\boldsymbol{\Theta}_g)$,

where $\boldsymbol{w}_i = \left[w_i^{(1)}, w_i^{(2)}, \dots, w_i^{(K)}\right]$,

$\boldsymbol{l}_i = \left[l_i^{(1)}, l_i^{(2)}, \dots, l_i^{(K)}\right]$

## Method

### Training pipeline




### Meta-learning Objective

$$\boldsymbol{\Theta}_f^*(\boldsymbol{\Theta}_g) = \underset{\boldsymbol{\Theta}_f}{\operatorname{argmin}}\,\mathcal{L}_{\mathrm{tr}}(\boldsymbol{\Theta}_f,\boldsymbol{\Theta}_g) \triangleq \underset{\boldsymbol{\Theta}_f}{\operatorname{argmin}}\sum_{k=1}^{K}\frac{1}{N}\sum_{i:\mathbf{x}_i\in\mathbf{X}_{\mathrm{tr}}} g^{(k)}(\boldsymbol{l}_i;\boldsymbol{\Theta}_g)\cdot l_i^{(k)}(\boldsymbol{\Theta}_f).$$

$$\boldsymbol{\Theta}_g^* = \underset{\Theta_g}{\operatorname{argmin}}\,\mathcal{L}_{\mathrm{meta}}\left(\boldsymbol{\Theta}_f^*(\boldsymbol{\Theta}_g)\right) \triangleq \underset{\boldsymbol{\Theta}_g}{\operatorname{argmin}}\sum_{k=1}^{K}\frac{1}{N_k}\sum_{j:\mathbf{x}_j\in\mathbf{X}_{\mathrm{meta}}^{(k)}} l^{(k)}\left(\mathbf{x}_j, y_j;\boldsymbol{\Theta}_f^*(\boldsymbol{\Theta}_g)\right).$$

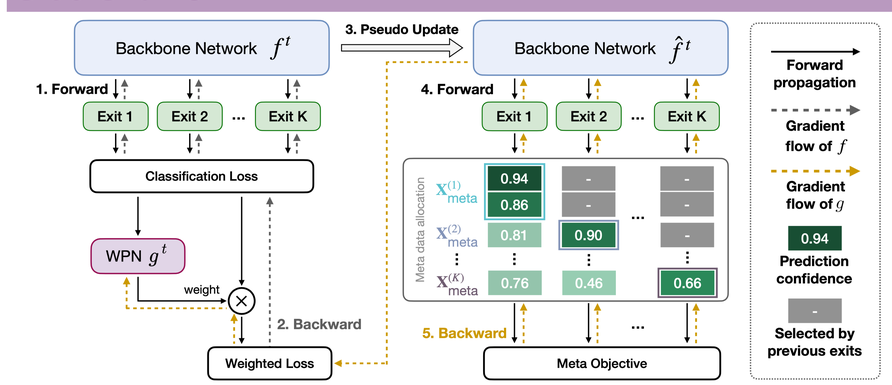
### Meta-learning Optimization

$$\widehat{\Theta}_f^t(\boldsymbol{\Theta}_g) = \boldsymbol{\Theta}_f^t - \alpha\frac{\partial\mathcal{L}_{\mathrm{tr}}(\boldsymbol{\Theta}_f,\boldsymbol{\Theta}_g)}{\partial\boldsymbol{\Theta}_f}\Big|_{\boldsymbol{\Theta}_f^t}$$

$$\boldsymbol{\Theta}_g^{t+1} = \boldsymbol{\Theta}_g^t - \beta\frac{\partial\mathcal{L}_{\mathrm{meta}}\left(\widehat{\Theta}_f^t(\boldsymbol{\Theta}_g)\right)}{\partial\boldsymbol{\Theta}_g}\Big|_{\boldsymbol{\Theta}_g^t}$$

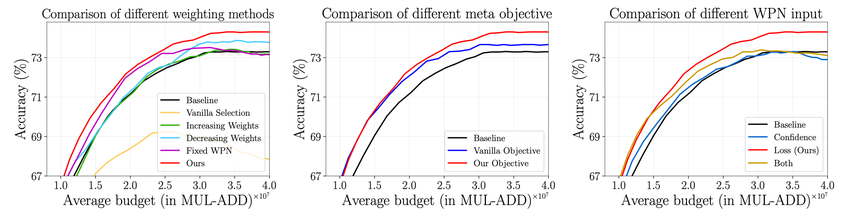
$$\boldsymbol{\Theta}_f^{t+1} = \boldsymbol{\Theta}_f^t - \alpha\frac{\partial\mathcal{L}_{\mathrm{tr}}(\boldsymbol{\Theta}_f,\boldsymbol{\Theta}_g^{t+1})}{\partial\boldsymbol{\Theta}_f}\Big|_{\boldsymbol{\Theta}_f^t}$$

### Gradient Flow



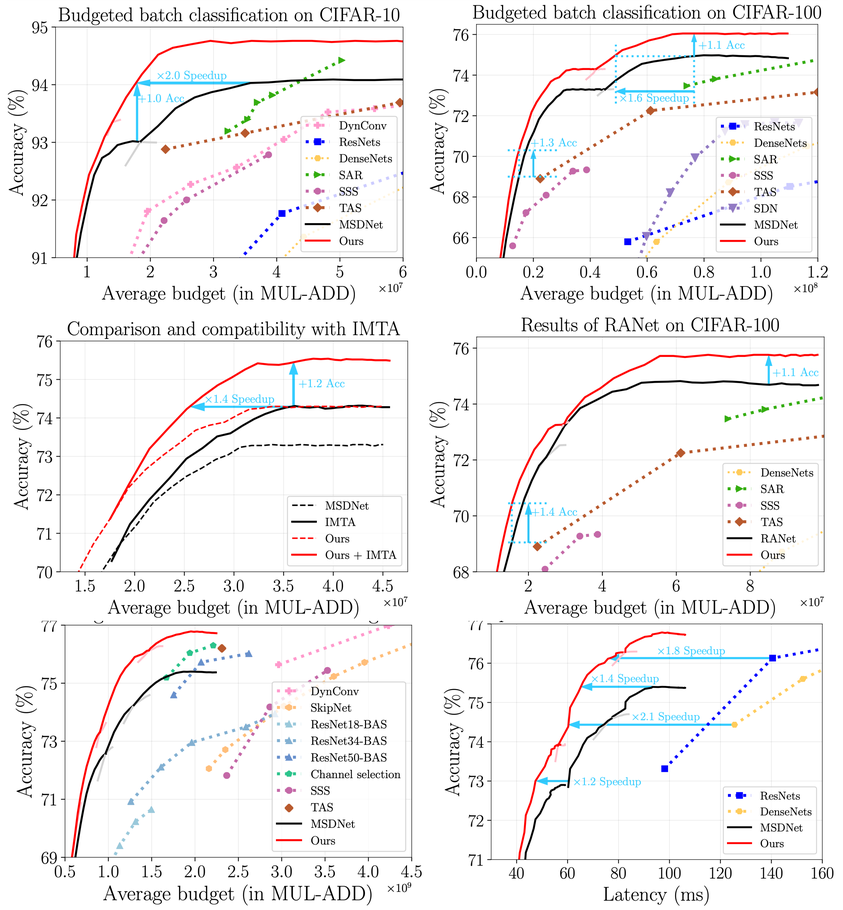

## Experiments

### Ablation Studies




- Sample **weighting** > Sample selection ;
- **Predicted** weights > Hand-designed weights ;
- **Meta-learning** procedure > Frozen (learned) weight prediction;
- Our specialized **meta objective** > naïve meta objective;
- Using **loss** to produce weights is preferable to using confidence.

### Main Results




- Significantly boost the performance;
- Effective on different multi-exit structures;
- Compatible with other training techniques.

Paper

Code