Long-tailed Recognition by Routing Diverse Distribution-Aware Experts

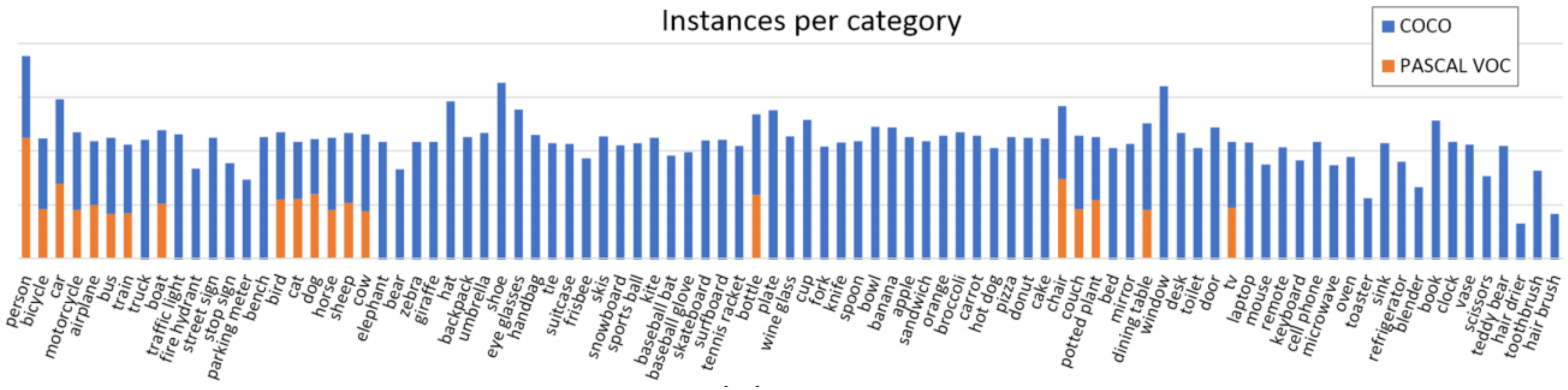
ICLR 2021

Catalogue

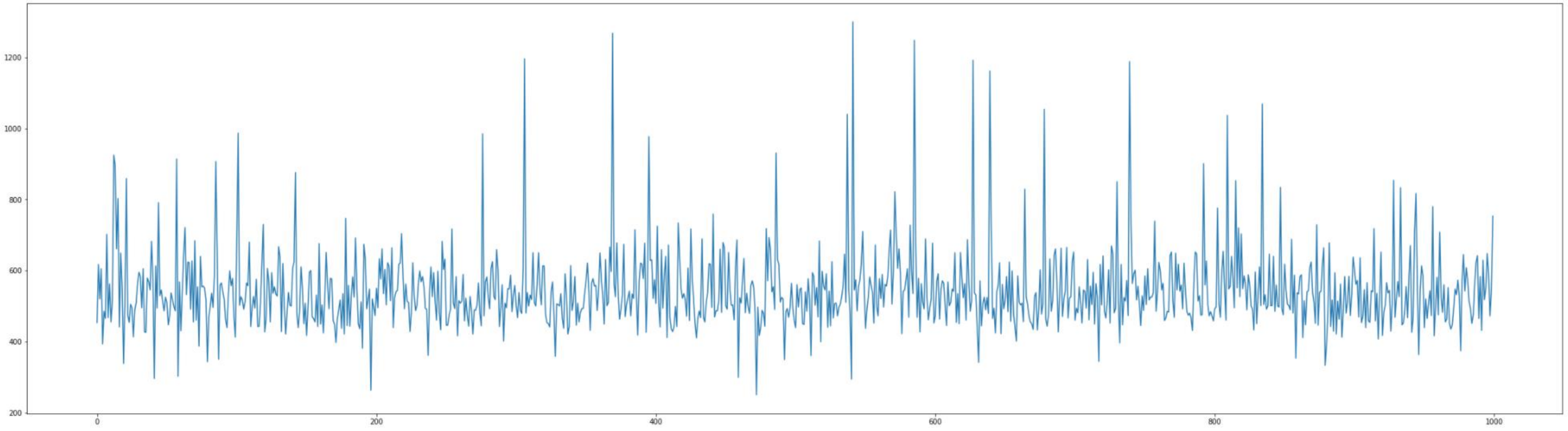
- Background
- Related work
- Introduction
- Method
- Conclusion
- My opinions & questions

Background

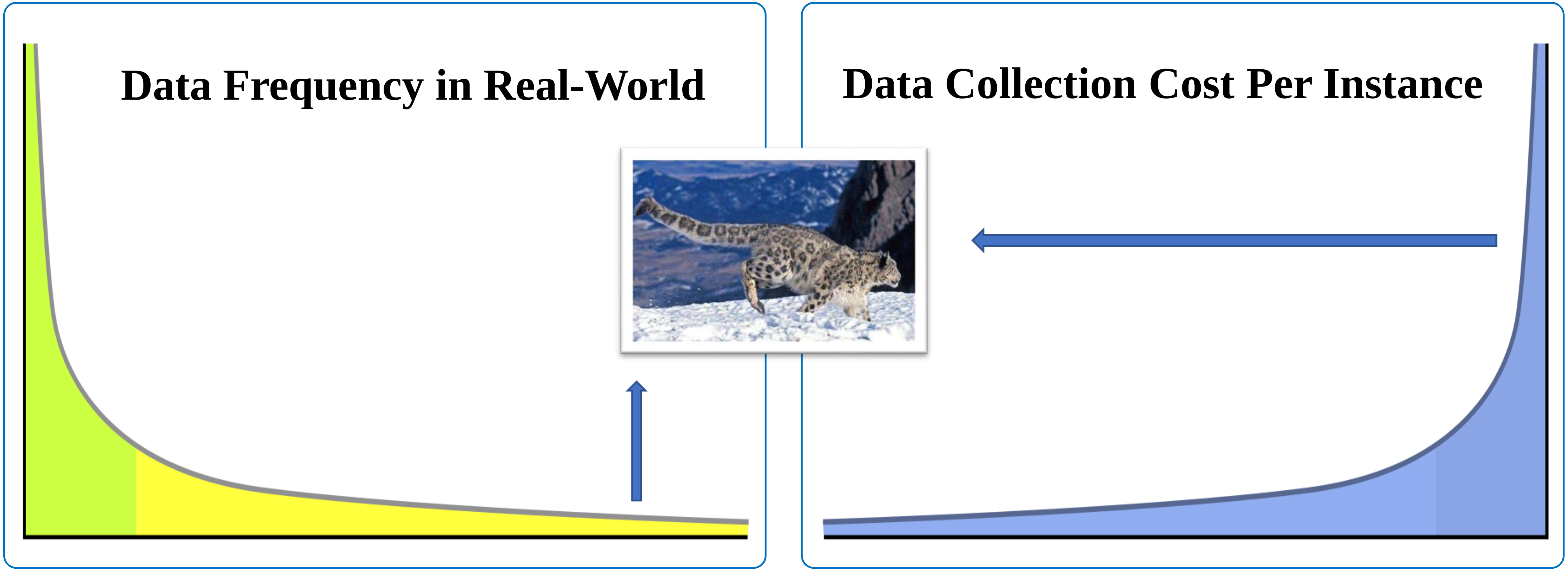
MS-COCO (Object Detection & Instance Segmentation)



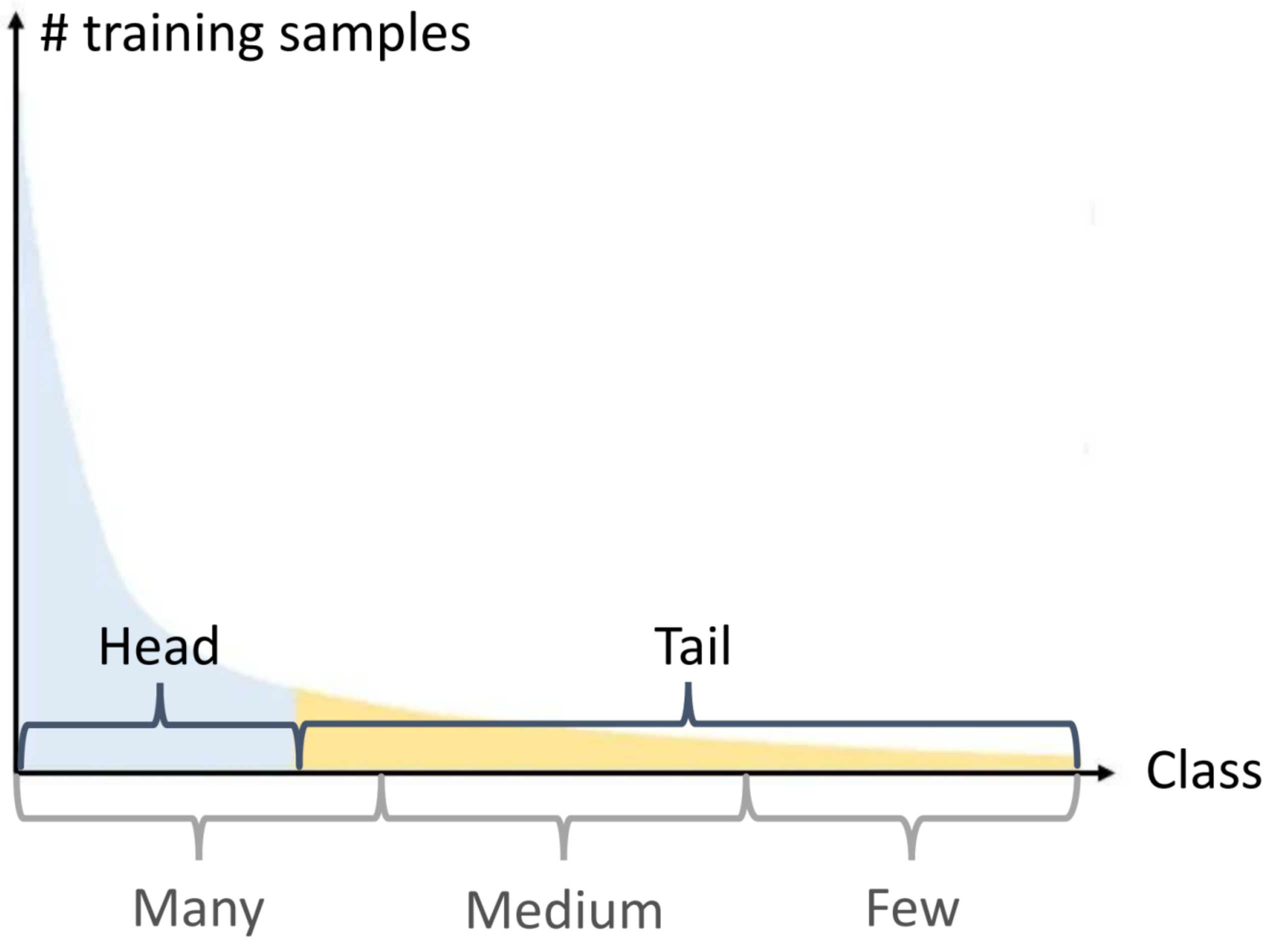
ImageNet (Image Classification)



Background


Data Frequency in Real-World
Data Collection Cost Per Instance

Background



Training set
Long-tailed distribution

Testing set
Balanced distribution

Evaluation set
Overall testing set
Three subsets of testing set(many, medium, few)

Related work

Classical Rebalanced learning

- Re-sampling(under-sampling, over-sampling)
- Re-weighting(Focal loss, CB loss…)

Feature Enhancement
(including Knowledge transfer)

- Transfer learning(OLTR)
- Domain adaption, Synthetic samples
- Semi-supervised learning, self-supervised

Two-stage Rebalanced learning

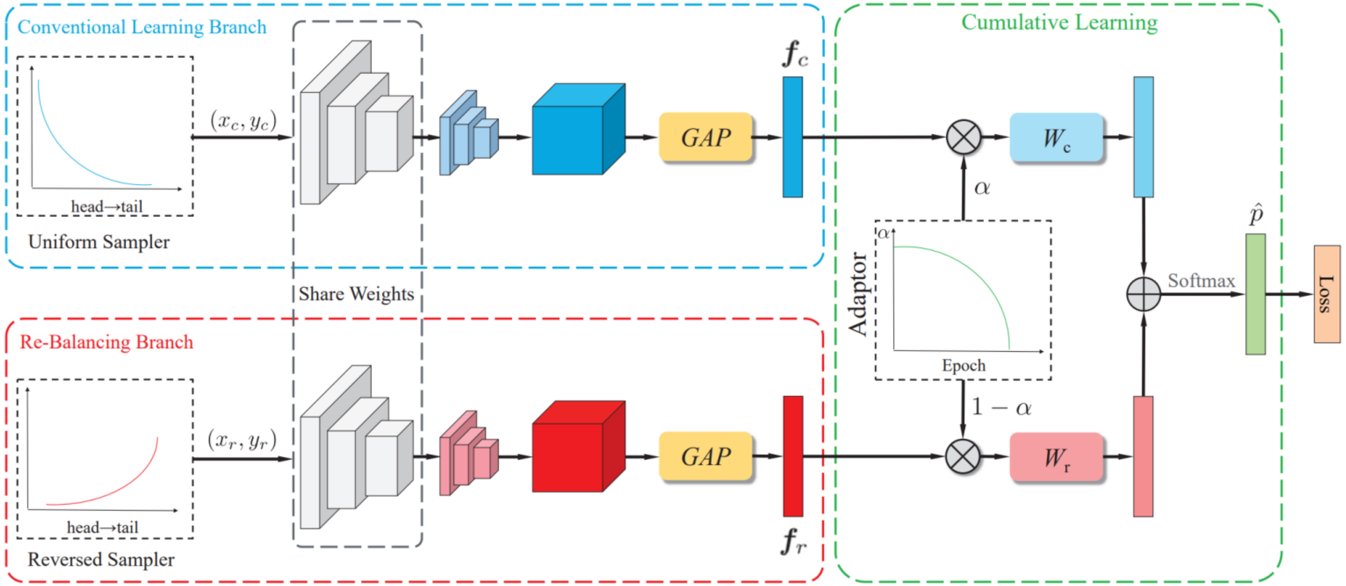
- BBN, Decoupling representation & classifier

Defects

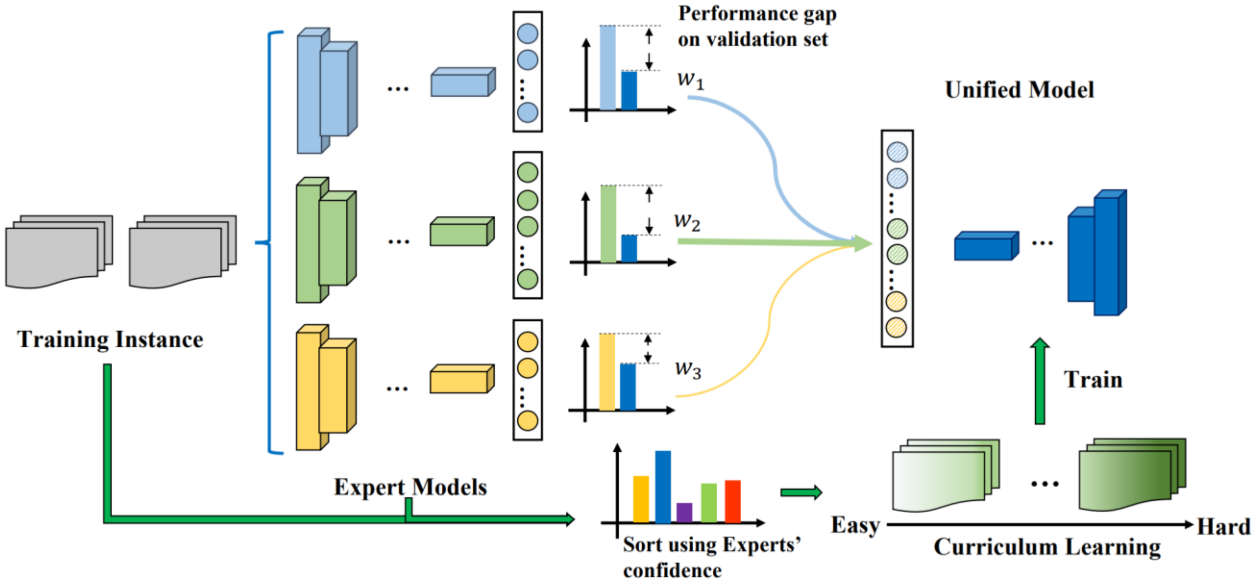
All these methods generally gain accuracy on tail classes at the cost of performance loss on head classes.

Related work-Ensemble and grouping

- BBN



- LFME



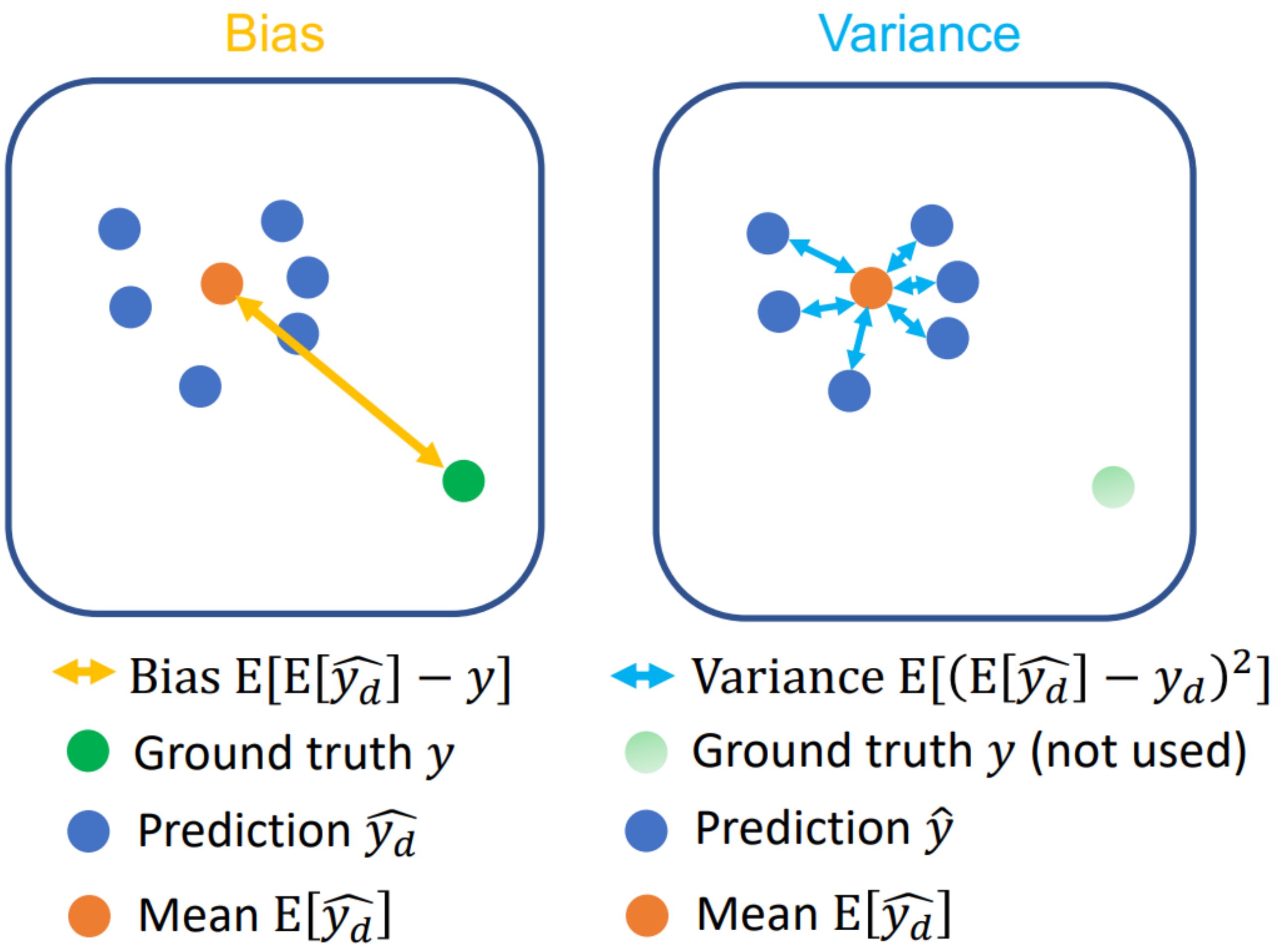
Similarity

Each expert do not have a balanced access to the whole dataset. Thus, they damage the overall generalizability, especially head classes.

Introduction - Motivation

Bias-variance decomposition: $\mathrm{Error}(f;D) = \mathrm{Bias}^2(x) + \mathrm{Variance}(x) + \varepsilon^2$

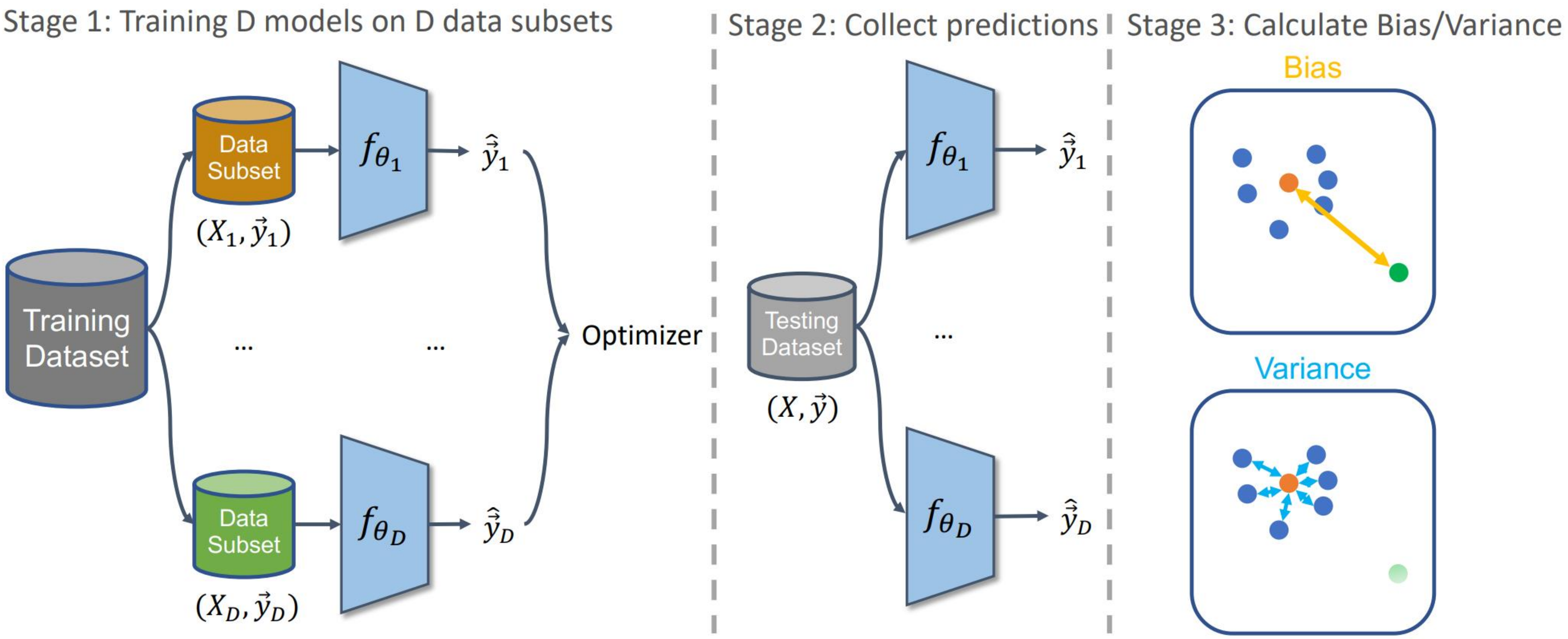
The bias measures the true accuracy of prediction



The var measures the stability of prediction

Introduction - Motivation

How to estimate the bias and variance of the classifiers in classification?



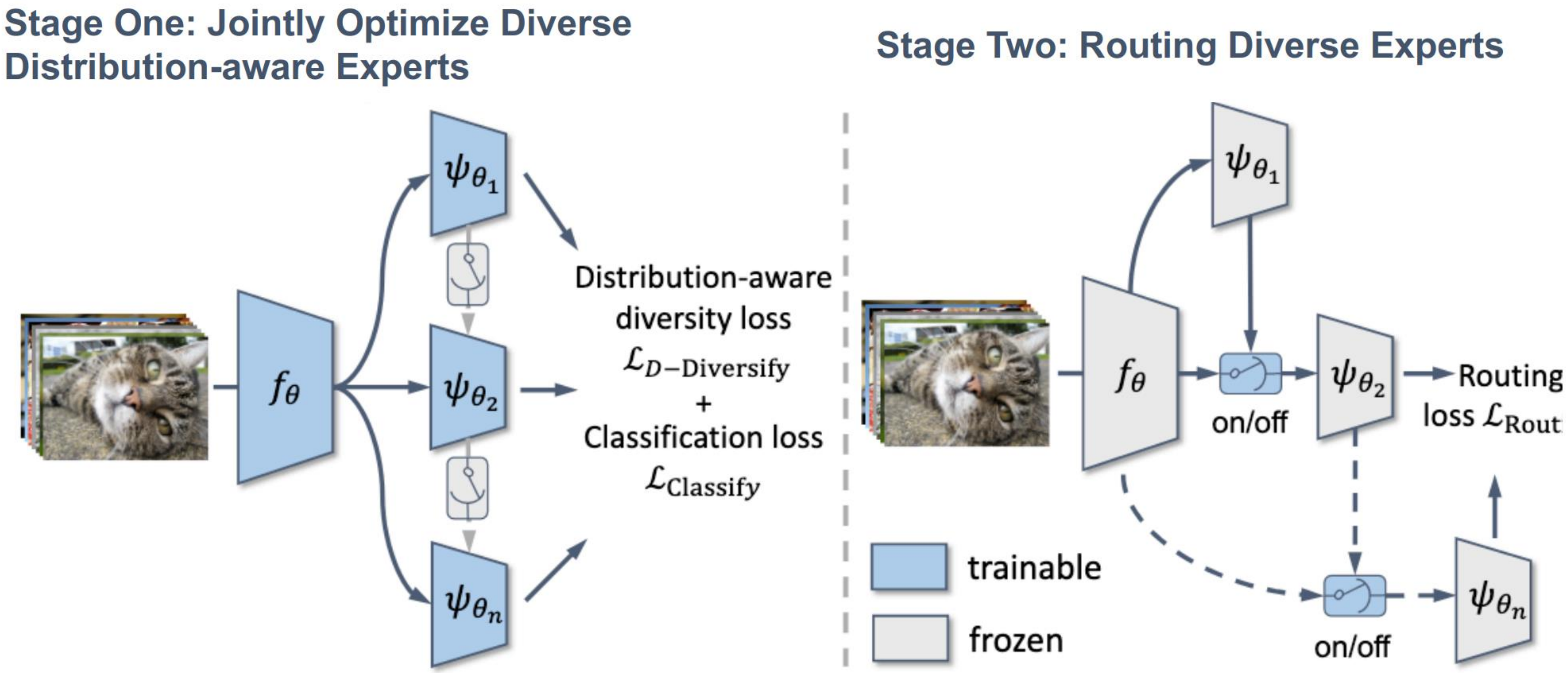
Introduction - Motivation

	Head Classes			Tail Classes		
	Acc	Bias	Variance	Acc	Bias	Variance
Current SOTAs	Worse	Comparable	Worse	Better	Better	Worse

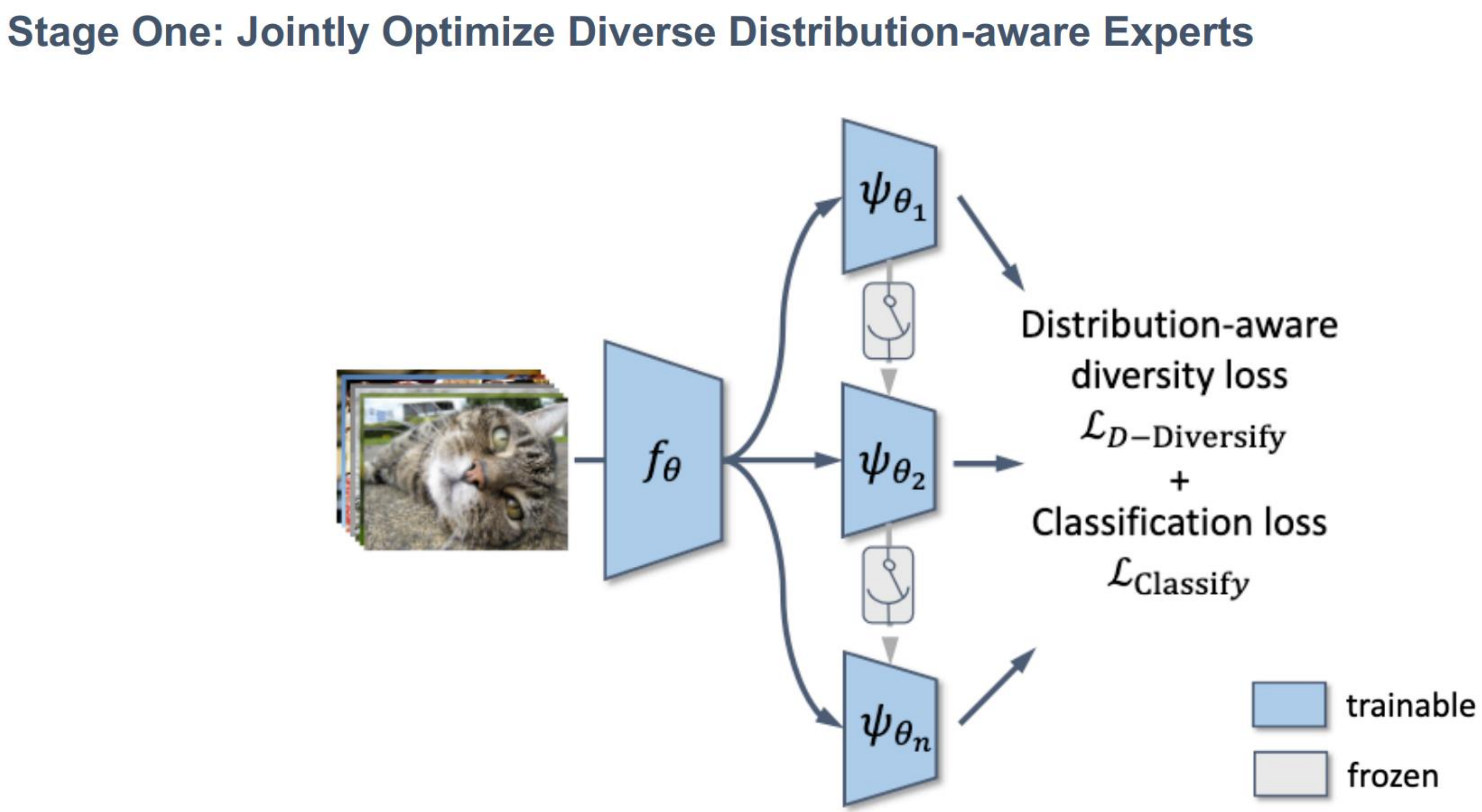
Notes:

- The increased variance leads to a worse bias-variance trade-off.
- Thus, we should further reducing variance and bias, especially the variance.

Method - Overall model



Method - Stage 1



Method - Stage 1 - framework

The shared part:

f_θ

(which is considered task-agnostic)

The independent part:

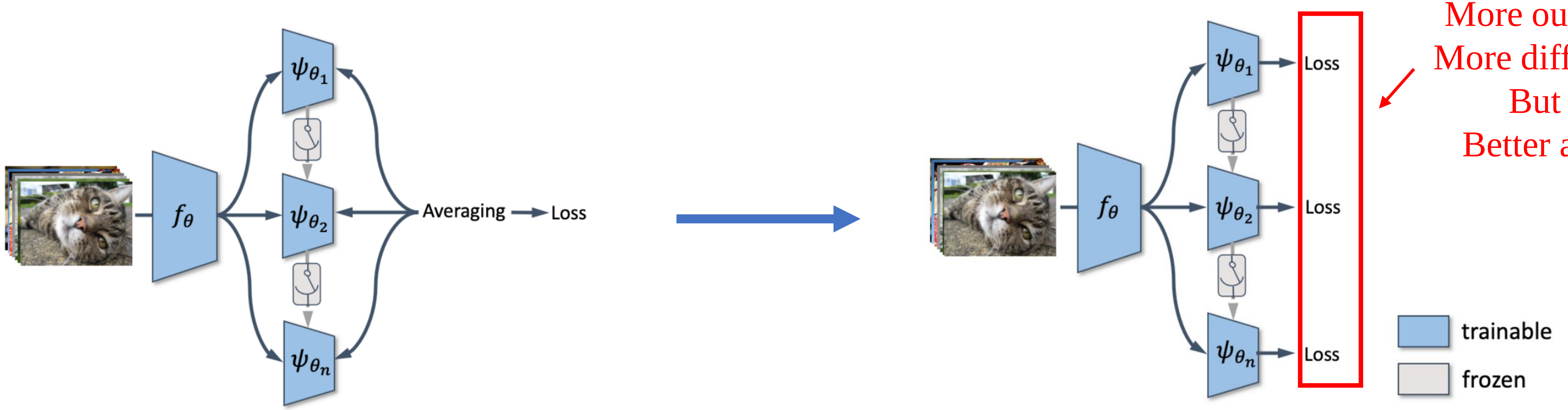
φ_{θ_i}

(the number of filters in φ_{θ_i} is reduced by 1/4)

```
    cumulative_sample_num_experts[num - 1] += count
RuntimeError: expected device cuda:0 and dtype Float but got device cuda:0 and dtype Long
(LS) mk@mk-Z10PE-D8-WS-Invalid-entry-length-16-Fixed-up-to-11:~/ZZC/Long-Tailed/RIDE-LongTailRecognition-main$ CUDA_
d/models/Imbalance_CIFAR100_LT_RIDE/0207_153507/model_best.pth" --reduce_dimension 1 --num_experts 4
Files already downloaded and verified
Files already downloaded and verified
ResNet32EAModel(
  (backbone): ResNet_s(
    (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (layer1): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential()
      )
      (1): BasicBlock(
        (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential()
      )
      (2): BasicBlock(
        (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential()
      )
      (3): BasicBlock(
        (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential()
      )
      (4): BasicBlock(
        (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (shortcut): Sequential()
      )
    )
    (layer2s): ModuleList(
      (0): Sequential(
        (0): BasicBlock(
          (conv1): Conv2d(16, 24, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
          (bn1): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (conv2): Conv2d(24, 24, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
          (bn2): BatchNorm2d(24, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (shortcut): LambdaLayer()
        )
```

32

Method - Stage 1 – how to decouple different experts model



More output
More difficult
But
Better acc

Collaborative loss leads to correlated experts ✗

Individual loss decorrelates experts ✓

$$\mathcal{L}^{\text{collaborative}} = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(\boxed{\frac{1}{n}}\sum_{j=1}^{n} \psi_{\theta_j}(f_\theta(\vec{x_i})), y_i\right)$$

$$\mathcal{L}^{\text{individual}} = \frac{1}{\boxed{n}N}\sum_{i=1}^{N}\sum_{j=1}^{n} \mathcal{L}_i(\psi_{\theta_j}(f_\theta(\vec{x_i})), y_i)$$

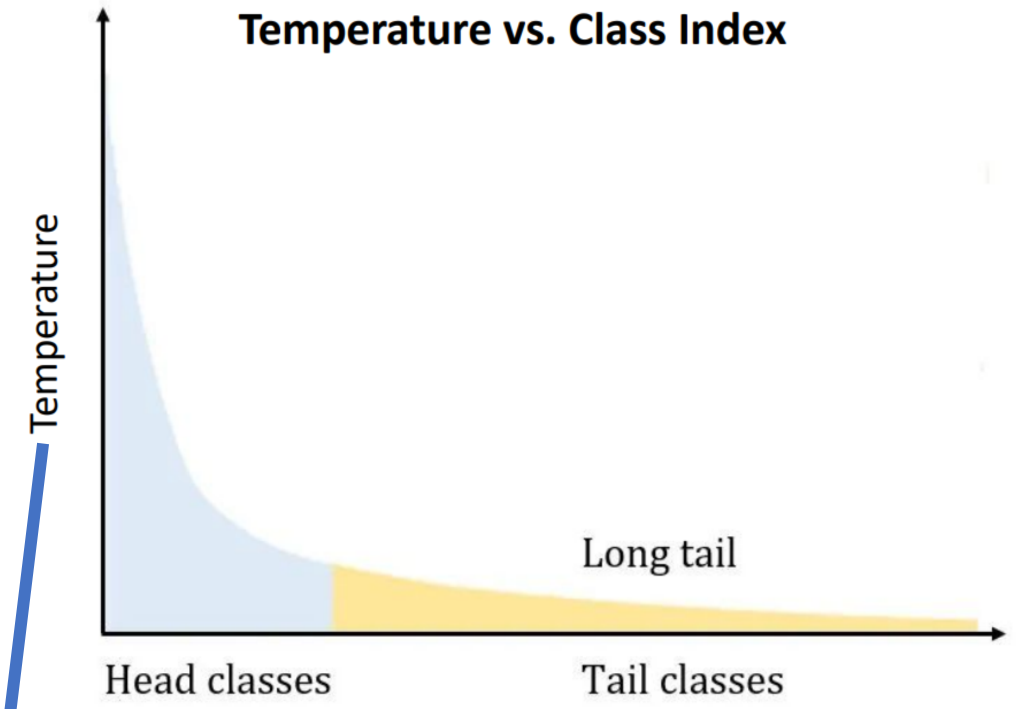
Method - Stage 1 – how to decouple different experts model

The distribution-aware diversity loss is proposed to penalize the inter-expert correlation, formulated as:

$$\mathcal{L}^i_{\text{D-Diversify}} = -\frac{\lambda}{k-1}\sum_{j\neq i}^{n} \boxed{\mathcal{D}_{KL}}(\phi^i(\vec{x},\vec{T}), \boxed{\phi^j(\vec{x},\vec{T})})$$

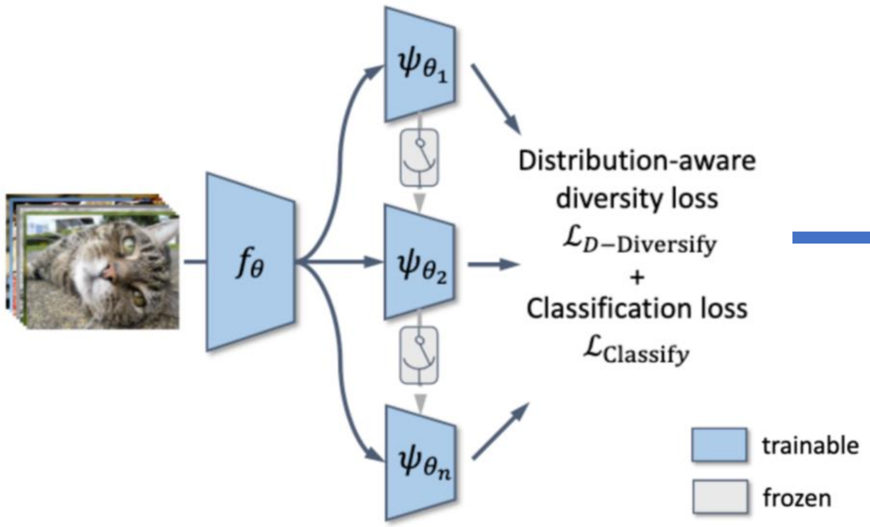
KL divergence   Softmax with temperature

$$\text{where } \phi^i(\vec{x},\vec{T}) = \text{softmax}(\psi_{\theta_i}(f_\theta(\vec{x}))/\boxed{\vec{T}})$$



Note:
The idea of temperature in contrastive loss (Hadsell et al., 2006; Wu et al., 2018) to enable the diversity loss to be distribution aware

Method - Stage 1 – Total loss



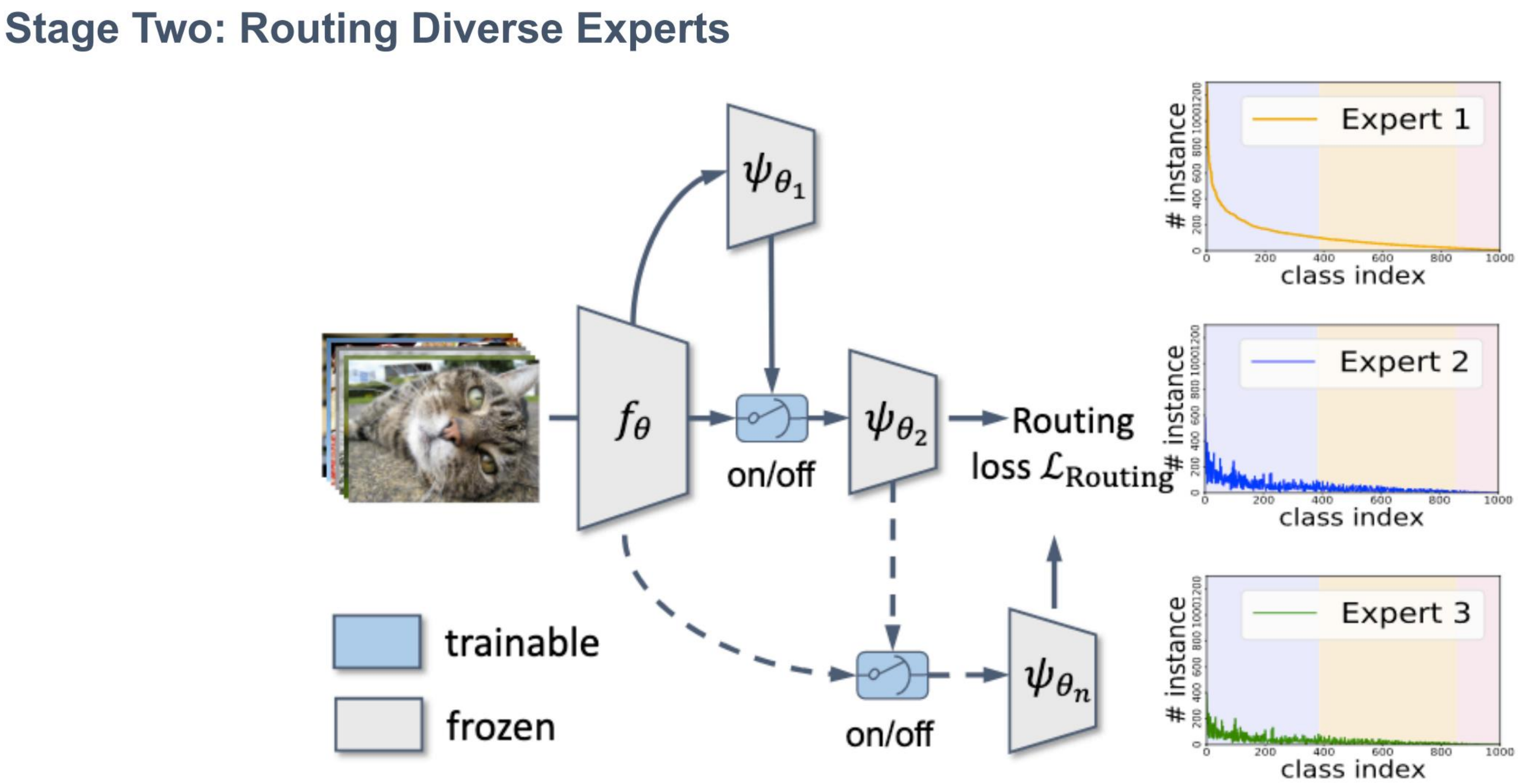
$$\mathcal{L}_{\text{Total}} = \frac{1}{nN}\sum_{i=1}^{N}\sum_{j=1}^{n}\mathcal{L}^{j}_{\text{Classify}}(\phi^j(\vec{x}), y) - \frac{\lambda}{n-1}\sum_{j\neq k}^{n} D_{\text{KL}}(\phi^j(\vec{x}, \vec{T}), \phi^k(\vec{x}, \vec{T}))$$

where j and k are the expert indices, L j Classify(., .) can be LDAM loss, focal loss, etc., depending on the training mechanisms we choose.

Method - Stage 1 – Code results(4 experts)

```
Train Epoch: 199 [0/85 (0%)] Loss: 26.500248 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 199 [11/85 (13%)] Loss: 25.753616 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 199 [22/85 (26%)] Loss: 23.234282 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 199 [33/85 (39%)] Loss: 17.670929 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 199 [44/85 (52%)] Loss: 33.627335 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 199 [55/85 (65%)] Loss: 21.283491 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 199 [66/85 (78%)] Loss: 21.298191 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 199 [77/85 (91%)] Loss: 16.154081 max group LR: 0.0000 min group LR: 0.0000
    epoch           : 199
    loss            : 23.570067484238567
    accuracy        : 0.874804093297686
    val_loss        : 2.504270945923238
    val_accuracy    : 0.4916
Saving current best: saved/models/Imbalance_CIFAR100_LT_RIDE/0207_153507/model_best.pth ...
Train Epoch: 200 [0/85 (0%)] Loss: 16.164438 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 200 [11/85 (13%)] Loss: 22.286634 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 200 [22/85 (26%)] Loss: 16.666439 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 200 [33/85 (39%)] Loss: 29.576056 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 200 [44/85 (52%)] Loss: 20.979742 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 200 [55/85 (65%)] Loss: 18.791460 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 200 [66/85 (78%)] Loss: 22.410103 max group LR: 0.0000 min group LR: 0.0000
Train Epoch: 200 [77/85 (91%)] Loss: 21.128582 max group LR: 0.0000 min group LR: 0.0000
    epoch           : 200
    loss            : 23.535649265962487
    accuracy        : 0.87249930856458
    val_loss        : 2.490911396243904
    val_accuracy    : 0.491
(LS) mk@mk-Z10PE-D8-WS-Invalid-entry-length-16-Fixed-up-to-11:~/ZZC/Long-Tailed/RIDE-LongTailRecognition-main$ CUDA_VISIBLE_D
```

Method - Stage 2



Note: The data imbalance ratio for later experts can be automatically reduced without any distribution-aware loss.

Method - Stage 2 - loss

$$\mathcal{L}_{\text{Routing}} = -\omega_{\text{p}} \boxed{y} \log\left(\frac{1}{1+e^{-y_{\text{ea}}}}\right) - \omega_{\text{n}}(1-\boxed{y}) \log\left(1-\frac{1}{1+e^{-y_{\text{ea}}}}\right)$$

where the ground truth y is constructed as: if the current expert does not predict the sample correctly but one of the next experts gives correct prediction, the ground truth is set to 1 (considered as a positive sample), otherwise it is 0.

and y_{ea} is calculated as (v_i is the output of expert i, and l_i is part of v_i):

$$y_{\text{ea}} = \mathbf{W}_2(\vec{l_i} \oplus \sigma(\mathbf{W}_1 \vec{v_i}))$$

Method - Stage 1 – Code results(4 experts)

```
expert (3): 0.37890625
expert (1): 0.5625
expert (2): 0.482421875
expert (3): 0.4375
expert (1): 0.58203125
expert (2): 0.494140625
expert (3): 0.435546875
expert (1): 0.6015625
expert (2): 0.49609375
expert (3): 0.453125
expert (1): 0.5859375
expert (2): 0.505859375
expert (3): 0.470703125
expert (1): 0.58203125
expert (2): 0.482421875
expert (3): 0.41796875
expert (1): 0.58984375
expert (2): 0.46484375
expert (3): 0.41015625
expert (1): 0.548828125
expert (2): 0.453125
expert (3): 0.43359375
expert (1): 0.6102941176470589
expert (2): 0.5183823529411765
expert (3): 0.46691176470588236
Samples with num_experts: 42.24 9.83 4.75 43.18
    epoch          : 5
    loss           : 5.290547425096685
    accuracy       : 0.8905688208721305
    top_k_acc      : 0.9758458559970499
    val_loss       : 2.991361844539642
    val_accuracy   : 0.483
    val_top_k_acc  : 0.7486
Saving checkpoint: saved/models/Imbalance_CIFAR100_LT_RIDE/0207_170337/checkpoint-epoch5.pth ...
(LS) mk@mk-Z10PE-D8-WS-Invalid-entry-length-16-Fixed-up-to-11:~/ZZC/Long-Tailed/RIDE-LongTailRecognition-ma
```

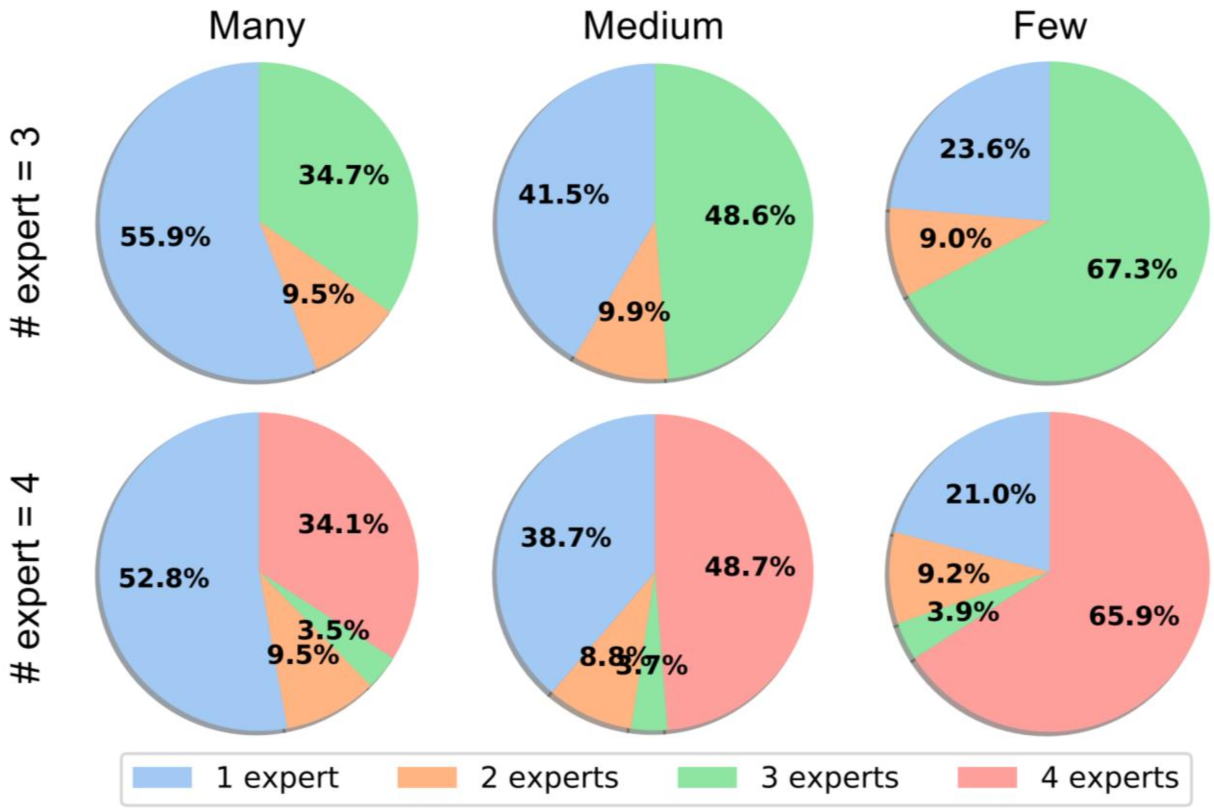
Experiments – Impact of RIDE

	Head Classes			Tail Classes		
	Acc	Bias	Variance	Acc	Bias	Variance
Current SOTAs	Worse	Comparable	Worse	Better	Better	Worse
RIDE	Better	Better	Better	Better	Better	Better

Notes:

- Better accuracy for all splits including the head classes(the many shot subset) when using RIDE.
- Better bias-variance trade-off for all splits when using RIDE.

Experiments – Impact of RIDE



Notes:

- Most instances in the many-shot classes are assigned only 1 expert.
- Those instances in few-shot classes are often assigned more experts.

Introduction - Experiments on ImageNet-LT

Methods	ResNet-50 GFlops	ResNet-50 Acc. (%)	ResNeXt-50 GFlops	ResNeXt-50 Acc. (%)
Cross Entropy (CE) †	4.11 (1.0x)	41.6	4.26 (1.0x)	44.4
OLTR † (Liu et al., 2019)	-	-	-	46.3
NCM (Kang et al., 2020)	4.11 (1.0x)	44.3	4.26 (1.0x)	47.3
τ-norm (Kang et al., 2020)	4.11 (1.0x)	46.7	4.26 (1.0x)	49.4
cRT (Kang et al., 2020)	4.11 (1.0x)	47.3	4.26 (1.0x)	49.6
LWS (Kang et al., 2020)	4.11 (1.0x)	47.7	4.26 (1.0x)	49.9
RIDE (2 experts)	**3.71 (0.9x)**	54.4 **(+6.7)**	**3.92 (0.9x)**	55.9 **(+6.0)**
RIDE (3 experts)	4.36 (1.1x)	54.9 **(+7.2)**	4.69 (1.1x)	56.4 **(+6.5)**
RIDE (4 experts)	5.15 (1.3x)	**55.4** **(+7.7)**	5.19 (1.2x)	**56.8** **(+6.9)**

Conclusion & My opinion & questions about the paper

Conclusion:

- The paper proposed a new way of using the multi-expert which needs decoupling these experts. Specifically, the paper designs two novel loss function to achieve it.
- The paper proposed a dynamic expert routing module which is more effective and flexible.
- It effectively reduce the computational complexity of our multi-expert model to a level even lower than a baseline model with the same backbone.

Opinion:

It's really good that the paper focus on the realistic problem of the drop of many shots' acc while the few shots' acc is improving and rethink the problem through a easy but classical bias-variance decomposition .

The idea of multi-expert has been proposed in early paper, but it's novel to feed these experts model with the same original data distribution and decouple these experts. I think it's the core idea of the paper.

My questions about the paper

Question:

- What if we use the attention mechanism of transformer in the dynamic regulation part of this model, such as the second stage and the control of the capacity of the backbone in stage 1 in case of over-fitting in tail class.
- It's confusing that the results of Stage 2 is a little(about 1%) worse than Stage 1, could it be that the stage 2 decreases the complexity of the model at the expense of the drop of the acc.

Thanks for listening.