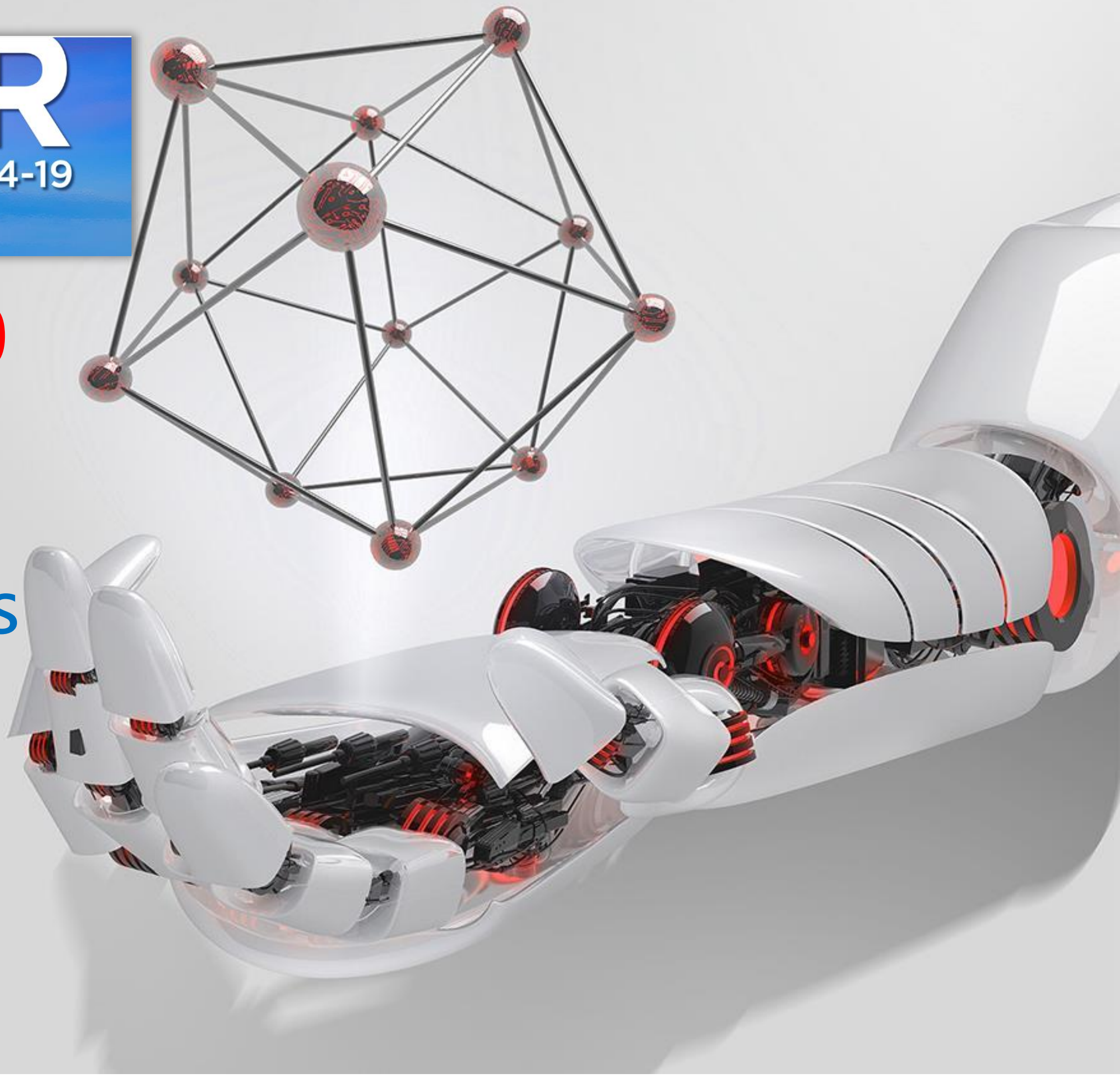
Our Solution @ WebVision 2020

An Effective Approach for Learning from Large-scale Web Images



Speaker: Lingxi Xie

The *Smart_Image* Team, Huawei Inc.

The *Smart_Image* Team

- A combined team* from *Huawei Noah's Ark Lab* and *Huawei Cloud EI*
  - **Model training:** Zewei Du, Bincheng Liu, Longhui Wei, Zhao Yang
  - **Model ensemble:** Hang Chen, Yaxiong Chi
  - **Technical support:** Zhengsu Chen, Jianzhong He
  - **Overall schedule:** Lingxi Xie, Xiaopeng Zhang
  - **Computational resource:** Xiaolong Bai
  - **Team organization:** Hongjie Si, Qi Tian

* People in each group are sorted in the alphabetical order.

Outline

- An overview of the WebVision 2020 challenge
- Our solution: learning, mining, and fusion
  - **Learning:** the selection of network backbones
  - **Mining:** playing with the noisy dataset
  - **Fusion:** a community works better
- Failure trials also deliver knowledge
- Summary and conclusions

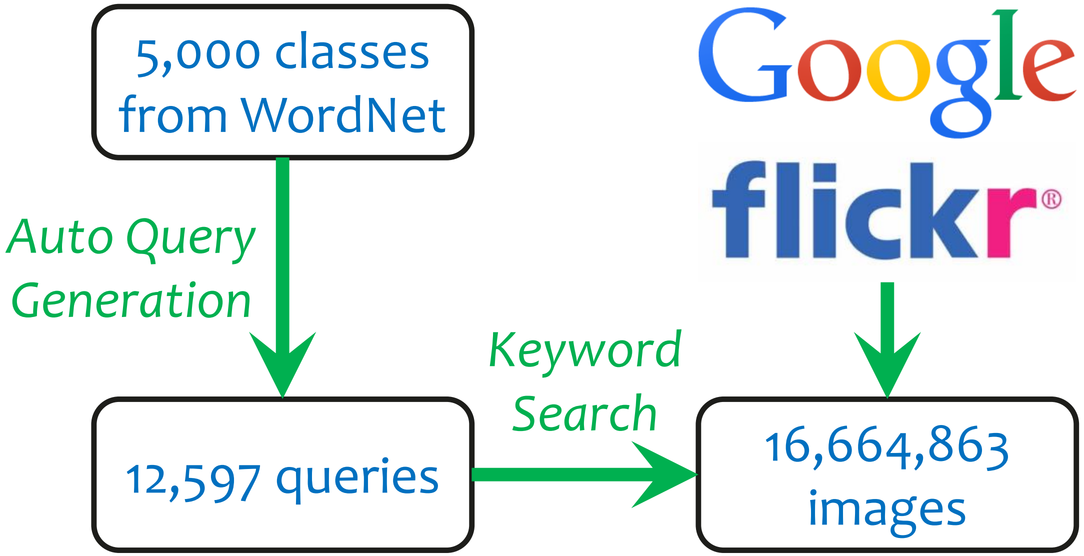
Outline

- **An overview of the WebVision 2020 challenge**
- Our solution: learning, mining, and fusion
  - **Learning:** the selection of network backbones
  - **Mining:** playing with the noisy dataset
  - **Fusion:** a community works better
- Failure trials also deliver knowledge
- Summary and conclusions

HUAWEI

Overview: WebVision 2020 Challenge

- The WebVision 2.0 dataset
  - 5K classes, covering coarse and fine classes
  - Crawled from the Web using text queries
- The challenge
  - Training data: ~16M (with duplicate)
  - Validation data: ~300K (relatively clean)
  - Test data: ~300K (labels are unavailable)
  - Top-5 accuracy, class-level average



Dataset		Images
Training	Google	8,366,429
	Flickr	7,710,236
Validation		294,099
Testing		294,099

W. Li *et al.*, WebVision Database: Visual Learning and Understanding from Web Data, *arXiv preprint: 1708.02862*, 2017.

HUAWEI

The Real Challenges!

- The dataset has 5,000 object categories
  - There exist a lot of fine-grained concepts that are difficult to recognize
  - There also exist some abstract concepts that are **almost impossible** to learn

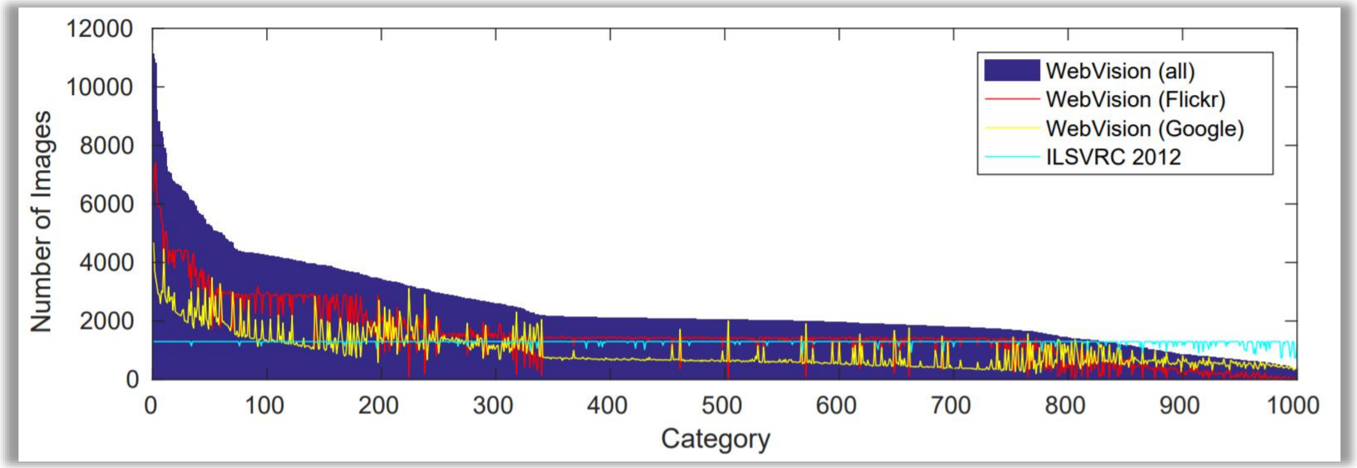
#3005: common+man, commoner, common+person

#2070: peak+limit, extremum+limitation, ...

#1476: life+soul, life+person, life+normal, life+someone, ...

The Real Challenges!

- The training data has 5,000 object categories
  - There exist a lot of fine-grained concepts that are difficult to recognize
  - There also exist some abstract concepts that are almost **impossible** to learn
- Training data distribution and noise
  - But, the difficulty of a class is not necessarily related to the number of training images of the class (*e.g.* some abstract classes may have a lot of noisy data)
  - The noise may come from different aspects (*e.g.* wrong labels, ambiguity, *etc.*)



Outline

- An overview of the WebVision 2020 challenge
- **Our solution: learning, mining, and fusion**
  - **Learning:** the selection of network backbones
  - **Mining:** playing with the noisy dataset
  - **Fusion:** a community works better
- Failure trials also deliver knowledge
- Summary and conclusions

Learning: Which Backbones Are Effective?

- We used ResNet-based backbones
  - ResNet-50/101/152: the original networks with different depths
  - ResNeXt-152: the network with group convolutions
  - ResNeSt-269: adding the split-attention modules
  - Other combinations: SE-ResNet-154, SE-ResNeXt-152, *etc.*
- We tried EfficientNet-B0/B4 but decided not to use them
  - EfficientNet-based models converge much slower
  - EMA is very important for improving single-model performance for EfficientNet-based models, but using EMA may harm the performance of model ensemble

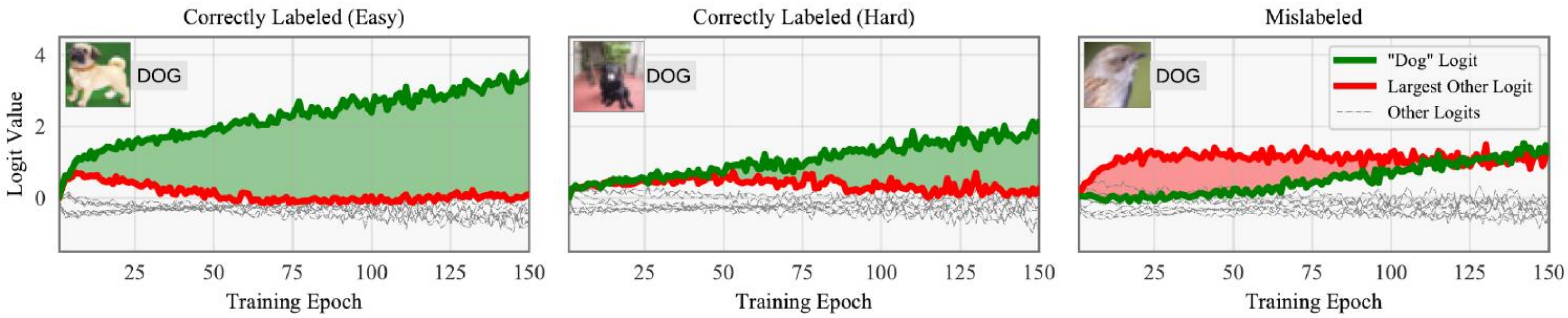
K. He *et al.*, Deep Residual Learning for Image Recognition, *CVPR*, 2016.
S. Xie *et al.*, Aggregated Residual Transformations for Deep Neural Networks, *CVPR*, 2017.
J. Hu *et al.*, Squeeze-and-Excitation Networks, *CVPR*, 2018.
M. Tan *et al.*, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, *ICML*, 2019.
H. Zhang *et al.*, ResNeSt: Split-Attention Networks, *arXiv preprint: 2004.08955*, 2020.

Bag-of-Tricks of Network Training

- We applied Huawei’s ModelArts for large-scale distributed training
  - For an introduction to ModelArts, please visit:
    https://support.huaweicloud.com/en-us/productdesc-modelarts/modelarts_01_0001.html
- Tricks for network training and testing
  - We used RandAugment to alleviate over-fitting
  - We applied class-level sampling balance (using 3,600 training images for each class, if not enough, then duplicating some images)
  - We tuned the starting learning rate carefully
  - We used multi-scale training
  - We used multi-scale, multi-crop testing

Mining: Filtering out Noise in the Dataset

- We used AUM to measure the cleanness of each training image
  - After a complete training process, each image is assigned a value of AUM, which can be used to determine if this image will be used in the next round
  - We filtered 20% of training images with lowest AUM values
  - Typically, AUM can improve the class-averaged top-5 accuracy by ~0.5%
- We also tried knowledge distillation, but observed little accuracy gain



G. Pleiss *et al.*, Identifying Mislabeled Data using the Area Under the Margin Ranking, *arXiv preprint:* 2001.10528, 2020.

HUAWEI

Fusion: Ensemble with 100+ Results!

- Average fusion works sufficiently well
    - We have trained more than 50 models: different backbones, different training strategies (*e.g.* AUM on/off), different input sizes, *etc.*
    - Some models contributed single-crop, 5-crop, and 10-crop results individually
    - We have fused 128 results: the more we used, the better results we obtained
    - The best single model reports ~81% accuracy, but even so, adding some weak models with ~70% accuracy can improve fusion performance
- We have adjusted model-wise weights to boost stronger models
    - A standard genetic algorithm with crossover and mutation
    - This slightly improves accuracy (~0.1%) on both validation and test sets

Results and Our Submission

- Single model (take SE-ResNet-154 as an example)
  - The baseline top-5 validation accuracy is ~79.8%
  - After AUM is applied, the top-5 validation accuracy is boosted to ~80.5%
  - After KD is applied, the top-5 validation accuracy is boosted to ~80.7%
  - Many others, ignored here
- Model ensemble
  - A simple ensemble with score average reports 82.80%
  - When the genetic algorithm is used, the accuracy is boosted to 82.94%
  - **The 82.94% method reports 82.97% on the test set**

Outline

- An overview of the WebVision 2020 challenge
- Our solution: learning, mining, and fusion
  - **Learning:** the selection of network backbones
  - **Mining:** playing with the noisy dataset
  - **Fusion:** a community works better
- **Failure trials also deliver knowledge**
- Summary and conclusions

Lessons Learned from Failure Trials

The reason for the following observations remains mostly unclear

- Powerful architectures on ImageNet do not work well on WebVision
  - EfficientNet-B4 was just a little bit stronger to ResNet-50
  - ResNeSt-269 did not show great advantage over SE-ResNeXt-154
- Data mining methods do not improve upon the high baseline
  - Reducing the weight of 70%-90% AUM-ranked data produced worse results
  - Fine-tuning with training data of the worst 500 classes and then fusing the fine-tuned model with the original model did not improve overall accuracy
- Advanced ensemble does not bring much gain beyond a naïve average
  - We noticed that the score distribution of different models vary a lot

Outline

- An overview of the WebVision 2020 challenge
- Our solution: learning, mining, and fusion
    - **Learning:** the selection of network backbones
    - **Mining:** playing with the noisy dataset
    - **Fusion:** a community works better
- Failure trials also deliver knowledge
- **Summary and conclusions**

Summary and Conclusions

- What have we done?
  - WebVision 2020: 5K classes with 16M **noisy** training images
  - A top-5 accuracy of **82.97%**, advancing the previous state-of-the-art
- What have we learned from the challenge?
  - In a large-scale dataset, tricks obtained on small datasets might not work
  - There are different types of noise, so aggressive filtering might not work well
- What shall we do in the future?
  - Exploring the solution of using stronger backbones
  - Diagnosing the noise and looking for a better way to alleviate it
  - Building an automatic flowchart for learning from a large-scale, noisy dataset

Thanks!

- Questions, please?