Window-Based Early-Exit Cascades for Uncertainty Estimation
Supplementary Material

Guoxuan Xia
Imperial College London
g.xia21@imperial.ac.uk

Christos-Savvas Bouganis
Imperial College London
christos-savvas.bouganis@imperial.ac.uk

1. Additional Setup Details

Training. We train two families of computationally efficient CNNs for image classification on ImageNet-1k [11]: EfficientNet [13] and MobileNet-V2 [12]. For EfficientNet, we scale width, depth and resolution as the original authors [13] from B0→B4. For MobileNet-V2 we use the scalings in Tab. 1, which are taken from the Keras github repository.[1] For each model we train a Deep Ensemble size $M = 2$ using random seeds $\{1,2\}$ (everything is the same between ensemble members other than the random seed). This results in 10 individual ensembles, 5 composed of EfficientNets and 5 composed of MobileNet-V2s.

input resolution	160	192	224	224	224
width factor	1.0	1.0	1.0	1.3	1.4

Table 1. MobileNet-V2 scaling used in experiments.

We train all models for 250 epochs on ImageNet-1k, and hold out a random subset of 50,000 images from the training set for validation (we then evaluate on the original validation set). We train using standard cross entropy. We use stochastic gradient descent with a learning rate of 0.2, weight decay of 4e-5 and a batch size of 1024 for all models other than EfficientNet-B4, for which we use a learning rate of 0.1 and batch size of 512 due to GPU memory constraints (following the scaling recommendations in [2]). We use cosine learning rate decay with a 5 epoch linear warmup.[2] We use default random resize-cropping and random horizontal flipping for data augmentation, and images are scaled using bicubic interpolation.

We train all of our models using PyTorch [9] and Lightning [1] distributed over 8 NVIDIA V100 32GB GPUs using Automatic Mixed Precision.[3] Training and evaluation code can be found here: https://github.com/Guoxoug/window-early-exit. Please follow the instructions in the `README.md` file in order to reproduce our results and plots.

[1] https://github.com/keras-team/keras/blob/master/keras/applications/mobilenet_v2.py.

[2] This is a combination of the hyperparameters from https://github.com/d-li14/mobilenetv2.pytorch and the scaling approaches recommended in [2].

[3] https://developer.nvidia.com/automatic-mixed-precision

Setting windows. In general, we find τ on the validation set and then vary $[t_1,t_2]$ by placing them at increasing symmetric percentiles on either side of τ. If either side of the window hits either the zeroth or 100th percentile, then the expansion will only apply to the other side, *e.g.* if τ is set at TPR=95% then there is only room for 5% (of ID data) on the side more uncertain than τ. As mentioned previously, we leave further optimisation of $[t_1,t_2]$ to future work.

Packed Ensembles. For the experiments involving Packed Ensembles we use the same ResNet-50 models as those in Tab. 2 of [6], with weights kindly provided by Laurent et al. [6]. Implementation is the same as in the main experiments, and we treat Packed Ensemble outputs in the same way as (non-adaptive) Deep Ensembles.

2. A Note on Uncertainty Scores U

We remark that our approach is dependent on the compatibility of the uncertainty scores $U^{(1)},U^{(2)},\ldots,U^{(M)}$ between different exits, as ultimately a single τ is used for the downstream uncertainty task. We find that in our experiments, simply using the same score method (e.g. Energy [8]) across all exits is sufficient. However, in a similar scenario, Lin et al. [7] find it necessary to perform an additional score normalisation step. Although this may seem like common sense, we remark that we don't expect our approach to work if different exits use different score methods (e.g one exit uses MSP and another uses Energy), as these score methods may take very different absolute values.

3. Additional Early-Exit Architectures

In addition to MSDNet [3], we also evaluate on GFNet [17] and the ViT-based DVT [10] in Fig. 1. For all early-exit architectures we use publically available pretrained

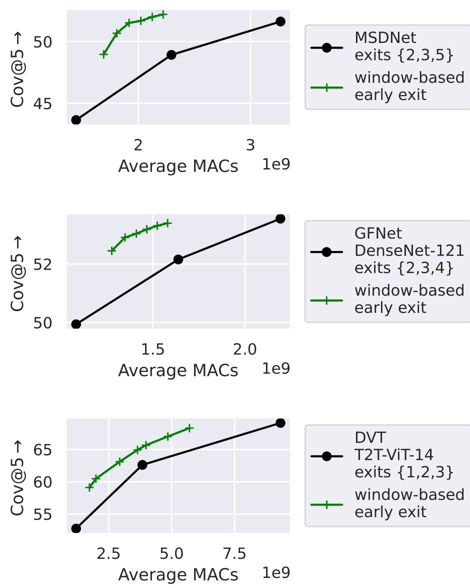

Figure 1. SC results for different early-exit architectures using 3 exits. Top: MSDNet, middle: GFNet, bottom: DVT. We achieve more efficient trade-offs compared to using individual exits.

weights.[4] For MSDNet we use the version of the model designed for ImageNet with step=7, and experiment on a subset $\{2,3,5\}$ of the five available exits. For GFNet we use the model built on DenseNet-121 and exits $\{2,3,4\}$. For DVT we use the model built on T2T-ViT-14 and all 3 exits. The percentiles used for $[t_1,t_2]$ are listed in Tab. 2:

Early-Exit Arch.	$\%\pm\tau^{(1)}$ for $[t_1,t_2]^{(1)}$	$\%\pm\tau^{(2)}$ for $[t_1,t_2]^{(2)}$
MSDNet	[10,15,20,25,30,35]	[10,10,10,10,10,10]
GFNet	[10,15,20,25,30,35]	[10,10,10,10,10,10]
DVT	[10,15,20,30,35,40,45]	[0,0,10,10,10,15,20]

Table 2. Window widths for early-exit architectures.

For all three specialised early-exit architectures we achieve a better uncertainty-computation trade-off compared to using individual exits, validating our approach.

4. Additional Exit Policy Comparisons

We show comparisons between the single-threshold policy, and our window-based policy for OOD detection (FPR@95↓, Openimage-O, $\alpha = 0.5$). For the single-threshold and non-adjusted window approach we set t and $[t_1, t_2]$ based on the percentage of p_{ID} passed on to the next cascade stage. For the *adjusted* window we set $[t_1,t_2]$ according to percentiles measured on p_{mix} instead.

[4] https://github.com/kalviny/MSDNet-PyTorch
https://github.com/blackfeather-wang/GFNet-Pytorch
https://github.com/blackfeather-wang/Dynamic-Vision-Transformer

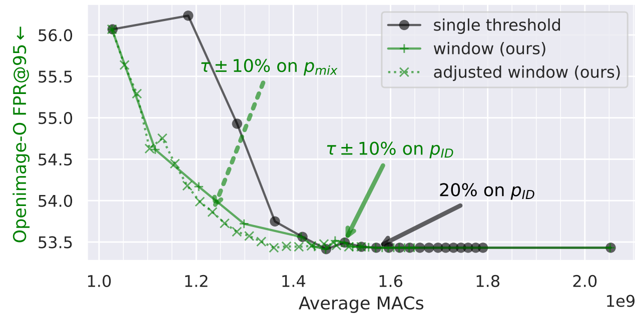

Figure 2. Comparison of OOD detection efficiency for different exit policies. The ratio of ID:OOD data is 1:1 ($\alpha = 0.5$). Using a single threshold is inefficient compared to using a window. Adjusting $[t_1,t_2]$ using statistics from p_{mix} significantly reduces the slowdown caused by distributional shift (if the window is set on p_{ID}). We use an EfficientNet-B2 ensemble with $M = 2$.

Similarly to SC, Fig. 2 shows that the single-threshold approach only improves after t (the exit threshold) passes over τ (the detection threshold). However, this happens earlier as the operating point TPR=95% happens to be closer to the starting point of the single-threshold sweep (it starts from *most* uncertain). Our window-based approach more efficiently improves OOD detection.

Different specific exit policies are also marked. It can be seen that setting the window $[t_1,t_2]$ according to p_{mix} rather than p_{ID} significantly reduces slowdown caused by distribution shift. Setting $[t_1,t_2]$ to $\pm 10\%$ around τ on p_{ID} leads to $\sim 50\%$ of samples from p_{mix} passing through to the second stage. Note that setting $[t_1,t_2]$ at ± 10 percentiles around TPR=95% on ID data only allows 15% of ID data through as the window caps out on one side.

5. Additional Selective Classification Results

We include additional SC results at two more operating points, Risk@50↓ and Cov@10↑ (Fig. 4), which represent, compared to the main results, a lower coverage requirement and a higher risk tolerance respectively. The results are similar to those in the main paper, showing that cascades are able to achieve efficient uncertainty estimation compared to model scaling over a range of different operating thresholds.

6. Accuracy-Computation Results

Fig. 5 shows the accuracy-computation trade-off using single-threshold cascades for EfficientNet and MobileNet-V2. We pass the most uncertain 20% of samples from the 1st model to the second cascade stage. The results are unsurprisingly similar to those in [16], i.e. ensembles are less efficient than single models in the low-compute region, but outperform them for higher computation levels. Cascades then allow ensembles to become more efficient for all levels of compute. We note the baseline accuracy of our Efficient-

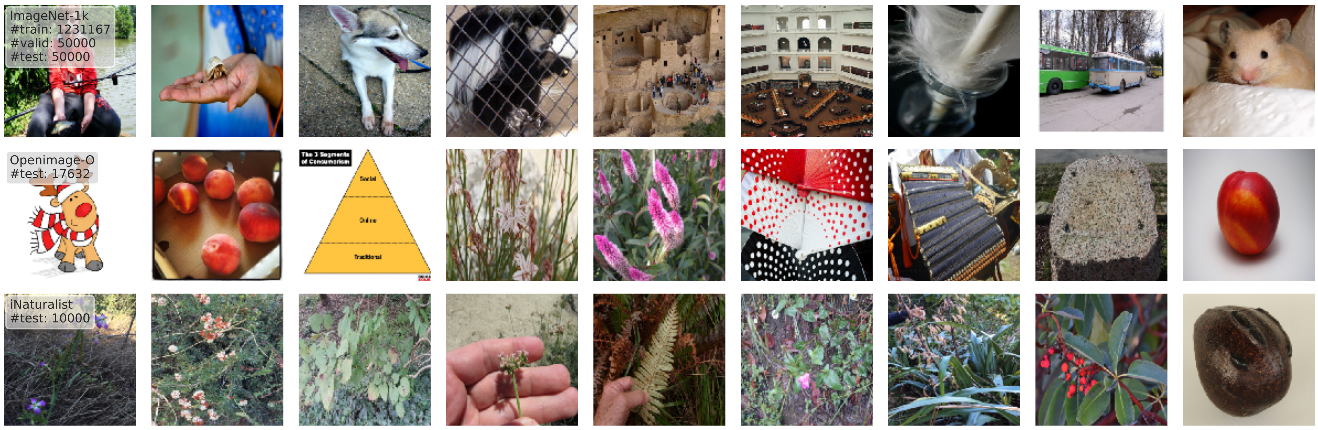

Figure 3. Example images from each image dataset used, with #samples in each split.

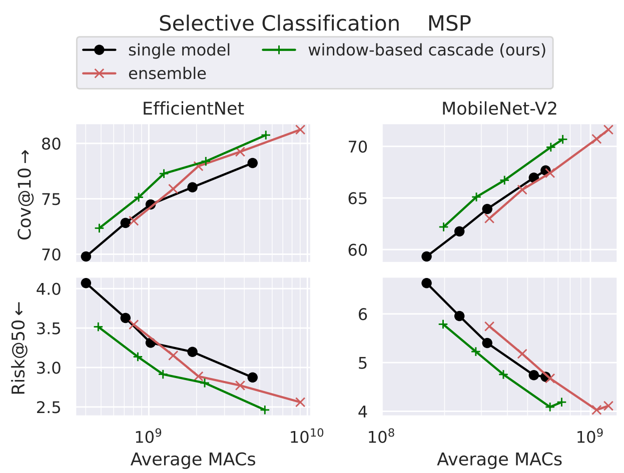

Figure 4. SC–computation comparison for single models, ensembles and window-based cascades, on additional operating thresholds. Results tell the same story as the main paper – cascades are able to achieve the best uncertainty-computation trade-off.

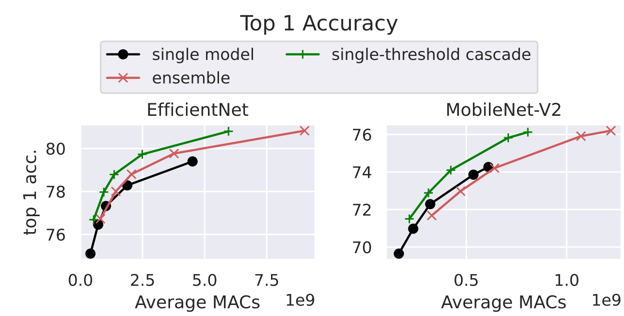

Figure 5. Accuracy–computation comparison for single models, ensembles and single-threshold cascades.

Nets is lower than in [16] as we train them for fewer epochs with a simpler recipe and larger validation set, however, we do not believe this affects the takeaways from our results.

7. Additional Dataset Information

We include randomly sampled example images from the datasets used in this work: ImageNet-1k [11], Openimage-O [5, 15], and iNaturalist [4, 14]. We also show information about the number of samples in each dataset split (Fig. 3). The OOD datasets are recently released high-resolution benchmarking datasets. They aim to move vision-based OOD detection evaluation beyond CIFAR-scale images into more *realistic* image-classification scenarios. The samples in each dataset have been carefully chosen to be semantically disjoint from the label space of ImageNet-1k. Openimage-O contains a wide range of classes like ImageNet, whilst iNaturalist contains botanical images.

8. Acknowledgements

Guoxuan Xia is funded jointly by Arm ltd. and EPSRC. We would like to thank Alexandros Kouris, Pau de Jorge and Francesco Pinto for their helpful discussions and feedback. We would also like to thank Yulin Wang, Olivier Laurent and Gianni Franchi for kindly providing pre-trained weights for our experiments.

References

[1] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. 1

[2] Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *ArXiv*, abs/1706.02677, 2017. 1

[3] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 1

[4] Rui Huang and Yixuan Li. Mos: Towards scaling out-

of-distribution detection for large semantic space. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8706–8715, 2021. 3

[5] Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://github.com/openimages*, 2017. 3

[6] Olivier Laurent et al. Packed ensembles for efficient uncertainty estimation. In *ICLR*, 2023. 1

[7] Ziqian Lin, Sreya Dutta Roy, and Yixuan Li. Mood: Multi-level out-of-distribution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15313–15323, June 2021. 1

[8] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 2020. 1

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 1

[10] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. 1

[11] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 1, 3

[12] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 1

[13] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019. 1

[14] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset, 2017. 3

[15] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. 3

[16] Xiaofang Wang, Dan Kondratyuk, Eric Christiansen, Kris M. Kitani, Yair Movshovitz-Attias, and Elad Eban. Wisdom of committees: An overlooked approach to faster and more accurate models. In *International Conference on Learning Representations*, 2022. 2, 3

[17] Yulin Wang et al. Glance and focus: a dynamic approach to reducing spatial redundancy in image classification. In *NeurIPS*, 2020. 1