Test-time Specialization of Dynamic Neural Networks

Sam Leroux[1,*] Dewant Katare[2] Aaron Yi Ding[2] Pieter Simoens[1]
[1]IDLab, Department of Information Technology, Ghent University - imec, Belgium.
[2]Department of Engineering Systems and Services, Delft University of Technology, The Netherlands.
[*]Corresponding author: sam.leroux@ugent.be

Abstract

In recent years, there has been a notable increase in the size of commonly used image classification models. This growth has empowered models to recognize thousands of diverse object types. However, their computational demands pose significant challenges, especially when deploying them on resource-constrained edge devices. In many use cases where a model is deployed on an edge device, only a small subset of the classes will ever be observed by a given model instance. Our proposed test-time specialization of dynamic neural networks allows these models to become faster at recognizing the classes that are observed frequently, while maintaining the ability to recognize all other classes, albeit slightly less efficient. We benchmark our approach on a real-world edge device, obtaining significant speedups compared to the baseline model without test-time adaptation.

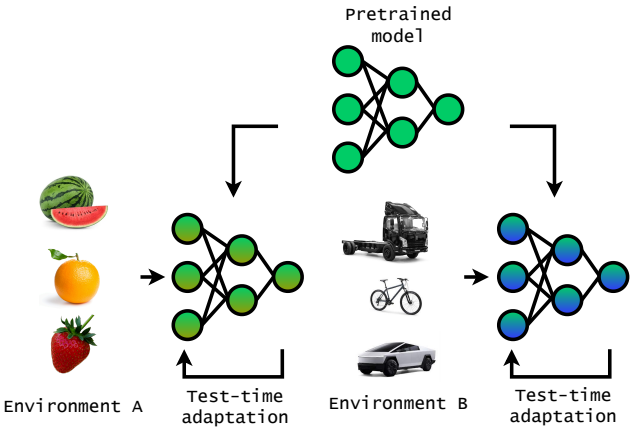


Figure 1. We propose to first train a model on a large and diverse dataset. This model is then deployed on edge devices where it is immediately able to make useful predictions. Over time, the model is updated in a self-supervised way (test-time adaptation) to become more specialized and efficient at processing the data that is commonly observed in this environment.

1. Introduction

For many years, the ImageNet1K dataset [36] was considered to be the go-to large-scale image recognition dataset. With 1,000 object classes and over a million training images, it was a challenging problem, both in terms of task performance and training cost. However with advances in hardware [16], distributed approaches for training [42] and semi-automatic data labeling [49], it became feasible to collect and train on much larger datasets such as the ImageNet22K dataset (22K classes, 14M images), Tencent ML-images (11K classes, 18M images) [45] and even larger, proprietary datasets such as JFT-300M (18K classes, 300M images) [38] or JFT-3B (30k classes, 3B images)[49]. To capture the information contained in these datasets, increasingly large models are being developed. Where a commonly used ResNet50 model had 25M parameters [13], we are now seeing computer vision models with 2B [4], 3B [28], or even up to 22B parameters [6]. There is increasing evidence that very large, overparameterized models are needed to generalize when training on these extremely large datasets [2], making it likely that even larger models will be developed in the near future. Because of the use of proprietary datasets and the vast computational resources required, only a few organizations are able to train models of this scale. Fortunately, trained models are often made public and developers are increasingly using them to power their applications. However, the data these models encounter after deployment tends to be less varied than the data used for training. The ImageNet1K dataset for example, contains examples of “Gazelle”, “Ambulance”, “Yurt”, “Pirate ship” and 996 other wildly different classes. It is unlikely that an application deployed on an edge device will ever see examples of all of these in the real world. It is more likely that the model will need to differentiate between similar types of objects such as different types of fruits or vehicles. While it is of course possible to only train the model on the subset of interest, this would require a careful selection process and potentially limit the model’s adaptability to unforeseen scenarios. We instead propose to

train a model on data from all classes and to use test time adaptation (TTA) to specialize for the subset of interest. Ideally, we can then train a very large model once, on a huge (proprietary) dataset with a very large number of classes and optimize it after deployment for each different environment. Figure 1 illustrates this idea, here a model is trained on a large dataset such as ImageNet1K. It is then deployed in two environments where it will either observe mostly fruits or vehicles. Through test-time adaptation, the first model instance will become more efficient at recognizing different fruits while the second instance will be optimized to recognize vehicles. Both models will remain capable of recognizing all 1,000 ImageNet1K classes although images of classes that are observed rarely, will require more processing time.

The remainder of this paper is organized as follows. In Section 2 we first give an overview of related work in the fields of adaptive neural networks and test time adaptation. We then introduce our approach in Section 3 and experimentally validate it in Section 4. We conclude in Section 5 and give a few pointers for future research.

2. Related work

We propose Adaptive Neural Networks that use Test Time Adaptation to obtain efficient location specific models. In this section, we briefly review the literature of these three topics.

2.1. Adaptive neural networks

A ResNet50 model obtains an ImageNet top-1 classification accuracy of around 76% and has $25M$ parameters [13]. Its larger sibling, ResNet152, obtains an accuracy of 78% but has almost three times as many trainable parameters. A similar observation can be made for most families of neural network architectures: while the additional capacity of larger models results in a higher overall performance, this only makes a difference for a small subset of the test samples. The computational cost of processing these samples will be much higher for the larger model, even for the samples that could be classified correctly by the smaller model. Adaptive Neural Networks (sometimes called Dynamic Neural Networks) try to address this issue by dynamically adjusting their computational costs based on input complexity [12]. All these models incorporate some sort of conditional computation component that selectively activates parts of the model. We briefly describe some of the most relevant approaches in the following paragraphs and refer to [12] for a more in-depth overview.

A popular approach is to incorporate early exits into a neural network architecture that allow for premature predictions. Traditionally, deep neural networks process the input through all layers before producing a final output. However, with early exits, intermediate predictions or decisions can be made at earlier layers, based on the information available at that point. If the model is sufficiently confident in an early prediction, the remaining layers are not evaluated anymore, reducing the computational cost. Early exits in deep neural networks trained for image classification were first proposed in [18] and later revisited by [1, 20, 39]. Early exits are now also being used in object detection [47], generative models [17], anomaly detection [41] and intrusion detection [31]. Our work relies on these early exits but combines them with test-time adaptation to improve the model's efficiency over time.

An alternative approach is to use multi-branch architectures where multiple parallel network branches can be selectively executed based on the current input. Each branch learns to specialize, resulting in a Mixture-of-Experts (MoE) model [3, 33, 35]. At runtime, only a subset of the branches are used, depending on the current input sample, resulting in a lower computational cost.

Other techniques to selectively disable parts of the model at runtime are based on pruning, a common strategy to reduce the computational cost and memory footprint of deep neural networks by pruning weights that have minimal impact on the network's overall performance [25]. By identifying and removing these insignificant weights, which contribute less to the network's output, the model can be made more efficient without compromising its accuracy. Pruning techniques typically involve setting small-weight connections to zero or removing them entirely, resulting in a sparser network architecture that requires fewer computations and storage resources. Dynamic pruning techniques apply this at inference time to deactivate neurons or convolutional filters of the next layer, typically based on statistics of previous layers [8, 15, 19, 23, 26].

An interesting research direction is to design models that learn how to dynamically allocate their resources by balancing two loss functions during training where one is the typical classification or regression loss and the other is a regularization term that encourages the model to use fewer resources if possible. Models can for example learn what part of the input image it should focus on, using fewer computations for background parts [7] or can dynamically adjust the number of visual tokens in a Visual Transformer model [44, 48]. Other techniques learn to iteratively refine the features extracted by previous layers and decide at runtime how much refinement is needed to process the current input [21].

2.2. Location specific models

Through the utilization of extensive and diverse datasets, along with techniques such as data augmentation and regularization, developers strive to train models that can effectively generalize to novel scenarios. Depending on the specific application, the model can be deployed in a centralized, cloud-based system, or alternatively, a decentralized approach can be chosen, where the model is evaluated on edge devices situated near the data-generating sensor. The deployment of models at the edge offers various advantages, including enhanced privacy, bandwidth efficiency, and scalability. This paper presents the argument that in many instances where models are deployed on edge devices, their ability to generalize to different environments becomes less crucial. Given that the model will solely encounter inputs from a specific environment throughout its lifespan, a certain degree of overfitting to that environment is tolerable or even beneficial. Based on this notion, it is reasonable to anticipate comparable accuracy by employing smaller models that are specifically tailored to the given environment, even if they do not exhibit strong generalization to other environments. While only a few studies have explored this location-specific aspect of edge computing [5, 22, 30, 34], they have adopted a similar methodology. These works employ a large *teacher* model trained on diverse datasets to generate a training signal for a smaller *student* model specialized for a particular environment. In this research, we propose a similar approach but instead of using separate teacher and student models, we use self-supervision from a deep exit to an early exit of an adaptive neural network.

2.3. Test time adaptation

In conventional machine learning applications, the training and test phases are typically treated as separate entities. Once the model is trained, it is expected to generalize well to unseen test data and its weights remain fixed. However, an intriguing avenue of research explores the possibility of updating models during the test phase as well. Test time adaptation (TTA) can for example be used to handle domain shifts such as changes in weather conditions or sensor degradation [40]. Since labeled information is usually unavailable during testing, these approaches rely on self-supervised or unsupervised learning techniques. Broadly speaking, two categories can be distinguished: backward-based and backward-free methods [43]. Backward-based methods employ gradient-based optimization to adjust the model parameters, optimizing a loss function like entropy [40] or using pseudo-labeling [24]. However, these approaches have drawbacks, including higher computational costs and the risk of catastrophic forgetting where model performance deteriorates due to noisy gradient updates [43]. On the other hand, backward-free methods aim to align the features extracted during testing with the training distribution by adjusting parameters such as Batchnorm layers [32, 37].

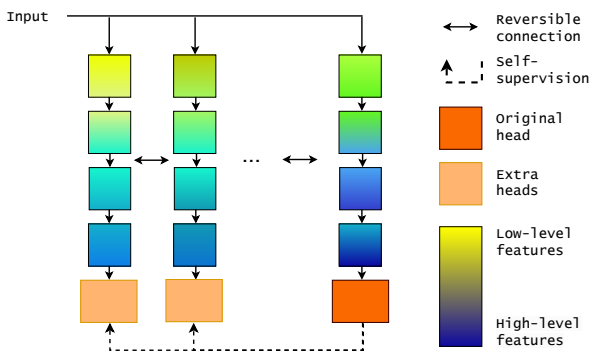


Figure 2. Our architecture is based on the Reversible Column Networks (RevCol) consisting of n columns that each produce a hierarchical series of feature representations. We add a classification head to each column that acts as an early exit. We use self-supervision from a latter head to an early head at inference-time to improve the performance of the earlier heads.

Our work can be classified as a backward-based approach. However to alleviate the aforementioned issues, we only update a small portion of the model, limiting the computational overhead and ensuring the model does not suffer from catastrophic forgetting. Unlike existing research on test time adaptation, our objective is not to compensate for domain shift, but rather to improve the efficiency of location-specific models.

3. Approach

In this section, we first explain the structure of the RevCol model that forms the backbone of our proposed architecture. We then explain how we transformed this architecture into an adaptive neural network and finally, introduce the test-time adaptation procedure. A detailed depiction of the architecture can be seen in Figure 2.

3.1. Backbone

Our model is based on the Reversible Column Network (RevCol) architecture [4]. This recently proposed architecture has demonstrated outstanding performance on various tasks, including image classification, object detection, and semantic segmentation. It strikes a balance between computational efficiency and memory requirements, similar to other high-performing models like Swin [27] and ConvNeXt [29]. The RevCol architecture is particularly intriguing due to its composition of N subnetworks known as columns, each possessing an identical structure. In this design, every column receives a copy of the input and produces a hierarchical series of feature representations.

While a conventional convolutional neural network (CNN) can be viewed as a RevCol with only one column, the RevCol architecture allows for the propagation of features from column i to column i+1. This enables subsequent columns to enhance the quality of features extracted by their preceding counterparts, similar to the iterative refinement process observed in ResNets [10].

To facilitate efficient training, the propagation of features between consecutive blocks utilizes reversible transformations [9]. This approach significantly reduces memory consumption by eliminating the need to store intermediate activations for gradient calculations during backpropagation. Instead, the intermediate activations can be reconstructed on-the-fly as required.

3.2. Early exits

In the original RevCol specification, the features extracted by the last column are used as input to a task specific head such as a linear layer for classifcation, a Cascade Mask R-CNN head for object detection [14] or a UPerNet head for semantic segmentation [46]. Our goal however is to design an adaptive neural network with early exits where hard input samples require more processing than easier to understand samples. This matches well with the multi-column RevCol architecture where subsequent columns learn to extract better features but where the features extracted by early columns might already be good enough to make an accurate prediction for some samples.

Instead of a single head attached to the last column, we propose to attach heads to all of the earlier columns as well. These follow the same structure as the final head and are also trained in the same way (i.e. by minimizing the cross entropy loss using gradient descent). As the model processes an input sample at inference time, a decision must be made regarding the column at which the processing should terminate. We follow the procedure of [18] and interpret the output of the softmax activation as a confidence metric. While it is known that neural networks with softmax activations are sometimes not well calibrated [11] (i.e. their confidence does not match their accuracy), we found that there was a very strong correlation between model confidence and accuracy. At inference time, we then simply evaluate each branch of the model, followed by the corresponding exit, we compare the maximum predicted score after softmax activation with a predefined threshold. If the score exceeds the threshold, we return the prediction of that branch and do not evaluate the subsequent branches. Since the threshold can be changed dynamically at runtime, this makes it possible to adapt the inference strategy based on the specific requirements and constraints of the deployment environment. For example, we can reduce the threshold when the battery level of the device reaches a critical point to prioritize efficiency over accuracy.

3.3. Test-time specialization

The adaptive computation aspect is just the initial component of our proposed model architecture. As stated in the introduction, our objective is to enable on-device specialization, where the model learns to enhance its efficiency in processing input samples from the specific deployment environment. In our framework, this entails the earlier heads gradually improving the accuracy of their predictions over time. Since labeled data is unavailable after deployment, we leverage the prediction of the selected head as target and employ backpropagation to minimize the cross entropy loss between this target and the prediction of the early exit. This updates the earlier heads in order to achieve similar predictions as the final head in the future.

However, we refrain from updating the backbone in this process. The backbone is shared among all heads, and modifying the parameters of early columns would alter the features provided to subsequent columns, potentially compromising the performance of subsequent heads. Moreover, updating the backbone would significantly increase the computational cost. Given that our objective is to obtain an efficient model for inference, it is crucial to minimize the computational expense associated with self-supervised specialization by only updating a small portion of the heads.

4. Results

In this section, we demonstrate the performance of our adaptive neural network with test-time adaptation, applied to the task of image classification. We use a pretrained RevCol-T architecture with four columns as the backbone. This model was trained on the Imagenet dataset and obtains a classification accuracy of 82.1%. We first explain how this model is transformed into an adaptive architecture with early exits in Section 4.1 and then show how test-time adaptation allows the model to specialize at runtime in Section 4.2. All latencies reported were measured on an NVIDIA Jetson AGX Orin development board using a batch size of 1.

4.1. Early exits

We trained three additional classification heads for the first three columns. These heads were trained using all available Imagenet training data (i.e. all 1,000 classes). The top part of Table 1 shows that these early exits obtain an accuracy of 61.8%, 72.9% and 78.7% respectively, lower than the baseline full model that obtains an accuracy of 82.1% but also each with a significantly lower computational cost than the full model. The latencies reported for these fixed exits also

Table 1. Inference latency measured on the NVIDIA Jetson Orin board and corresponding Imagenet validation accuracy (top 1).

Model		Latency (ms)	Accuracy
Fixed exit	0	15.4 ± 1.1	61.8 %
	1	30.3 ± 0.6	72.9 %
	2	44.6 ± 0.8	78.7 %
	3	55.3 ± 1.0	82.1 %
Threshold	0.5	22.7 ± 15.3	72.8 %
	0.6	25.1 ± 16.1	75.4 %
	0.7	27.9 ± 16.9	77.7 %
	0.8	30.7 ± 16.7	79.4 %
	0.9	34.8 ± 14.2	80.9 %
Baseline		53.1 ± 0.6	82.1 %

include the overhead of evaluating all previous intermediate heads. By comparing the inference latency of exit 3 with that of the baseline model, we conclude that this overhead is limited to two milliseconds in the worst case, negligible compared to the total computational cost of the model.

Once these early exits have been trained, we can use them to implement an adaptive inference procedure. The second part of Table 1 shows the obtained accuracies and corresponding runtime latencies for different threshold values. In this experiment, we used uniform thresholds for all branches but these can be further tuned to obtain an even better accuracy-latency trade-off. We can see that a threshold of 0.9 results in an accuracy of 80.9%, very close to the baseline accuracy of 82.1% while having a 30% lower inference latency.

In Figure 3, we present qualitative results that showcase examples of ImageNet validation images with varying inference latencies. The images near the left demonstrate instances where the model achieved low inference latencies, while those on the right exhibit higher latencies. We observe that images featuring a singular, prominently centered object tend to be easier for the model to recognize, whereas images depicting zoomed-in objects or lacking a distinct foreground object pose greater recognition challenges.

4.2. Test-time specialization

In the previous section, we trained our model on data from all 1,000 classes of the ImageNet dataset. In this paper, we argue that it is unlikely that the model will observe instances of all classes in real-world deployment scenarios. We would therefore prefer that the model is more efficient at recognizing those classes that are observed frequently in a specific location while maintaining the capability to recognize all other classes, albeit slightly less efficient. We also assume that we do not know beforehand which classes the model will eventually observe.

To simulate this, we created six distinct subsets of Imagenet classes, each containing similar types of objects: fruits/vegetables, dogs, insects, snakes, sharks and vehicles. The number of classes in these subsets range from three (sharks) to 118 (Dogs). In Figure 4, we depict the trade-off between inference latency, as measured on the NVIDIA Jetson board, and classification accuracy. The solid lines represent this trade-off for the model before test-time adaptation with the green solid line denoting classes belonging to the selected subset and the red solid line representing all other classes. The thresholds ranged from 0.5 to 0.9 to generate these curves. A curve closer to the upper left corner signifies a more favorable trade-off between inference latency and accuracy.

We leveraged the ImageNet test set to perform the test-time adaptation. Utilizing the pretrained baseline model, we extracted those images belonging to the selected subset and passed them through the adaptive model, following the procedure from Section 3 to update the model. We used the output of the selected branch as target to update the earlier, not selected branches. Note that no labeled data is needed for test-time adaptation. We only update the classification heads of the branches and leave all parameters of the convolutional layers fixed. We employed stochastic gradient descent with a learning rate of 10^{-4} and batch size 1. After processing 100 samples per class, we evaluate our model on the ImageNet validation set which contains 50 samples per class. It is worth mentioning that the names "test set" and "validation set" might be misleading in this setup. As is customary in the literature, we report the performance on the ImageNet *validation set*, which contains ground truth labels needed for evaluation. The ImageNet *test set* lacks publicly available labels and is typically not used. However, in our experiment, we employed it for test-time adaptation. Importantly, all reported accuracies are measured on a completely held-out set, unused for training and test-time adaptation.

The dotted lines in Figure 4 illustrate the results after test-time adaptation. By comparing the green solid line and the green dotted line, it's evident that the model becomes significantly more efficient at recognizing the objects from the selected subset. For instance, within the "snakes" subset, the base model can achieve an accuracy of 74% on this subset. This requires a high threshold, resulting in an average inference latency of 38ms. However, after test-time adaptation, we can reach the same accuracy with a considerable lower average latency of 27ms. Alternatively,

Figure 3. Examples of Imagenet validation images with a low computational cost (left) and a high computational cost (right).

given a maximum latency of 27 ms, we can now obtain an accuracy of 74% on this subset, compared to 69%. The red lines depict similar measurements but for all classes not belonging to the selected subset. Here, we observe a slight reduction in efficiency, necessitating a higher latency to maintain the same accuracy. Nevertheless, the decrease is marginal, and crucially, the model retains the ability to achieve the same maximum accuracy. The same observation can be made for all subsets. After test-time adaptation, the model is able to recognize images from the selected subset more efficiently while the efficiency of all other classes degrades slightly. For the Dogs subset (118 classes), this degradation is much more severe than for a smaller subset such as Sharks (3 classes).

Given that we propose to update the model at inference time, there is an associated increase in computational cost during deployment. However, it's important to note that we only update a minimal portion of the model, i.e. only the classification heads of the first branches. In the worst-case scenario, where no early branch returns a prediction, calculating the loss and executing the gradient update takes approximately 1 ms, as measured on the NVIDIA Jetson board. We contend that this is negligible relative to the total inference latency of the model and the speedup achievable through test-time adaptation.

5. Conclusion and future work

In this paper, we introduced a novel approach to specialize a model to become more efficient at recognizing frequently observed objects. Leveraging Reversible Column Networks (RevCol), we developed an adaptive neural network model capable of delivering predictions sooner when an early branch is sufficiently confident in its prediction. Through experimentation on the NVIDIA Jetson Orin development board, we demonstrated that this early-exit strategy yields significant real-world speedups.

Furthermore, we illustrated how these early exits can facilitate test-time adaptation. In cases where initial exits fail to provide a confident prediction, we employ the prediction of the selected exit as a target to update the classification heads of these early branches. Since we only update a small portion of the model, the update procedure is swift. Our experimental findings validate that this approach indeed enhances the runtime efficiency of the model for the subset of relevant classes during test-time.

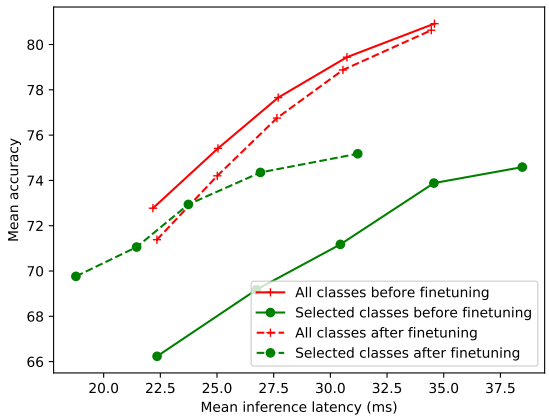
We believe that test-time adaptation represents a valuable research direction. In future work, we will investigate the issue of catastrophic forgetting, where the model gradually loses its ability to classify rare classes after updates. We anticipate that our approach will exhibit relative robustness against catastrophic forgetting, given that the internal layers remain untouched and thus maintain the capability to extract representative features for all classes. Additionally, we never update the last exit of the model, ensuring that it consistently maintains the same accuracy for all classes as the baseline model.

However, a significant challenge lies in model calibration. Our approach relies on a manually configured threshold value, which may not align precisely with the resulting accuracy. For instance, a threshold of 0.8 does not guarantee an accuracy of 80%. This discrepancy can complicate the selection of a suitable threshold, particularly as the model evolves over time and the distribution of confidences may change. While various model calibration techniques exist, their application in a dynamic setting with test-time adaptation remains unclear. Addressing these challenges will be central to advancing the effectiveness and applicability of our approach in practical settings.

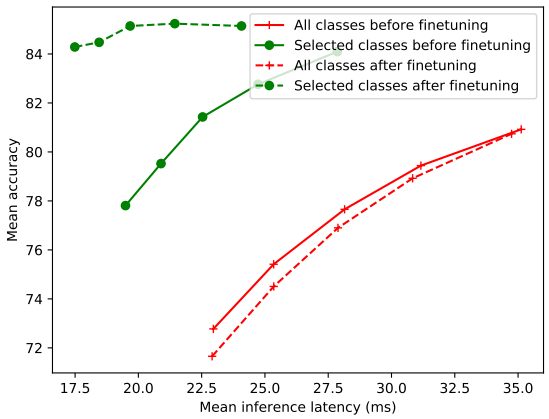
Finally, our aim is to expand our approach to encompass other tasks like object detection or semantic segmentation. Although the RevCol architecture has proven suitable for such tasks, determining the accuracy of early exits and updating them efficiently remains an open research question.

Acknowledgements

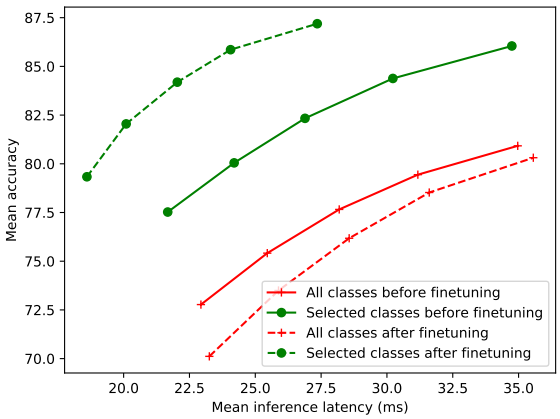
We gratefully acknowledge the support of Fonds Wetenschappelijk Onderzoek – Vlaanderen (FWO) in facilitating the research stay of the first author at the Department of Engineering Systems and Services at Delft University of Technology in The Netherlands. This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.



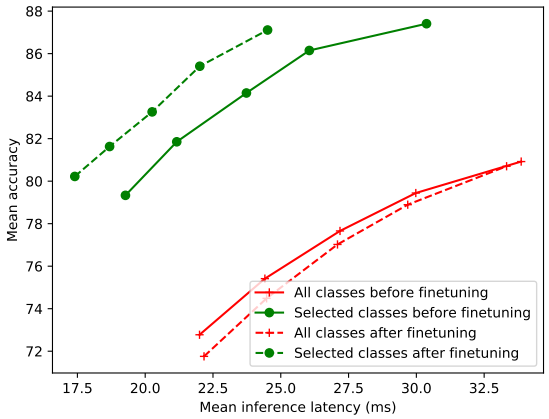
(a) Snakes (17 classes)



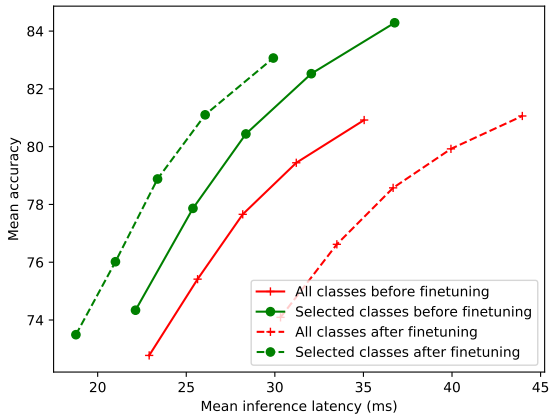
(b) Fruits and vegetables (21 classes)



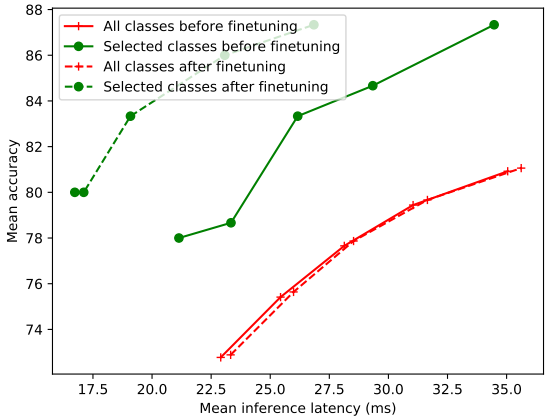
(c) Cars and vehicles (42 classes)



(d) Insects (27 classes)



(e) Dogs (118 classes)



(f) Sharks (3 classes)

Figure 4. Trade-off between inference latency and classification accuracy, measured on the NVIDIA Jetson board. Solid lines represent the model's performance before test-time adaptation, with green lines indicating selected subset classes and red lines representing all other classes. Dotted lines illustrate the efficiency after test-time adaptation, revealing significant reductions in latency for the selected subset classes.

References

[1] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for efficient inference. In *International Conference on Machine Learning*, pages 527–536. PMLR, 2017. 2

[2] Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34:28811–28822, 2021. 1

[3] Shaofeng Cai, Yao Shu, and Wei Wang. Dynamic routing networks. In *proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3588–3597, 2021. 2

[4] Yuxuan Cai, Yizhuang Zhou, Qi Han, Jianjian Sun, Xiangwen Kong, Jun Li, and Xiangyu Zhang. Reversible column networks. *arXiv preprint arXiv:2212.11696*, 2022. 1, 3

[5] Anthony Cioppa, Adrien Deliege, Maxime Istasse, Christophe De Vleeschouwer, and Marc Van Droogenbroeck. Arthus: Adaptive real-time human segmentation in sports through online distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 3

[6] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *International Conference on Machine Learning*, pages 7480–7512. PMLR, 2023. 1

[7] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1039–1048, 2017. 2

[8] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. *arXiv preprint arXiv:1810.05331*, 2018. 2

[9] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. The reversible residual network: Backpropagation without storing activations. *Advances in neural information processing systems*, 30, 2017. 4

[10] Klaus Greff, Rupesh K Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016. 4

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017. 4

[12] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7436–7456, 2021. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 4

[15] Weizhe Hua, Yuan Zhou, Christopher M De Sa, Zhiru Zhang, and G Edward Suh. Channel gating neural networks. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[16] Norman P Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. A domain-specific supercomputer for training deep neural networks. *Communications of the ACM*, 63(7):67–78, 2020. 1

[17] Polina Karpikova, Ekaterina Radionova, Anastasia Yaschenko, Andrei Spiridonov, Leonid Kostyushko, Riccardo Fabbricatore, and Aleksei Ivakhnenko. Fiancee: Faster inference of adversarial networks via conditional early exits. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12032–12043, 2023. 2

[18] Sam Leroux, Steven Bohez, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. Resource-constrained classification using a cascade of neural network layers. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2015. 2, 4

[19] Sam Leroux, Steven Bohez, Cedric De Boom, Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. Lazy evaluation of convolutional filters. *arXiv preprint arXiv:1605.08543*, 2016. 2

[20] Sam Leroux, Steven Bohez, Elias De Coninck, Tim Verbelen, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. The cascading neural network: building the internet of smart things. *Knowledge and Information Systems*, 52:791–814, 2017. 2

[21] Sam Leroux, Pavlo Molchanov, Pieter Simoens, Bart Dhoedt, Thomas Breuel, and Jan Kautz. Iamnn: Iterative and adaptive mobile neural network for efficient image classification. *arXiv preprint arXiv:1804.10123*, 2018. 2

[22] Sam Leroux, Bo Li, and Pieter Simoens. Automated training of location-specific edge models for traffic counting. *Computers and Electrical Engineering*, 99:107763, 2022. 3

[23] Changlin Li, Guangrun Wang, Bing Wang, Xiaodan Liang, Zhihui Li, and Xiaojun Chang. Dynamic slimmable network. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 8607–8617, 2021. 2

[24] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International conference on machine learning*, pages 6028–6039. PMLR, 2020. 3

[25] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021. 2

[26] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017. 2

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer:

Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 3

[28] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12009–12019, 2022. 1

[29] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 3

[30] Dani Manjah, Davide Cacciarelli, Baptiste Standaert, Mohamed Benkedadra, Gauthier Rotsart de Hertaing, Benoît Macq, Stéphane Galland, and Christophe De Vleeschouwer. Stream-based active distillation for scalable model deployment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4998–5006, 2023. 3

[31] Fares Meghdouri, Maximilian Bachl, and Tanja Zseby. Eagernet: Early predictions of neural networks for computationally efficient intrusion detection. In *2020 4th Cyber Security in Networking Conference (CSNet)*, pages 1–7. IEEE, 2020. 2

[32] M Jehanzeb Mirza, Jakub Micorek, Horst Possegger, and Horst Bischof. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14765–14775, 2022. 3

[33] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8080–8089, 2018. 2

[34] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online model distillation for efficient video inference. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 3573–3582, 2019. 3

[35] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021. 2

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015. 1

[37] Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in neural information processing systems*, 33:11539–11551, 2020. 3

[38] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017. 1

[39] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016. 2

[40] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020. 3

[41] Haoliang Wang, Chen Zhao, Xujiang Zhao, and Feng Chen. Layer adaptive deep neural networks for out-of-distribution detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 526–538. Springer, 2022. 2

[42] Shibo Wang, Jinliang Wei, Amit Sabne, Andy Davis, Berkin Ilbeyi, Blake Hechtman, Dehao Chen, Karthik Srinivasa Murthy, Marcello Maggioni, Qiao Zhang, et al. Overlap communication with dependent computation via decomposition in large deep learning models. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 93–106, 2022. 1

[43] Wei Wang, Zhun Zhong, Weijie Wang, Xi Chen, Charles Ling, Boyu Wang, and Nicu Sebe. Dynamically instance-guided adaptation: A backward-free approach for test-time domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24090–24099, 2023. 3

[44] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in Neural Information Processing Systems*, 34:11960–11973, 2021. 2

[45] Baoyuan Wu, Weidong Chen, Yanbo Fan, Yong Zhang, Jinlong Hou, Jie Liu, and Tong Zhang. Tencent ml-images: A large-scale multi-label image database for visual representation learning. *IEEE Access*, 7:172683–172693, 2019. 1

[46] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018. 4

[47] Le Yang, Ziwei Zheng, Jian Wang, Shiji Song, Gao Huang, and Fan Li. Adanet: An adaptive object detection system based on early-exit neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 2023. 2

[48] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10809–10818, 2022. 2

[49] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022. 1