Multi-Exit Resource-Efficient Neural Architecture for Image Classification with Optimized Fusion Block

Youva Addad, Alexis Lechervy, Frédéric Jurie

▶ **To cite this version:**

Youva Addad, Alexis Lechervy, Frédéric Jurie. Multi-Exit Resource-Efficient Neural Architecture for Image Classification with Optimized Fusion Block. 2023. hal-04181149v1

HAL Id: hal-04181149

https://hal.science/hal-04181149v1

Preprint submitted on 15 Aug 2023 (v1), last revised 18 Aug 2023 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multi-Exit Resource-Efficient Neural Architecture for Image Classification with Optimized Fusion Block

Youva Addad Alexis Lechervy Frédéric Jurie

Normandy University, ENSICAEN, UNICAEN, CNRS, GREYC, France
youva.addad@unicaen.fr

Abstract

In this paper, we propose a test-time resource-efficient neural architecture for image classification. Building on MSDNet [12], our multi-exit architecture excels in both anytime classification, which allows progressive updates of predictions for test examples and facilitates early output, and budgeted batch classification, which allows flexible allocation of computational resources across inputs to classify a set of examples within a fixed budget. Our proposed multi-exit architecture achieves state-of-the-art performance on CIFAR10 and CIFAR100 in these two critical scenarios, thanks to a novel feature fusion building block combined with an efficient stem block.

1. Introduction and related works

State-of-the-art convolutional neural networks (CNNs) such as EfficientNet [25, 26], ResNet [7], and DenseNet [13] possess remarkable network depth, enabling them to achieve exceptional accuracy. However, these deep models often incur significant computational costs, making real-time inference unattainable on resource-constrained platforms such as smartphones, wearable health monitoring devices, or robotic platforms.

In recent years, considerable efforts have been made to improve the inference efficiency of deep CNNs. Various approaches have been explored, including efficient architecture design [4, 34, 11, 10], network pruning [5, 8, 31, 20], weight quantization [17, 14, 23, 16], knowledge distillation [9, 18, 1], and adaptive inference [28, 12, 6, 30, 19]. Notable contributions in this area include MobileNet [11], ShuffleNet [34], and SqueezeNet [15], which introduced innovative strategies such as depth-wise convolutions, channel shuffling, and fire modules to minimize computational effort while maintaining satisfactory accuracy. In addition, recent advances in neural architecture search (NAS) [29, 2] and knowledge distillation [3, 9] have also played a key role in producing compact and efficient models without significant performance degradation. As the importance of efficient neural networks continues to grow, this paper aims to enrich ongoing efforts by presenting a novel approach that further enhances model efficiency in specific real-world applications.

Adaptive inference aims to reduce computational redundancy on "easy" examples. Specifically, this method involves designing a model with the ability to intelligently select specific segments of the network to execute during test time, depending on the input it receives. "Easy" samples require less computation than "hard" ones. An example of an adaptive inference technique is early exit [12, 30, 22, 6].

Dynamic Early Exit Networks create multiple classifiers within the depth of a network, enabling rapid prediction of high-confidence samples at early stages (easy samples) without activating deeper layers. This recognizes that not all samples require the same model complexity for accurate prediction. Unlike traditional architectures, dynamic early-exiting networks introduce branching points [27] at different depths to assess prediction confidence early. Shallow classifiers provide fast predictions for simple samples, saving computation, while complex samples progress through deeper layers for accurate predictions at a slightly higher cost, balancing efficiency and accuracy. In addition, these networks adapt inference based on resource and latency requirements, prioritizing early exit for speed or exploring deeper branches for improved accuracy.

MSDNet, proposed by Huang et. al. [12], is a leading approach for dynamic early exit that effectively addresses two main challenges to achieve resource-efficient image classification. The first challenge, where classifiers change the internal representation, is solved by using dense connectivity [13], which prevents the dominance of a single early exit and balances the tradeoff between early and later classification through the loss function. The second challenge, the lack of coarse-scale features in early layers, is addressed by employing a multiscale network structure. At each layer,

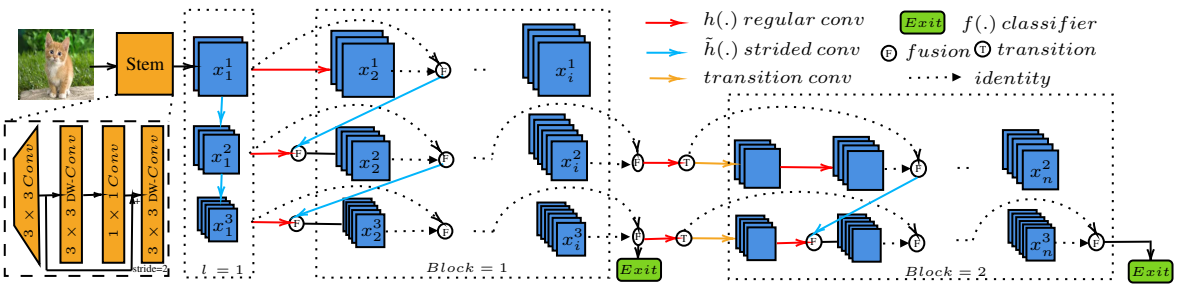

Figure 1: The proposed architecture operates with $S = 3$ scales. It consists of two blocks, with a reduction in the number of scales within each successive block. F represents the concatenation operator, while T denotes the transition operator.

the network produces features of all scales, from fine to coarse. MSDNet has served as the basis for a number of other works, such as those by Meronen et. al. [21].

One limitation of MSDNet lies in the way it reuses previously computed features, as the last features of each scale are not shared with subsequent scales. To overcome this limitation, we present a novel fusion layer that improves the reuse of previous features. In addition, we introduce a novel stem that effectively limits the overall computation of the network, further improving its efficiency.

2. Method

The proposed method extends the foundational principles of MSDNet [12]. The architecture of our model is given in Fig. 1. As mentioned in the introduction, it includes a novel operator for effectively fusing layers across scales and depth, and a novel stem block that further enhances the model's capabilities.

An improved fusion layer. As shown in Fig. 1, the key issue with multi-exit architectures is knowing which features (across scales and depths) to combine at each level, and how to combine them. Our model relies on a feature fusion technique that incorporates both local and global context, enabling the model to make well-informed predictions and decisions. The integration of features through concatenation and strided convolutions plays a central role in enhancing the model's ability to learn complex patterns, leading to significant improvements in its overall performance.

In our model, the output x_l^s of layer l at the s^{th} scale is obtained using the concatenation operator denoted by $[...]$. The transformation $h_l^s(.)$ represents a regular convolution operation, while $\tilde{h}_l^s(.)$ corresponds to a strided convolutional operation. $\tilde{h}_1^1(.)$ correspond to the stem layer which will be described in the next paragraph.

More specifically, the fusion is done according to the following equations:

x_l^s	$l=1$	$l=2$	$l=3$	$l=4$
$s=1$	$\tilde{h}_1^1(x_0^1)$	$h_2^1(x_1^1)$	$h_3^1(x_1^1, x_2^1)$	$h_4^1(x_1^1, x_2^1, x_3^1)$
$s=2$	$\tilde{h}_1^2(x_1^1)$	$\begin{bmatrix} h_2^2([x_1^1, x_2^1]) \\ h_2^2([x_1^2]) \end{bmatrix}$	$\begin{bmatrix} h_3^2([x_1^1, x_2^1, x_3^1]) \\ h_3^2([x_1^2, x_2^2]) \end{bmatrix}$	$\begin{bmatrix} h_4^2([x_1^1, x_2^1, x_3^1, x_4^1]) \\ h_4^2([x_1^2, x_2^2, x_3^2]) \end{bmatrix}$
$s=3$	$\tilde{h}_1^3(x_1^2)$	$\begin{bmatrix} h_2^3([x_1^2, x_2^2]) \\ h_2^3([x_1^3]) \end{bmatrix}$	$\begin{bmatrix} h_3^3([x_1^2, x_2^2, x_3^2]) \\ h_3^3([x_1^3, x_2^3]) \end{bmatrix}$	$\begin{bmatrix} h_4^3([x_1^2, x_2^2, x_3^2, x_4^2]) \\ h_4^3([x_1^3, x_2^3, x_3^3]) \end{bmatrix}$

Although this mechanism may seem similar to MSDNet's, it has one important difference: it makes greater use of features computed at the previous scale level. While this results in better performance, it has the potential disadvantage of increasing the number of computations required to process the features. For this reason, we have also introduced a transition layer (which is equivalent to the original version, except it disregards the prior scale in the convolution process.) that is designed to effectively optimize the reduction of spatial dimensions and channel numbers. Instead of merging concatenated features and halving the number of channels, we directly halve the number of channels within the current scale. This reduction is achieved using a 1x1 convolution in our transition layer. The key advantage of this approach is that we strategically downsample features within the fine-scale branch. In this way, we optimize feature processing and facilitate a seamless flow of information within the current scale. The transition layer ensures that the fine-scale features are properly prepared before being fed into the current scale.

An efficient stem layer. The stem layer plays a crucial role in improving the efficiency of the architecture, as it is the key element responsible for effectively extracting essential features from the input data. Fig. 1 illustrates our proposed stem layer, which consists of four successive convolutional layers. It starts with a 3x3 standard convolutional layer to capture initial patterns, followed by a 3x3 Depthwise Separable Convolution layer to extract spatial information. The 3rd layer uses a 1x1 convolution to compress and refine the features. To improve the model's ability to capture important patterns and structures, the fourth layer uses a 3x3 Depthwise Separable Convolution (DW-Conv) with a step size of 2 to downsample the feature maps, making it highly suitable for our specific dataset. In addition, the stem incorporates a residual connection between the output of the

first convolutional layer and the output of the third convolutional layer. This connection allows the model to retain and propagate essential information, promoting effective feature extraction and overall performance improvement.

Moreover, the architectural design includes a dedicated pair of convolutional layers designed exclusively for the classifier function, each boasting 128 output channels. The initial layer in this pair conducts a 3×3 convolutional process, smoothly incorporating a downsampling stride of 2. Following this, the subsequent layer applies a 1x1 convolution with a stride value set at 1. This configuration is subsequently succeeded by the introduction of adaptive average pooling, a technique adeptly employed to reshape the spatial attributes into a streamlined 1x1 framework.

3. Experiments

We experimentally validate the effectiveness of our method on two widely recognized image classification datasets: CIFAR-10 and CIFAR-100, and compare our performance with state-of-the-art architectures, namely MSDNet [12] and RANet [30]. To ensure a fair comparison, the experimental settings adopted are consistent with those described in the original MSDNet and RANet papers. We also include two other competing models, namely $ResNet^{MC}$ and $DenseNet^{MC}$ [13]. While we do not give ablative comparisons due to space limitations, each of the two contributions (the fusion layer and the stem layer) result in performance gains of the same order of magnitude. The stem serves the dual purpose of extracting initial valuable features while efficiently reducing floating point operations (FLOPs). In addition, the merging process plays a critical role in significantly improving the overall classification performance, making it a key component in this context. We stress that during the evaluation of the model's performance, it's important to take into account that each baseline is duplicated in order to achieve its performance through the utilization of the existing implementation.

Datasets. The CIFAR-10 and CIFAR-100 datasets consist of 32 × 32 natural RGB images and include 10 and 100 classes, respectively, with each dataset containing 50,000 training images and 10,000 test images. To ensure consistency with previous studies [12], we specifically selected 5,000 images from the training set to form a validation set. This validation set played a crucial role in our research, as it allowed us to fine-tune and optimize the parameters of our method, ultimately identifying the optimal confidence threshold required for adaptive inference. By carefully validating our approach, we were able to improve its performance and generalization capabilities, thereby producing more accurate and reliable results in real-world image classification scenarios.

Training and Data Augmentations. We train the proposed models for 300 epochs using stochastic gradient descent (SGD) with an initial learning rate of 0.1. After 20 epochs of linear warm-up, the schedule transitions to cosine decay. We use a batch size of 512, a momentum of 0.9, and a weight decay of $1e^4$. For data augmentation, we follow the approach described in [7], which involves randomly cropping images to 32×32 pixels after zero padding (4 pixels on each side). In addition, images are flipped horizontally with a 50% probability, and RGB channels are normalized by subtracting their respective channel mean and dividing by their standard deviation. To further improve performance, we integrate popular schemes such as Mixup [33], Cutmix [32], and network regularization with label smoothing [24].

Experiments on Anytime prediction These experiments highlight a model's ability to make predictions with varying degrees of accuracy and computational cost. In traditional image classification tasks, a model processes an input image and generates a single class prediction. In contrast, in an anytime prediction scenario, the model predicts at different computational stages, with each prediction becoming more accurate as additional computational resources are allocated. The classification accuracies are shown in Figure 2. The evaluation includes three classifiers: MSDNet, represented by a black line, RANet, represented by a yellow-green line, and our method, shown in yellow. While MSDNet and RANet show similar performance, RANet performs better when computational resources are limited. However, our model performs significantly better on both the CIFAR-10 and CIFAR-100 datasets. For CIFAR-10, our model achieves an impressive 94.1% accuracy for the last classifier, requiring 32% fewer FLOPs than RANet's 93.8% last classifier, and 15% fewer FLOPs than MSDNet's 93.6% last classifier. Moreover, when the computational budget ranges from 0.25×10^8 FLOPs to 0.64×10^8 FLOPs, the accuracies of the various classifiers in our method consistently exceed those of MSDNet or RANet by 0.5% to 1%. In the case of CIFAR-100, our model achieves a remarkable 76.3% accuracy for the last classifier, using 32% fewer FLOPs than RANet's 74.28% last classifier, and 15% fewer FLOPs than MSDNet's 74.3% last classifier. Again, the accuracies of the various classifiers in our method outperform those of MSDNet or RANet by 1% to 2% when the computational budget is in the range of 0.25×10^8 FLOPs to 0.64×10^8 FLOPs.

Budgeted Batch Classification Experiments. This approach efficiently processes image batches within a predefined computational, memory, or time budget. It divides the batch into smaller subsets and processes them sequentially with different computational resources. Partial predictions for each subset are aggregated to produce the final batch predictions. This strategy enables reasonably accurate classifications with limited resources, making it valuable for applications on resource-constrained devices or systems. Figure 3 shows the results obtained with the budgeted batch

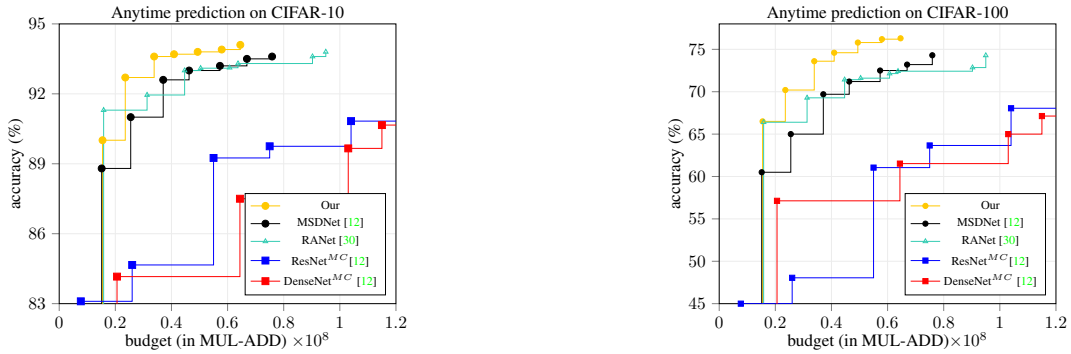

Figure 2: Accuracy (top-1) of anytime prediction models as a function of computational budget on CIFAR-10 (left) and CIFAR-100 (right). Higher is better.

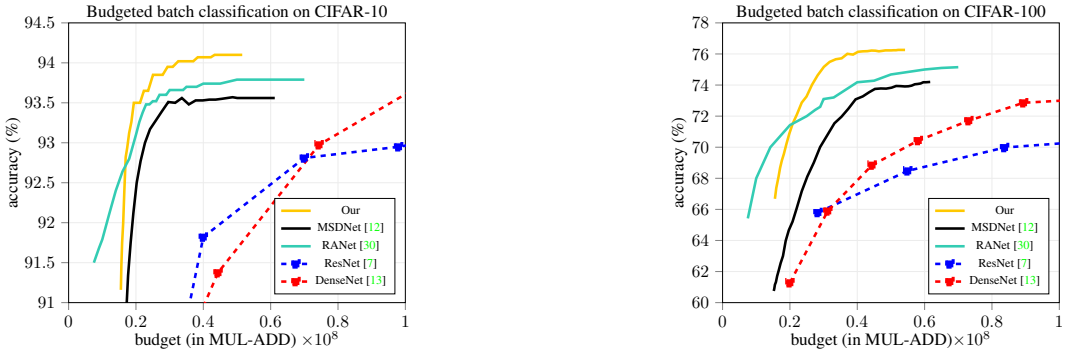

Figure 3: Accuracy (top-1) of budgeted batch classification models as a function of average computational budget per image on CIFAR-10 (left) and CIFAR-100 (right). Higher is better.

classification setting. To identify the optimal model for each budget, we evaluate its accuracy on the test set and plot the corresponding curves for MSDNet, RANet, and our model. For both CIFAR datasets, our model consistently outperforms MSDNets, RANets, and other baseline models for all budgets, except for RANet's low flops performance, which slightly outperforms ours. In particular, networks with a multiscale dense connection architecture consistently achieve significantly higher accuracy than other baseline models with equivalent computational cost, underscoring the strengths of our approach in the budgeted batch classification setting. For computational budgets above 0.2×10^8 FLOPs on CIFAR-10, our proposed model requires 32% fewer FLOPs to achieve a classification accuracy of 93.5% compared to MSDNet and 15% fewer FLOPs than RANet. Similarly, on CIFAR-100, our model achieves a classification accuracy of 74.01% with only about 53% fewer FLOPs than MSDNet and 34% fewer FLOPs than RANet. While RANet and MSDNet perform similarly on CIFAR-10 within the computational budget range of 0.15×10^8 to 0.5×10^8, our model outperforms them, requiring only 0.4×10^8 to reach 94%. On CIFAR-100, the classification accuracies of our model consistently exceed those of MSDNet and RANet by 1% to 2% in the median and high budget intervals (over 0.2×10^8 FLOPs). Furthermore, our model achieves an accuracy of 94.1% when the budget exceeds 0.4×10^8 FLOPs, outperforming MSDNet and RANet by 0.5% and 0.3%, respectively, under the same computational budget conditions. In addition, the experiments show that our model is up to 5 times more efficient than ResNets on both CIFAR-10 and CIFAR-100 datasets. This efficiency further underscores the superiority of our proposed model in the context of budgeted batch classification settings.

4. Conclusions

We have proposed a resource-efficient neural architecture for image classification based on MSDNet. Preliminary results show that this multi-exit design excels in anytime and budget batch classification, achieving state-of-the-art performance on CIFAR10 and CIFAR100. Key contributions include a novel feature fusion block and an efficient stem block. Our approach, which seems promising for real-world scenarios with limited resources, still needs to be validated on more challenging and diverse tasks.

Acknowledgments. Research reported in this paper was supported by the ANR under award number ANR-19-CHIA-0017.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 1

[2] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. 2018. 1

[3] Minghong Gao. A survey on recent teacher-student learning studies, 2023. 1

[4] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[5] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2015. 1

[6] Yizeng Han, Yifan Pu, Zihang Lai, Chaofei Wang, Shiji Song, Junfeng Cao, Wenhui Huang, Chao Deng, and Gao Huang. *Learning to Weight Samples for Dynamic Early-Exiting Networks*, pages 362–378. 11 2022. 1

[7] 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 (CVPR)*, June 2016. 1, 3, 4

[8] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[9] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. 1

[10] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[11] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. 1

[12] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018. 1, 2, 3, 4

[13] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 3, 4

[14] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. 1

[15] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016. 1

[16] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1

[17] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 1

[18] xu lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. 1

[19] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[20] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1

[21] Lassi Meronen, Martin Trapp, Andrea Pilzer, Le Yang, and Arno Solin. Fixing overconfidence in dynamic neural networks. *CoRR*, abs/2302.06359, 2023. 2

[22] Mary Phuong and Christoph H. Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[23] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 525–542, Cham, 2016. Springer International Publishing. 1

[24] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 3

[25] 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

[26] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. 2021. 1

[27] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks, 2017. 1

[28] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. 2018. 1

[29] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers, 2023. 1

[30] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 3, 4

[31] Le Yang, Haojun Jiang, Ruojin Cai, Yulin Wang, Shiji Song, Gao Huang, and Qi Tian. Condensenet v2: Sparse feature reactivation for deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3569–3578, June 2021. 1

[32] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 3

[33] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 3

[34] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 1