NASERex: Optimizing Early Exits via AutoML for Scalable Efficient Inference in Big Image Streams

Aakash Kapoor
Cornell University
New York, USA
ak2247@cornell.edu

Rajath Elias Soans
Merck Research
Atlanta, USA
soans@merck.com

Soham Dixit
University of Southern California
Los Angeles, USA
dixitsoham@gmail.com

Pradeep NS
Qualcomm
Bangalore, India
pns@qti.qualcomm.com

Brijraj Singh
Sony Research
Bangalore, India
brijraj.singh@sony.com

Mayukh Das
M365 Research, Microsoft
Bangalore, India
mayukhdas@microsoft.com

Abstract*—We investigate the problem of smart operational efficiency, at scale, in Machine Learning models for Big Data streams, in context of embedded AI applications, by learning optimal early exits. Embedded AI applications that employ deep neural models depend on efficient model inference at scale, especially on resource-constrained hardware. Recent vision/text/audio models are computationally complex with huge parameter spaces and input samples typically pass through multiple layers, each with large tensor computations, to produce valid outputs. Generally, in most real scenarios, AI applications deal with big data streams, such as streams of audio signals, static images and/or high resolution video frames. Deep ML models powering such applications have to continuously perform inference on such big data streams for varied tasks such as noise suppression, face detection, gait estimation and so on. Ensuring efficiency is challenging, even with model compression techniques since they reduce model size but often fail to achieve scalable inference efficiency over continuous streams. Early exits enable adaptive inference by extracting valid outputs from any pre-final layer of a deep model which significantly boosts efficiency at scale since many of the input instances need not be processed at all the layers of a deep model, especially for big streams. Suitable early exit structure design (number + positions) is a difficult but crucial aspect in improving efficiency without any loss in predictive performance, especially in context of big streams. Naive manual early exit design that does not consider the hardware capacity or data stream characteristics is counterproductive. We propose NASERex framework that leverages Neural architecture Search (NAS) with a novel saliency-constrained search space and exit decision metric to learn suitable early exit structure to augment Deep Neural models for scalable efficient inference on big image streams. Optimized exit-augmented models perform $\approx 2.5\times$ faster having $\approx 4\times$ aggregated lower effective FLOPs, with no significant accuracy loss.

***Index Terms*—DNNs, Early Exits, Image Streams, AutoML, NAS**

I. Introduction

Deep Neural Networks (DNNs) are essentially stacked transformation functions (layers) that generate progressively complex features/encoding which makes them universal approximators and allows for unprecedented success in complex tasks. This inferential effectiveness comes at the cost of increased computational complexity, making them hard to scale for operational efficiency in AI applications, especially when running on resource constrained hardware. Existing research on compute/power/size-efficient models [9], [21]–[23] focus primarily on model architecture/parameter space compression via quantization, pruning or distillation [1], [2]. However, in reality, operational efficiency of AI-driven applications/features powered by deep neural models pivots on not just lighter models but also on how the models can adapt to input signals. Incoming signals for most AI-driven applications or features, running either on-edge or on-cloud platforms, tend to be streams of instances or samples. For example, smartphone camera background blurring models running at the preview stage receive streams of image frames.

Operational efficiency in real machine learning systems is not only subject to latency over a single inference/execution run but instead depends more on tail latency and/or throughput for inference workloads, scaled over extended periods of time. This operational efficiency is indicative of the expected user experience of real machine learning applications. Traditional model compression strategies such as pruning, quantization etc., focus on reducing the architecture/parameter space and optimizing tensor operations. These techniques typically improve latency for isolated runs but tend to be ineffective for improving efficiency at scale in extended inference workloads with data streams. A promising alternative is to capitalize on the fact that not all instances in a stream are of similar complexity and while some instances might require end-to-end computation across all DNN layers to compute the output, others might be able to make a reasonable estimate of output after transformation over just a few layers. For instance, image streams or video frames often contain irrelevant frames [25] or frames of varying complexity that may either not offer useful insights relevant to the use-case or not require all the complex computations across all the layers to produce meaningful output. Such adaptive execution on data streams

979-8-3503-2445-7/23/$31.00 ©2023 IEEE

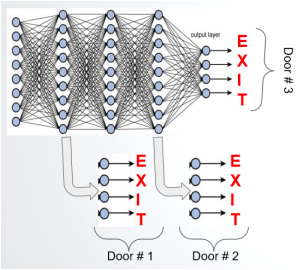


Fig. 1: Exit gates at terminal and/or non-terminal layers.

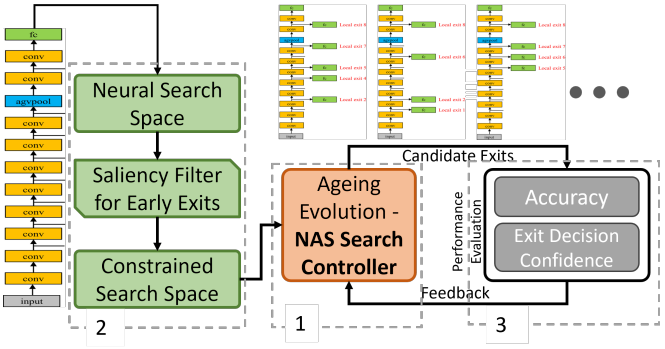


Fig. 2: Overview of our proposed NASEREX framework

has the potential to reduce aggregate/tail latency or throughput over data streams.

Early Exits [3], [6], [14], [15], [20], [24] enable adaptive inference by allowing us to obtain outputs of a given input signal from any pre-final layer of a DNN. This reduces information flow bottlenecks in DNNs leading to reduced aggregate inference latency without adversely affecting predictive performance on big streams of samples/instances over a period of time. For instance FrameExit [3] processes fewer video frames for simpler videos, via early exits based on frame complexity, for efficient video processing. This is even more critical for modern large language models [19] since their size makes their usage prohibitive in rapid streams of text. However, in this work, to ensure clarity of motivation and presentation, we focus primarily on big image streams scenarios that include sequence of images, video frames, etc.

Obtaining structures for early exits (feasible number of exits and their suitable placement at one or more DNN layers) is a key factor towards task-aware operational efficiency. Fig. 1 illustrates a general idea of how exit gates can be placed in a model. As can be clearly seen, the number of exit gates placed along with the location where they are placed can significantly impact both a model's predictive performance as well as its operational efficiency. This necessitates careful design of exit gates. However, manual design of exits is not possible at an industrial scale. ANNExR [16] studies the challenges of automated early exit design. We propose NASEREX to automatically learn optimal early exit structure via Neural Architecture Search (NAS)/AutoML[1] [13], [26]. Employing NAS, especially on pre-trained base architectures, is not trivial. Several challenges exist, including designing a manageable search space and obtaining a robust / efficient scoring metric that accounts for output uncertainty/confidence while balancing additive complexity due to exit structures. While different NAS approaches address the architecture learning problem in various ways, be it first order optimization (RL or evolutionary algorithms) on discrete search spaces [26] or efficient gradient based optimization on differentiable search spaces [13], none of them directly fit our problem setting of task-aware learning of exit structure. We make the following contributions - (1) A method to augment base DNN model to generate space of candidate early exit structure, (2) NAS framework to optimize the number and placement of exits, (3) A robust metric to regulate early exit decisions on input stream samples, and (4) Tractable implementation strategy for our proposed framework to seamlessly work at an industrial scale.

II. Methodology

A. Problem Setting

Given a base model architecture, we aim to automatically learn optimal early exit structure (number and positions) to augment the base model, and effectively maximize inference efficiency on big image stream tasks without loss in accuracy.

Let the base model M with at most $|\ell| = \mathcal{N}$ layers. In many recent models, however, layers may not always be sequential, and can have parallel paths. For clarity of presentation we assume each layer has a unique index $\ell_j : 1 \leq j \leq \mathcal{N}$. We aim to learn exit gates $\mathcal{G} = \{G_i\}_{i=1}^{n}$; s.t. $0 \leq n \leq \mathcal{N}$ and a mapping function $\mathcal{G} \rightarrow \ell : \mathcal{F}(i) = j$ which determines the position of the exits to generate an augmented model $m \in M'$[2]. For a streaming image sample $D = d_1, \ldots d_k, \ldots, d_T$, for some time window T, our optimization problem can be formally presented as,

$$\arg\min_{\Phi} \mathbb{E}_D[Latency(m)] + \mathbb{E}_D[|M(d_k) - m(d_k)|] \quad (1)$$

where $\Phi = (n, \mathcal{G} \rightarrow \ell)$ is the tuple of optimizable dimensions that represent the space of exit gate structures. Note that this is not a straightforward optimization problem. We need to jointly optimize over a count variable (n) and mapping function $\mathcal{F}(i) = j$ with a multi-criteria objective. There is no closed form solution and gradient-based optimization is also not feasible since we are operating in the (non-numeric) architecture/design space of DNNs. Also, it is subject to the base architecture M that is being augmented as well as the big image stream D. Thus we propose to address this problem using a Neural Architecture Search (NAS) approach that leverages zero-order optimization over a hybrid search space of early exit structures. Our framework (Fig. 2) **NAS**-designed **e**a**r**ly **Ex**its, NASEREX, has 3 main components: (1) NAS pipeline, (2) Design of architecture for exit augmentation, and (3) Novel real-time early exit decision criteria.

[1] AutoML and NAS are used interchangeably throughout this paper

[2] M' is the set of all feasible exit-augmented models

B. Problem & Solution Framework Components

1) Neural Architecture Search: Given a base model M, we augment it by providing tuneable hyper-parameters θ and architectural parameters α, to generate the augmented model $m_{\alpha,\theta} \in M'$. An important thing to note here is that exit gate structure space (from Eqn. 1) $\Phi \subseteq \alpha$, since there is no restriction against optimizing other architectural parameters along with the exit gates. Our NAS problem can be summarized using Eqns. 2–4

$$w^{opt}_{\alpha,\theta} = \arg\min_{w} L(m_{\alpha,\theta}(w)) \tag{2}$$

$$\alpha^{opt}, \theta^{opt} = \arg\max_{\alpha,\theta} S(m_{\alpha,\theta}(w^{opt}_{\alpha,\theta})) \tag{3}$$

$$M^{opt}_{\alpha,\theta} = m_{\alpha^{opt},\theta^{opt}}(w^{opt}_{\alpha,\theta}) \tag{4}$$

where $m_{\alpha,\theta} \in M'$ is a candidate augmented model sampled from the set of all possible augmented models M', L is the training loss and S is the scoring metric to estimate performance of trained candidate augmented models. Our architecture search explores α, θ values over a tractable discrete search space. Candidate model architectures are trained and scored, with the scores being used as a reward in our AgingEvolution-based NAS controller [17] to generate an optimal model architecture, denoted by $M^{opt}_{\alpha,\theta} \in M'$. AgingEvolution works by randomly initializing a pool of models from the discrete search space mutates them using the scores as fitness function to iteratively select better pools of models. Optimality is guaranteed, subject to the level of exploration, but may converge early.

2) Architecture Design: We need to consider two aspects of model architecture, the base architecture that forms the backbone of our exit-augmented model and the exit gate placements themselves. Our optimization problem is independent of any specific baseline model feature which alleviates any restriction on the choice of backbone. Since the primary purpose of early exits is faster inference, we note that – **(1)** Exit gates need to be reasonably simple to ensure that they do not significantly add to computational complexity of the base model itself. **(2)** Exit gate outputs need to be of the same shape as the backbone output. **(3)** Exit gates can, in theory, be placed anywhere but the complexity of modern DNNs precludes us from feasibly implementing this naively with discrete search spaces.

[SALIENCY CRITERIA]: Discrete search spaces grow at an exponential rate, especially if formulated as binary decisions, which will not scale well in our setting. To keep the search space polynomial, factors that grow with logarithmic complexity are used in place of binary decisions. Formally this reduces to the problem of ranking model nodes $\{\mathcal{N}\}$ by a metric, and choosing $O(\log|\mathcal{N}|)$ nodes. In practice, however, a heuristic approach of manually selecting a criteria that allows suitable coverage of the search space, suffices. For example, given the highly residual nature of the models (II-D) in our study, we decided to choose our saliency criteria such that only the locations where the residual connections merge into the main branch were considered for our search space.

3) Early Exit Decision Strategy: The learned exit-augmented model $M_{\alpha,\theta}$ relies upon confidence measurement bounds for exit decision at any given exit for a given input signal. Our goal, post-training, is to estimate decision confidence bounds by the use of predictor functions shown below

$$T_{low} = f_{low}(C_0, C_1, ..., C_K);\ \ T_{high} = f_{high}(C_0, C_1, ..., C_K) \tag{5}$$

where f_{low} is the predictor that yields the lower confidence threshold T_{low}, f_{high} the upper confidence threshold T_{high}, and K is the number of exit gates augmented to the model. C_i is the confidence score at exit gate i. The predictor functions may either be deterministic or stochastic. Universal approximators such as DNNs may be used as predictor functions. In this paper, we learn Gaussian regressors (parameterized with mean and variance) as predictor functions. Estimation of model uncertainty and decision confidence is critical to regulating the exit gates. Softmax outputs as proxies for model confidence is not reliable in real scenarios since the nature of relationship between softmax outputs and model confidence is unknown. Therefore, we adopt softmax with temperature scaling [5] which softens the DNN outputs in a manner that better represents true probability distributions.

C. Algorithm

Algorithm 1 outlines our NASERREX framework. Firstly, we generate a manageable search space from base model via saliency. AgingEvolution controller draws models as per fitness from our search space with a certain exploration bias. Each model drawn is trained, calibrated, and used to obtain the threshold values (T_{low}, T_{high}) for early exit decisions. At validation, an input sample D is passes through the model until an exit gate is encountered. At the exit gate, we compute the confidence score which, if higher than T_{high}, generates positive decision, or, if lower than T_{low}, a negative decision for D. If no decision can be made, execution proceeds along the model until the output is obtained either form an early exit or the final layer. This ensures that we consider both highly confident positive predictions (for inputs where target is easy to predict) and highly confident negative predictions (for out-of-distribution samples). As is clear, earlier exits favour lower latency while later exits favour better accuracy. This trade-off can be controlled by the use of appropriate predictor functions. In addition to the algorithmic details, we outline the tractable implementation choices, next, which make NASERREX usable in practice at an industrial scale.

D. Tractable Implementation of NASERREX

Our tractable implementation has 4 major aspects : (1) A practically feasible search space design for Neural Architecture Search, (2) Designing the exit gate architecture on the basis of suitable baseline models chosen, (3) Choosing a suitable criterion for exit decisions, and (4) Implementing an effective training pipeline that serves our objectives. We illustrate the same with ResNet50 [7] and MobileNetV2 [18] as base architectures.

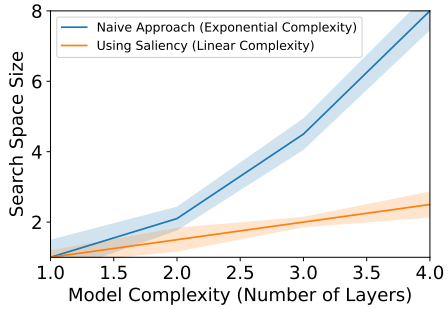


Fig. 3: Search Space Complexity with and without saliency

Algorithm 1 Optimized Early Exit via Efficient NAS

Require: $M \leftarrow$ Base Model, $D_T, D_V \leftarrow$ Dataset
function NASEREX(M, D_T, D_V)
  $P_M \leftarrow$ GetSearchSpace(M) ▷ P_M is the Search Space
  $P_C \leftarrow$ GenerateModelPool(P_M) ▷ Initial Model Pool
  while P_M is not empty **do**
    $P_M \leftarrow P_M - P_C$; $R \leftarrow 0$
    for $M_{\alpha,\theta}$ in P_C **do**
      $M^*_{\alpha,\theta}, T_{high}, T_{low} \leftarrow$ TRAIN($M_{\alpha,\theta}$)
      $Score \leftarrow$ VALIDATE($M^*_{\alpha,\theta}, T_{high}, T_{low}$)
      if $Score > R$ **then**
        $R \leftarrow Score$; $M_{opt} \leftarrow M_{\alpha,\theta}$
      end if
    end for
    $P_C \leftarrow$ AgingEvolution(P_M, R)
  end while
  return M_{opt}
end function
function TRAIN($M_{\alpha,\theta}$)
  while Convergence **do**
    $O_M, O_E \leftarrow M_{\alpha,\theta}(D_T)$ ▷ Final/Early Exit Output
    $T_{high} \leftarrow f_{high}(O_E)$; $T_{low} \leftarrow f_{low}(O_E)$
    $M^*_{\alpha,\theta} \leftarrow$ Backward($M_{\alpha,\theta}$) ▷ Backward Pass
    for i in N_E **do** ▷ N_E : Number of Exits
      $M^*_{\alpha,\theta} \leftarrow$ Backward(M^i_s)
    end for
  end while
  return $M^*_{\alpha,\theta}, T_{high}, T_{low}$
end function
function VALIDATE($M_{\alpha,\theta}, T_{high}, T_{low}$)
  $O_M, O_E \leftarrow M_{\alpha,\theta}(D_V)$ ▷ Model/Exit Output
  if Confidence(O^i_E) $> T_{high}$ **then**
    $O \leftarrow O^i_E$
  end if
  if max(Confidence(O^i_E)) $< T_{low}$ **then**
    $O \leftarrow O_{neg}$
  end if
  $Score \leftarrow$ Accuracy(O, Pred)
  return $Score$
end function

1) Search Space: In our setting, the search space is essentially encoded as an array of binary choices between placing an exit gate or not at a given position. However, in this scheme the search space may grow exponentially with the number of potential positions for placing the exit gates, illustrated in Fig. 3. We solve this problem by identifying and exploiting salient points (outlined in section II-B2) present in our base architectures. So, in practice we get $O(\log|\mathcal{N}|)$ exit points. $|\mathcal{N}|$ indicates number of layers, i.e. unique places we can locate our exit gate in the PyTorch implementation of the given models. Thus, given $|\mathcal{N}| = 175$ layers for ResNet50 and $|\mathcal{N}| = 152$ for MobileNetV2, without saliency, the size of the search space would be $\approx 2^{175}$ and $\approx 2^{152}$, however, with saliency it is reduced to $\approx 175k$ and $\approx 152k$ respectively (where k is a simple linear scaling factor).

2) Exit Gate Structure: The most basic building block within the MobileNetV2 and ResNet50 architectures is a sequence of Convolution, Batch Normalization, and Activation layers. We choose this as a refinement block for our exit gate. Furthermore, we allow our NAS controller to decide how many of these sequences to add per exit gate. The collection of these sequential blocks, however, do not provide the correct output shape. Hence, the outputs from the sequential layers are further processed by pooling, followed by flattening the tensors and then adding a fully connected layer to map the flattened tensors to the correct output shape.

3) Exit Decision Criterion: The backbone/base model outputs are augmented with Temperature controlled Softmax to ascertain a measure of confidence for class selection. Using temperature helps improve the reliability of using the softmax classifier by making it confidence-calibrated [11]. The temperature value is typically set to 1 in traditional softmax usage but in our experiments, we set it to a higher value of 5, empirically, to smooth out the softmax distribution. Since our task is multi-class classification, we use cross-entropy loss along with L1 regularization on our model (and exit gate) outputs for training. During training, we keep track of the per-class output values from the base model output which we pass to the predictor functions, formulated as $\mu_c + 0.5\sigma_c$, where μ_c is the mean and σ_c is the standard deviation of the output vector for class c, for the upper threshold predictor function.

4) Training Details: We define a sub-model $M^{(sub)}_i$ as the set of layers beginning from the input nodes of the backbone model and culminating in the output layer of the i^{th} exit gate. Each candidate sub-model needs to behave as close to the main backbone as possible, within limitations of its respective complexity, to provide effective inference. To ensure this, our pipeline trains each sub-model with as much independence as possible, with each sub-model backpropagating in an interleaving fashion rather than parallel, shown in Algorithm 1. This is essential in order to avoid race conditions during model weight updates.

III. EXPERIMENTS

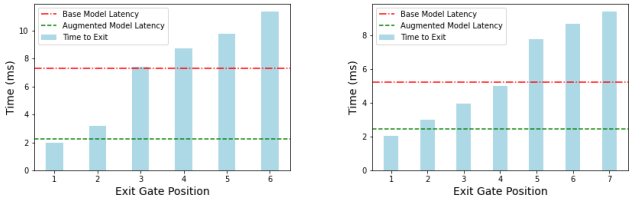
A. Experimental Settings

We evaluate our exit-augmented architectures against the backbone architectures as baselines on 2 types of vision problems with big data streams, image classification and video classification. *Image classification experiments are sort of a control experiment to demonstrate that our method is effective (lower expected latency) even on static image datasets that can be executed in batch workloads. Video classification task highlights the primary claim of the paper about scalable operational efficiency on big image stream inference tasks.*

Backbones: Note that the backbone models we have evaluated in this early work are traditional CNN-based architectures for vision tasks. This choice is motivated by the clarity of CNN

Dataset	ResNet50 : Base		ResNet50 : NASERex	
	Accuracy (%)	Latency (ms)	Accuracy (%)	Latency (ms)
CIFAR-10	82.73	7.27	**85.28**	**1.78**
CIFAR-100	**51.95**	7.32	24.57	**2.82**
CalTech-101	63.50	7.40	**78.07**	**1.43**
CalTech-256	26.58	7.34	**35.23**	**2.41**
Dataset	MobileNetV2 : Base		MobileNetV2 : NASERex	
	Accuracy (%)	Latency (ms)	Accuracy (%)	Latency (ms)
CIFAR-10	81.08	5.22	**84.14**	**2.24**
CIFAR-100	**51.61**	5.62	49.76	**2.17**
CalTech-101	50.61	5.25	**61.52**	**2.83**
CalTech-256	**21.01**	5.51	10.2	**2.41**

TABLE I: Performance Comparison of Exit-Augmented Optimal Architectures with Base Architectures.



(a) **ResNet50 Backbone.** (b) **MobileNetV2 Backbone.**

Fig. 4: **Latency Profile at Exit Points for augmented models.**

architectures even with parallel execution paths. This allows us to demonstrate the impact of intelligently learned early exit structure on aggregate latencies over image streams. However, our proposed method can potentially construct exit-augmented model with respect to any kind of backbone architectures, including transformers.

For image classification we use CIFAR-10 [10], CIFAR-100 [10], CalTech-101 [12], and CalTech-256 [4] using an NVIDIA TESLA P40 GPU server. Datasets chosen represent problems with different complexities. For example, CIFAR-100 and CalTech256 are complex problems to solve, especially for the architectures of our choice. We also evaluate NASERex on phantom videos with irrelevant frames added to it in a controlled manner. The ratio of irrelevant frames is varied and the improvement in latency savings is characterized. The video frames consisted data from imagenette [8] for MobileNetV2, and cifar10 for ResNet50 along with custom-frames with low entropy.

B. Results

1) Image Classification: Table I shows the accuracy and latency comparisons between the backbone baselines and NASERex-augmented models. We observe negligible drop in overall validation accuracy, with significant reduction in average inference latency. Augmented models are scored on test datasets based on the accuracy they provide while functioning as they would during deployment.

2) Video (Big Image Streams): Table II shows how NASERex-designed exit-augmented models have lower ex-

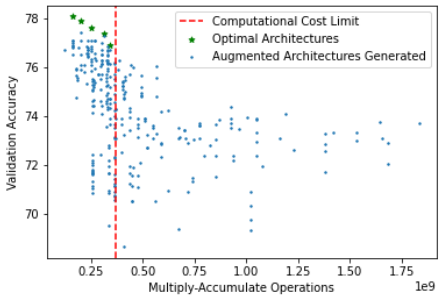


Fig. 5: Pareto-Optimality Boundary of NAS Search. The computational costs, (Multiply-Accumulate Operation) vs accuracy of exit augmented models. (Cost cut-off indicated in red.)

Architecture	Base Model Expected Latency l_b (s)	Irrelevant Frames	Augmented Model Expected Latency l_a (s)	Speed-Up $(\frac{l_b}{l_a})$
ResNet50	70.19	0%	22.35	3.14
		10%	21.95	3.19
		30%	21.12	3.32
		50%	20.54	3.41
		75%	20.73	3.38
MobileNetV2	54.67	0%	24.63	2.21
		10%	24.45	2.23
		30%	22.92	2.38
		50%	22.31	2.45
		75%	21.32	2.56

TABLE II: Inference latency gain on a frame-by-frame analysis of a simulated video with Augmented architectures.

pected cumulative latency on simulated video frames. Observe the substantial speed-up even with low percentage of irrelevant frames. This proves that it is not just irrelevant frames that exit out early, but frames of varying complexity exit at appropriate gates leading to overall efficiency.

Figs. 4a & 4b illustrate the latency profiles with respect to the exit positions for ResNet50 and MobileNetV2 backbones. We have observed that the optimal placement of exit gates for MobileNetV2 favour the latter part of the model due to high complexity in training data. However, ResNet50 has simpler datasets which allowed the low entropy frames to exit early.

C. Layer Pruning

Furthermore, we also propose layer pruning on exit-augmented models, motivated by an observation that the desired accuracy might be achieved at a certain non-terminal exit gate in the model and the layers succeeding to that gate can be removed if needed. Table III presents the results of of our study. After an optimal NASERex generated model is obtained, we remove layers after every exit point and record the overall accuracy each tsime. Our results show that this method may be leveraged to yield architectures with up to $4\times$ fewer FLOPs and $2.5\times$ lower latency at competitive accuracies. The average inference latencies also show how, for simpler tasks such as CIFAR-10 highlighted in 4a and 4b, our model is able to maintain competitive accuracy while

Model	Accuracy (%)	Latency (s)	FLOPS (M)
Base	81.08	5.22	6.4
NASERX optimized	83.4	2.35	9.9
Exit 1	82.62	2.09	2.4
Exit 2	83.07	2.10	3.2
Exit 3	83.36	2.11	3.8
Exit 4	83.44	2.15	4.6
Exit 5	83.57	2.16	7.6
Exit 6	83.45	2.14	8.9
Exit 7	83.31	2.11	9.9

TABLE III: **Model Compression by layer pruning**. Results obtained using MobileNetV2 on CIFAR-10 data at each Exit Gate by removing all the succeeding layers.

exiting at the first exit for a large number of image samples. Thus, based on the target problem complexity and choice of the user, designers have the option to choose subset of layers to be pruned that balance accuracy and efficiency needed for the use case.

D. Pareto Optimality & Limitations

Given sufficient exploration, pareto-optimality in the space of accuracy versus model complexity is an implicit outcome of NASERX, as early exits by nature enhance expected operational efficiency and our saliency-guided search space design further improve the odds of identifying the right candidate(s). The novel learned exit decision criteria, on the other hand, takes care of accuracy frontier. As illustrated in the scatter plot (Fig. 5), we observe a high density of exit-augmented candidates very close to the true optimal structures.

A major limitation, however, is that the degree of exploration over the search space versus computational cost of NAS search is a delicate balance and with limited training resources shared between data loading of large data sets and search algorithm, exhaustive exploration is challenging.

IV. CONCLUSIONS

In this paper, we have proposed a NAS framework to learn optimal early exit structure, thereby, providing an automated way to enable task-aware efficient adaptive inference for any backbone model on big image streams. While our proposed method is conceptually generic for all types of models and tasks (discriminative and generative), extending the implementation of the framework such that any developer/designer can generate exit augmented networks along with post-pruning for arbitrary model types and data sets as well as extensive evaluation are some of the key objectives in our on-going and future efforts.

ACKNOWLEDGEMENT

AK, RS, SD, PNS, BS and MD gratefully acknowledge Samsung R&D Institute, Bangalore (SRIB) for its support during initial stages of the work (when the authors were employed there). MD gratefully acknowledges Microsoft for the support and funding towards continuance of this work and for conference participation.

REFERENCES

[1] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proc. IEEE*, 107(8):1655–1674, 2019.
[2] Yunbin Deng. Deep learning on mobile devices: a review. In *Mobile Multimedia/Image Processing, Security, and Applications 2019*, volume 10993, page 109930A. International Society for Optics and Photonics, 2019.
[3] Amir Ghodrati, Babak Ehteshami Bejnordi, and Amirhossein Habibian. Frameexit: Conditional early exiting for efficient video recognition. In *CVPR*, 2021.
[4] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. *CalTech Report*, 03 2007.
[5] 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, 2017.
[6] Amirhossein Habibian, Davide Abati, Taco S Cohen, and Babak Ehteshami Bejnordi. Skip-convolutions for efficient video processing. In *CVPR*, 2021.
[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
[8] FastAI Jeremy Howard. The imagenette dataset.
[9] Ziheng Jiang, Tianqi Chen, and Mu Li. Efficient deep learning inference on edge devices. *ACM SysML*, 2018.
[10] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
[11] Meelis Kull, Miquel Perelló-Nieto, Markus Kängsepp, Telmo de Menezes e Silva Filho, Hao Song, and Peter A. Flach. Beyond temperature scaling: Obtaining well-calibrated multiclass probabilities with dirichlet calibration. In *NeurIPS*, 2019.
[12] Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato. Caltech101 image dataset. *CalTech Report*, 2003.
[13] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
[14] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *arXiv preprint arXiv:2103.04505*, 2021.
[15] Nikolaos Passalis, Jenni Raitoharju, Anastasios Tefas, and Moncef Gabbouj. Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. *Pattern Recognition*, 105:107346, 2020.
[16] Annapurna P Patil, Rajarajeswari Subramanian, Varun Cornelio, S Venkatesh, M Varun, K Shavin, Mayukh Das, and NS Pradeep. Annexr: Efficient anytime inference in dnns via adaptive intermediate decision points. In *IntelliSys*, 2022.
[17] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. In *AAAI*, 2019.
[18] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.
[19] Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *NeurIPS*, 2022.
[20] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, pages 2464–2469. IEEE, 2016.
[21] Marian Verhelst and Bert Moons. Embedded deep neural network processing: Algorithmic and processor techniques bring deep learning to iot and edge devices. *IEEE Solid-State Circuits Magazine*, 9(4):55–65, 2017.
[22] Mário P Véstias, Rui Policarpo Duarte, José T de Sousa, and Horácio C Neto. Moving deep learning to the edge. *Algorithms*, 13(5):125, 2020.
[23] Sahar Voghoei, Navid Hashemi Tonekaboni, Jason G Wallace, and Hamid R Arabnia. Deep learning at the edge. In *CSCI*, 2018.
[24] Meiqi Wang, Jianqiao Mo, Jun Lin, Zhongfeng Wang, and Li Du. Dynexit: A dynamic early-exit strategy for deep residual networks. In *SiPS*, pages 178–183. IEEE, 2019.
[25] Jiaojiao Zhang, Kunqian Li, and Wenbing Tao. Multivideo object cosegmentation for irrelevant frames involved videos. *IEEE Signal Processing Letters*, 23(6):785–789, 2016.
[26] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.