Project Proposal: Automatic Early Stopping for Improving the Efficiency of Pre-Trained Image Classification Models

Sanie, Blake
bsanie3@gatech.edu
Small, Dylan
dylansmall@gatech.edu
Mace, Dylan
d.mace@gatech.edu
Chemelli, Austin
austin.chemelli@gatech.edu
Wong, Anthony
awong307@gatech.edu
Zacharis, Nicolas
p nzacharis@gatech.edu

November 16, 2023

1 Introduction

When training an image classification model, one must consider the inherent variance in the complexity of an image dataset. For example, in the case of classifying dogs, some images may be an obvious depiction of a dog. However, some images may only show the dog's face. The first case requires significantly less computational power, yet traditional models will spend the same amount of resources on classifying it. This is where early stopping can become extremely valuable. Early stopping involves the placement of classifiers at certain locations within an image classification model. When classifying an image, a confidence value is computed and compared to a pre-determined, human-appointed threshold value at each early stopping classifier. If the confidence value meets or exceeds this threshold, then further computation for that image can cease. This highlights the power of early stopping as it saves computational resources without losing accuracy.

Early stopping can provide a huge cost-saving benefit; however, current capabilities may be improved. We can consider the human-appointed thresholds at each early stop as hyperparameters. Directly training such hyperparameters instead of relying on human experimentation will enable the optimization of the early stopping within image classification models.

2 Motivation & Objectives

Pre-trained Image Classification Models are highly complex, and can have millions (and sometimes billions) of parameters. When a model of this size is under heavy load in production, progressing through all layers of the model can be slow, which could leave a negative impact on the end-user. Early stopping largely solves this problem, but there are a number of motivations for us to surpass the current unsolved problems in this domain. Three main problems arise: how many early exit layers to insert, where to insert new early exit layers, and how to teach a model to utilize the new layers. Largely, these problems are currently being solved using human intervention [1]. Our focus for this project revolves around solving the third problem: training a model to optimize the use of early exit layers. To do this, we will introduce learned parameters into the model layers that will decide the criteria for early stopping within these chosen layers. We will use PyTorch's autograd algorithm to train these parameters and optimize the stopping criteria as a task in itself. This will result in a model which naturally learns to utilize the new exit layers on an image-by-image basis. If additional time remains after completing our initial objective, we will explore inserting early exit frameworks into every model layer; given our model will learn which layers to exit from, this would resolve the other remaining issues with the early exit paradigm.

3 Related Work

In 2017, a team of researchers from Harvard University created the first colloquial early exit neural network architecture: BranchyNet [2]. The BranchyNet team utilized a pre-trained set of networks and pioneered the concept of early exit via entropy predictions. If the entropy of classification at a pre-defined early exit layer is deemed above a specified threshold, the model chooses the exit and returns the calculated prediction layer (via softmax) [2]. If the threshold is not met, the model continues through the pipeline. This, in conjunction with joint optimization, allows the model to automatically leave via an early exit layer without user interference. The largest issue with the BranchyNet architecture is that it requires a manual threshold to be set for early exiting. This approach is inherently unoptimal as it requires large amounts of experimentation to define the optimal threshold for each network the policy is applied to. Noted in the conclusion, the team recommended the exploration of meta-recognition algorithms to automatically define network improvement.

The most prominent innovation regarding where to place early exit layers stems from a 2021 issue of Applied Soft Computing, in which Vanderlei Bonato and Christos-Savvas Bouganis propose a new form of early-exit algorithm designed to identify prominent locations for early exit layers [3]. In addition, the work explores dense connections between early-exit branches and pre-trained base model layers for simultaneous forward propagation to allow for increased efficiency while maintaining baseline accuracy. The implementation also focused on scalability, aiming to mitigate effort required when applying the new method to larger

networks with a multitude of layers. This approach, however, also relies on a confidence threshold for taking an early exit layer; thus, while it may be more efficient than BranchyNet with improved early exit layer locations, it still requires fine-tuning via human experimentation and intervention.

A completely different approach is taken in the 2023 development of EENet, created via a joint endeavor between Georgia Tech and CISCO research [1]. This research aimed to remedy the hand-tuned exit strategies discussed in the prior works. EENet instead focuses on the concept of an inference budget, allotting a predetermined amount of flops for computation via early exiting. This new approach utilizes the novel idea of early exit utility scores, a composite metric combining the confidence of each layer's early exit prediction score with the class-wise prediction scores [1]. At validation time EENet learns the optimal early exit policy per class given a specified inference budget. This approach is completely model-agnostic, solving the need for manual classification thresholds. The model can also be parallelized, leading to increased performance across edge computing devices. The primary drawback of this method is the test-time early exit policy design; the approach still relies on thresholds, whether learned by class or not. This approach also assumes each class can be predicted in a similar manner with similar inference thresholds, which may not be the case.

At this point, there is no early exit policy capable of adapting to each individual image content and determining the proper exit location, completely without human intervention. We propose a novel approach capable of learning the optimal early exit strategy for any individual image, regardless of image class or manual thresholds.

4 Solution Approach & Engineering Choices

To achieve truly learnable exit layers, a module wrapper class, titled EarlyExit, abstracts logic to computationally define the proper exit strategy and classification at a given model layer. Each EarlyExit layer is thus responsible for its own exit and classification logic, avoiding erroneous propagation of irrelevant gradients and cross-contamination between exit nodes. This allows each layer to decide if an exit should be taken, and consequently either take the exit (immediately perform classification) or continue with forward propagation according to the base model's architecture.

Each EarlyExit instance creates a new set of parameters to fine-tune as training progresses. These parameters support the following transformation modules, with flattened input size n and a m possible output classes:

1. Gate Layer(s): a $\mathbb{R}^n \rightarrow \mathbb{R}$ Linear transformation (effective dot product) condensing all immediate layer inputs into a single scalar. Outputs at-least zero signify the exit should be taken, otherwise, the forward propagation continues across base model descendant layers.

2. Classification Layer(s): a $\mathbb{R}^n \rightarrow \mathbb{R}^m$ Linear transformation that performs immediate classification. Layer outputs are logits (before softmax) corresponding to each class, with higher values representing higher class prediction probability. This output could also be the result of a softmax layer, with the value at each index corresponding to the probability of the input image belonging to the aforementioned class.

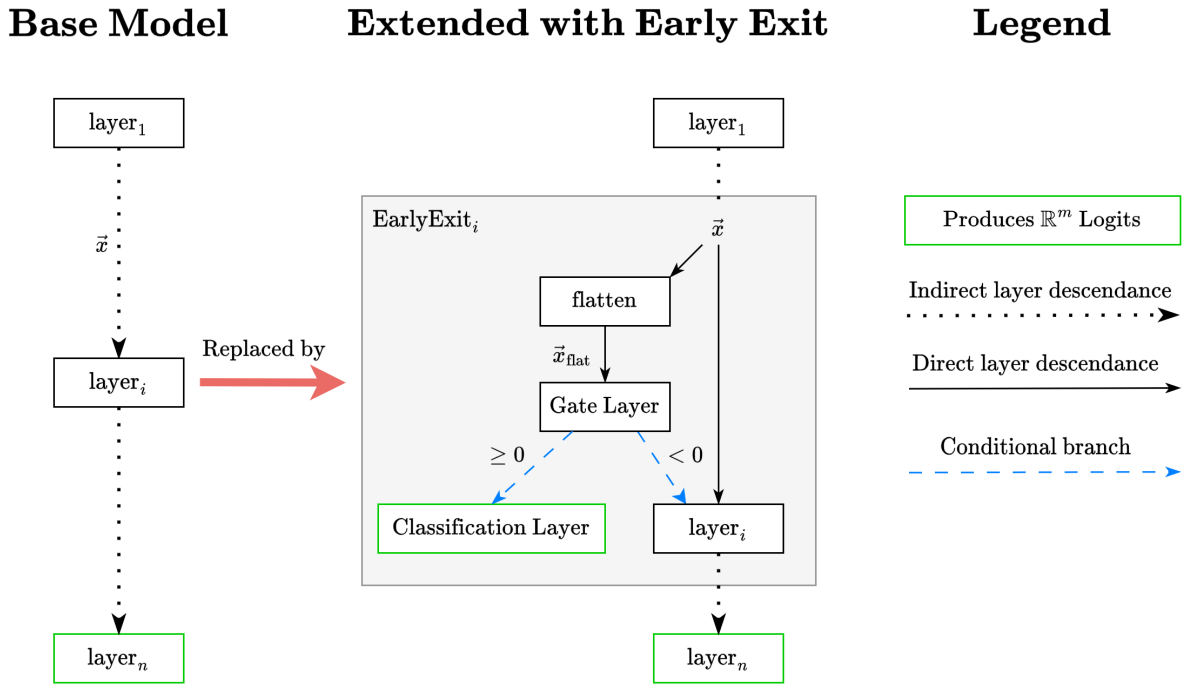

Figure 1: Graphical representation of EarlyExit module.

Note that the internal logic of the EarlyExit module could also take the design of a single $\mathbb{R}^n \rightarrow \mathbb{R}^{m+1}$ linear projection, with m resulting logits and 1 value reserved as an exit decision. However, we opt to leverage a two-step design to minimize computational overhead above base model levels. That is, only the n-size dot product is required to yield an exit decision, thus the nxm matrix multiplication is only applied if guaranteed to be useful. This is in contrast to computing the matrix multiplication in any case, only to discard the classifications if the exit is skipped.

To implement our proposed framework, we will leverage the PyTorch machine learning library for Python [4]. Doing so will enable direct access to state-of-the-art base models, such as Resnet50 [5], as well as increased access to relevant datasets. These resources will serve as our base on top of which early exit features will stand. Secondly, PyTorch's Auto-Grad function heightens the feasibility of our method's development, since we can focus on the logical definition of each module's forward propagation process with out-of-the-box gradient management based on a dynamic computation graph. PyTorch's included evaluation methods and callback hooks allow for a smaller and more simple training code structure, allowing for an increased focus on EarlyExit module speed and efficiency.

5 Implementation Plan

Items in bold are scheduled events

Date	Task
10/04 (4:15pm)	**Meeting with Professor about Proposal**
10/11	Set up base model and dataset
10/25	Build EarlyExit layer and integrate with model
11/8	Run training iterations and make initial evaluations
11/14 (2:20pm)	**Workshop Presentation**
11/20	Make improvements based on findings
11/27	Perform in-depth analysis and comparisons of results to base model
12/4	Write final report and prepare for demo presentation
12/5 (3:30pm)	**Project Demo, Report**

6 Evaluation and Testing

To comprehensively evaluate the performance of our newly developed dynamic early stopping model in terms of both efficiency and accuracy, we will conduct a series of comparative tests against the original Resnet50 network [5]. Our evaluation strategy consists of the following:

1. **Efficiency Benchmarking:** To gauge the efficiency gains of our model, we will create multiple image test sets of increasing sizes. Subsequently, we will assess the runtime of both our novel model and Resnet50 on these test sets at each dataset size [5]. By comparing the execution times between the two models and analyzing the rate at which runtime increases with dataset size, we will quantify the efficiency improvements achieved by our model.

2. **Accuracy Assessment:** To determine the impact of our architectural modifications on accuracy, we will calculate the percentage of correctly classified images for each model within each test set. This will allow us to establish a comparative accuracy profile, highlighting the performance differences resulting from our alterations to the Resnet50 architecture [5].

3. **Layer Depth Accuracy Monitoring:** Additionally, we will track the accuracies within our new model with respect to how far test images traverse through its layers. This investigation will help us ascertain whether we can maintain high accuracy levels despite compromising on computational power.

This comprehensive approach will provide a holistic understanding of the strengths and weaknesses of our dynamic early stopping model in comparison to the original network, of course with the goal of producing a minimal accuracy drop while maximizing efficiency [5].

References

[1] F. Ilhan, L. Liu, K.-H. Chow, W. Wei, Y. Wu, M. Lee, R. Kompella, H. Latapie, and G. Liu, "Eenet: Learning to early exit for adaptive inference," 2023.

[2] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," 2017.

[3] V. Bonato and C.-S. Bouganis, "Class-specific early exit design methodology for convolutional neural networks," *Applied Soft Computing*, vol. 107, p. 107316, 2021.

[4] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.