Routing Beats Latency

Anonymous authors
Paper under double-blind review

Abstract

The deployment of deep neural networks that operate under hard latency constraints has to compromise accuracy against speed, especially where the input difficulty is widely varying in practice. Although the high-capacity Transformers like DeiT-S-16 have good robustness, the inference cost cannot be deployed to the edges as it is prohibitive. Lightweight CNNs such as SqueezeNet on the other hand gain high throughput but in severe blur and noisy behaviour they struggle for accuracy. Our phase-based study indicates that heuristic measures of complexity such as entropy-based difficulty predictor ($R^2 = 0.112$) and complexity scores on CIFAR-10 derived by ICNet-9600 are not useful routing cues because of low variance and poor predictive structure.

To address these shortcomings, we propose a framework of Adaptive Mixture of Experts (MoE) that achieves train dense route sparse inference with a lightweight SqueezeNet-Light Gater. The Gater works at downsampled inputs 96×96 and launches a Gumbel-Softmax straight-through estimate which has only 0.82 ms overhead, and just a latency-aware composite loss with a trade-off parameter to be traded at (referred to as λ) allows the system to jointly optimise the accuracy and predicted inference cost.

Experiments of Intel Image Classification data (augmented with controlled Gaussian blur and noise) indicate that the system finds a dynamic Pareto frontier. When the router is configured to use $\lambda = 0.7$, the router puts 98% of clean images in the fastest expert, and switches to 90% DeiT routing with harsh corruption, reflecting human interpretable patterns of difficulty. It reaches 82.71% accuracy at an average of 5.39 ms latency - which is a speedup of approximately $\sim 29\%$ over DeiT-S-16 (7.56 ms) with only a small drop in robustness across different levels of corruption. We have shown, using our results, that latency-aware differentiable routing is both more efficient and adaptive than both static CNNs and Transformers in terms of accuracy-latency Pareto frontiers.

1 Introduction

Inference latency verses model correctness: When using deep neural networks in resource-constrained conditions, this trade-off is important to consider carefully, as can be achieved through efficient methods such as edge-based model optimization and graph connectivity errors (Jiang et al., 2018; Zhang et al., 2023). Although compact convolutional models (Iandola et al., 2016; Howard et al., 2017) are low-latency and their performance largely depends on the quality of their inputs, high-capacity Vision Transformers (Dosovitskiy et al., 2020; Vaswani et al., 2017) are not vulnerable to noise and corruption (Paul & Chen, 2021; Zhou et al., 2022) and carry a significant computational cost, which is why they cannot be used in latency-sensitive tasks To offer sound, effective inference of edge devices, however, it is now essential to create an adaptive system that allocates computational means based on the difficulty of the input (Han et al., 2021; Liu & Chen, 2021).

A close phase-based exploration of the issues of estimating image difficulty to select a dynamic model motivates our work. Early work considered heuristic methods using an output entropy of the expert models. A CNN that was trained to predict the entropy, based on the predictions of MobileNetV2, EfficientNet, and ViT, scored an extremely low R^2 of 0.112, and the correlation of entropy with genuine difficulty was weak. This predictor repeatedly sent images to the least-expensive experts and never successfully predicts instances that should be sent to high-capacity models, which proves that entropy, although a popular measure of uncertainty,

is not a good predictor of the routing decisions. To label the images in CIFAR-10 we next used the specific complexity-scoring model ICNet-9600. In spite of the fact that ICNet-9600 gave coherent outputs, data has limited complexity diversity, scores were centered around a mean of about 0.193 with a median of about 0.195. This limited scope did not allow significant difficulty gradients: the overwhelming majority of images were equally simple, and supervised routing was not that. The inability of both entropy-based prediction and ICNet-based complexity scoring pointed out a core weakness in heuristic-based and externally-supervised routing mechanisms the estimation of difficulty has to be a part of the optimization process and not enforced externally.

This knowledge inspired the switch to an end-to-end learnable routing model. In an attempt to create significant input variability, we based our experiment on the Intel Image Classification dataset and applied targeted degradations, including Gaussian blur and Gaussian noise at four levels of severity, including clean, mild, moderate, and severe, to all of us (Hendrycks & Dietterich, 2019). These degenerations produced a spectrum of natural difficulties in which expert models varied. Light CNNs like SqueezeNet (Iandola et al., 2016) and MobileNetV2 significantly deteriorated in accuracy when subjected to moderate and severe corruption, whereas transformer-based models like DeiT-S-16 models as well as ViT-B-16 models could be comparatively robust to exploits: lightweight convolutional models could be effective in dealing with easy samples, whereas challenging ones demanded the resistance of transformers.

Based on these observations, we created an adaptive Mixture of Experts framework (Jacobs et al., 1991; Shazeer et al., 2017), using the principle of Train Dense, Route Sparse. The first modules of SqueezeNet are used to produce a downsampled (96x96) variant of every single image into a lightweight (downsampled) version known as a Gater. In order to make routing differentiable, the Gater uses Gumbel-Softmax Straight-Through estimator (Jang et al., 2017; Maddison et al., 2017), so that when using the decision-time model, categorical one-hot expert selection can be done during inference, and back-propagated through soft samples during training. Even the experts, such as SqueezeNet, DeiT-S-16, and MobileViT-S are stored as a frozen model during the training to facilitate optimization stability and routing behavior isolation.

The main element of the system is a composite loss function that directly includes latency, which overcomes the challenge of high computational costs associated with the deployment of deep learning approaches. The Gater is trained to reduce a weighted combination of expert cross-entropy loss and expected latency, as a function of a trade-off scalar, λ. The three models utilized in this work have dramatically different computational costs SqueezeNet: 2.14 ms per image, DeiT-S-16: 7.56 ms, MobileViT-S: 10.59 ms an optimistic space of router operates richly and does not have to choose between reliability and efficiency.

The empirical assessment reveals that the system acquires interpretation and adaptive routing behaviors. At a physical layer of $\lambda = 1.0$, routing is focused nearly on DeiT-S-16, which is as accurate as the most reliable expert. In intermediate values, like when you have a value of the router like 0.8, he will start assigning the easiest images to SqueezeNet and transformers are applied to the uncertain or corrupted images. Within the range of 0.7, the system has a highly latency-biased behavior: the router routes approximately 98 percent of clean examples to SqueezeNet, and compressed with severe corruption, the router routes about 90 percent of images to transformers, which is a real sign of difficulty discrimination coming out of the optimization goal itself.

The system resulting sets a new Pareto revenue between accuracy and latency. It has an overall accuracy of 82.71 percent at a latency of 5.39 ms at a single layer of 0.7, which is about 29 percent lower than when DeiT-S-16 is used exclusively (7.56 ms). Notably, the overhead of making decisions is also negligible (the Gater cost is just 0.82 ms on its own), whereas the benefits of directing easy images to faster experts are substantial. These findings indicate that latency-aware differentiable routing represents a conceptually sound and practically efficient framework of adaptive inference, which serves as a realistic way to achieve efficient, robust and deployment-ready vision models that can both meet real-time demands as well as remain highly effective when faced with adversarial environments.

2 Related Work

2.1 Vision Transformers and Efficient Architectures

ViTs are transformers designed to adapt the transformer architecture (Dosovitskiy et al., 2020) to image data, using the mechanism of self-attention to image data. Nonetheless, the computational cost excludes resource-constrained deployment due to their prohibitive cost. Data-efficient training procedures and distillation techniques have also been added to enhance ViT training efficiency in DeiT (Touvron et al., 2021), and later supervised training recipes have been optimized in supervised training in DeiT (Touvron et al., 2022). Recent works have found that Vision Transformers are resistant to corruptions in nature and that self-attention mechanism and better mid-level representations explain the property in part, however, not entirely, due to their robustness to corruptions. Fire modules enable parameter efficiency in lightweight architectures like SqueezeNet, which has since been used extensively to deploy on the edge (Iandola et al., 2016) and MobileViT, which uses convolutional and transformer operations to generate models useful on mobile devices (Mehta & Rastegari, 2022a;b). These architectures are not only used in our work as heterogeneous experts in an adaptive routing model and use their best features where their contribution is more beneficial in different input challenges.

2.2 Dynamic Neural Networks and Adaptive Inference

Dynamic neural networks are a new paradigm, which customizes computation depending on the characteristics of inputs (Han et al., 2021). Methods of adaptive inference involve predictions at intermediate layers of the network (early-exit methods) (Scardapane et al., 2020; Bolukbasi et al., 2017; Teerapittayanon et al., 2016), and adjusting the depth/width of the network on the fly (Wang et al., 2018). Passalis et al. (2020) came up with hierarchical early-exit layers that have adaptive inference abilities, which showed the achievement of better efficiency-accuracy trade-offs. Hor et al., in their article on adaptive inference, give theoretical limits on the efficiency gains that can be attained with adaptive inference, showing that efficiency can be 10-100x higher with adaptive inference based on input distribution and model architecture. In contrast to early-exit methods, which are implemented using a single architecture, we base our work on independent expert models with significantly different architectural characteristics, making it possible to have more diversity in the space of accuracy-latency trade-offs and input-dependent computations available to it (Figurnov et al., 2017).

2.3 Mixture of Experts

Mixture of Experts (MoE) architectures (Jacobs et al., 1991; Jordan & Jacobs, 1994) separate the learning tasks into specific sub-networks and the selection of the experts is done by the gating mechanisms. The article by Shazeer et al. (2017) trained MoE to 137B parameters in language models using sparsity and showed that selective expert activation allows them to scale to large models with a manageable computational cost. The recent activity has used MoE to vision problems in the context of scaling up to all-state-scale (Riquelme et al., 2021) and examined other gating mechanisms to gain a better specialization. Nevertheless, current MoE systems are fundamentally based on the scaling model capacity instead of inferential latency optimization. We specifically include latency in our routing objective and learn to trade-off accuracy and computational cost by means of a differentiable composite loss, in contrast to the conventional MoE techniques that only optimize the prediction accuracy.

2.4 Differentiable Routing Mechanisms

The issue of discrete expert selection has to be overcome in order to have differentiable routing in neural networks. A categorical sampling is one that has been relaxed to form the Gumbel-Softmax trick (Jang et al., 2017; Maddison et al., 2017), which allows discrete choices to be optimized through gradient-based optimization. The straight-through estimator permits discrete forward passes but enables differentiable backward passes, a method which has been shown to be successful in a range of discrete optimization problems. This method is applied to the routing problem in our work, but is unique in the sense that we use the measured values of latency as direct inputs in the loss term, which allows the system to discover cost-sensitive routing policies without the added complexity of problem labels and hand-tuned policies.

2.5 Model Compression and Deployment

The necessity to deploy deep learning models on edge devices has driven intensive research on model compression methods down the road to efficient implementation of these models on edge devices and smartphones in particular (Zhu et al., 2024; Liu et al., 2025). Quantization-based numerical precision reduction techniques (Jacob et al., 2018; Nagel et al., 2021) to accelerate inference and reduce memory needs can be used, e.g., by using quantization-friendly training methods to allow models to maintain accuracy with low-bit quantization, e.g., through quantization-aware training approaches to training, such as quantization based on stochastic quantization and quantization based on AD inference methods. Knowledge distillation can be used to transfer knowledge stored in large teacher networks to small student architectures with large compression rates and performance levels (Hinton et al., 2015; Gou et al., 2021). The reduction of computational complexity (Han et al., 2015; Liu et al., 2019) is performed using pruning strategies that are built on the elimination of redundant parameters, such as learning, rethinking, and learning, and so on. These methodologies however optimise single models to be deployed. Our adaptive routing model offers an alternative set of solutions: maintaining many specialised models, and dynamically choosing between them based on the input complexity in order to achieve efficiency gains without indefinitely reducing model capacity.

2.6 Corruption Robustness Benchmarks

Vision models have adopted robustness against typical corruptions as an important evaluative criterion (Hendrycks & Dietterich, 2019). The ImageNet-C benchmark presented standardised types of corruption such as noise, blur, weather conditions, and digital distortions, which proved that the robustness of a model does not necessarily increase with capacity (Hendrycks et al., 2021). Corruption robustness research has been extended to include person re-identification (Chen et al., 2021) as well as semantic segmentation and video understanding tasks. Modern studies have concluded that the reduction of sensitivity to patch corruptions can significantly improve the robustness of transformers (Guo et al., 2023; Yu et al., 2024). Our designed process of corruption, which applies to the Intel Image Classification data, produces a difficulty continuum that is controlled and allows the router to learn meaningful allocation strategies to clean and corrupted inputs.

3 Methodology: The Differentiable Gater System

The proposed framework is an end-to-end differentiable Mixture of Experts (MoE) designed to allocate computation adaptively according to input difficulty. The system integrates a heterogeneous expert pool with a lightweight routing network, trained under a latency-aware composite loss. In contrast to heuristic or manually imposed criteria of difficulty, the methodology seeks to guarantee that expert selection arises from optimization dynamics.

3.1 Heterogeneous Expert Pool

The system consists three pretrained vision models whose computational and robustness characteristics differ substantially. This combination of varying architecture is essential for enabling a meaningful routing landscape. The first expert is SqueezeNet (Iandola et al., 2016), a compact convolutional based architecture optimized for inference efficiency. Its forward pass latency is approximately 2.14 ms, and it exhibits strong performance on clean, structurally simple scenes but deteriorates sharply as corruption severity increases. On the Intel dataset, its accuracy declines from roughly 90% on clean inputs to nearly one-third of that performance under severe Gaussian blur or noise. This pronounced decay forms the low-latency–low-robustness end of the expert spectrum.

MobileViT-S 2 is the second expert, which is a hybrid model, i.e. it consists of both convolutional feature extraction and transformer-style global attention (Mehta & Rastegari, 2022a). Its latency of 10.59 ms puts it at the lowest point of the pool and it has a moderate robustness against corruption. MobileViT-S is a competitive model on clean images, but similarly to most hybrid models, it suffers non-negligible degradation

on severe perturbations. It serves the purpose in the pool as an intermediate capacity substitute where the lightweight CNN is not enough, but it is not needed to call the entire transformer.

The high-robustness choice is the third one, which is DeiT-S-16 (Touvron et al., 2021). It is a refined Vision Transformer that does not drop its performance when the degradation is large. Its latency of 7.56 ms is larger than SqueezeNet but remains much lower than MobileViT-S. DeiT-S-16 is the most resistant to corruption, with about 70–72% accuracy in severe blur and noise. The professionals therefore determine a precise gradient of computation: SqueezeNet generates speed, MobileViT-S generates capacity, and DeiT-S-16 generates robustness in distributional shift (Hendrycks & Dietterich, 2019).

This organized diversity helps a router to acquire meaningful expert preference patterns. Frozen weights of expert models leads to fixed and predictable accuracy-latency profiles, which create a solid optimization foundation.

3.2 The Train Dense, Route Sparse Paradigm

The system follows a "Train Dense, Route Sparse" design philosophy. During training, routing decisions are differentiable, and gradients propagate through all experts indirectly via the routing distribution. This dense training mode ensures that the router experiences the full range of expert behaviors, including the sensitivity of each model to corruption levels and semantic complexity.

Sparsity is enforced that means exactly one expert is executed per sample at the time of inference. Single-expert selection is essential for achieving the targeted latency improvements and for ensuring predictable computational cost per input. Downsampled version of input image has given to routing network to operate, which is a 96×96 resolution image, where only high-level structural cues are preserved. Whereas, the expert models always receive the full 224×224 resolution input, allowing them to operate at their intended capacity.

Let $x \in \mathbb{R}^{3\times224\times224}$ denote the input image, and let $\tilde{x} \in \mathbb{R}^{3\times96\times96}$ denote its downsampled version. The experts are denoted $f_i(x)$ for $i \in \{1, 2, 3\}$. A categorical selection variable z has produced by routing based on which one expert ultimately evaluated. The resulting inference latency is therefore

$$T_{\text{infer}}(x) = T_{\text{gater}} + C_{i^*},$$

where $T_{\text{gater}} = 0.82$ ms and $C_{i^*} \in \{2.14, 10.59, 7.56\}$ ms depending on the selected expert. This architecture guarantees that routing overhead is minimal relative to expert costs, while allowing the router to reduce total computation based on acquired difficulty.

3.3 Gater Architecture

The routing network has been built based on the first five modules of SqueezeNet 1.1, which include an initial convolutional layer, ReLU activation, max-pooling operation and two successive Fire modules. The choice maintains the lightweight property of the original model of feature extraction but dramatically decreases depth and parameterization. The Gater output is in the form of an embedding vector, which is achieved via nonlinear transformation and convolutional downsampling, which is denoted as h.

Formally, the Gater computes

$$h = g(\tilde{x}),$$

in which $g(\cdot)$ refers to the truncated SqueezeNet. The embedding is then guided by a linear classifier to an embedding of routing logits:

$$\ell = Wh + b,$$

with $W \in \mathbb{R}^{3\times d}$ and $b \in \mathbb{R}^3$. These logits are the unnormalized scores of the three experts. The Gater is small, but it is sufficiently able to encode the levels of corruption, edge distortions, and blur induced spatial information to distinguish between inputs that need high-capacity processing and those that can be processed in fast inference.

3.4 Differentiable Routing Mechanism

One of the main contributions of the methodology is the application of Gumbel-Softmax reparameterization, which allows the use of differentiable sampling of expert indices, assists in this task (Jang et al., 2017; Maddison et al., 2017). The conventional MoE structures are based on hard gating or hard credit assignment with reinforcement learning which both present significant optimization problems. Rather, the suggested system uses the Straight-Through Gumbel-Softmax estimator that maintains differentiability and imposes discrete expert selection at the forward pass.

Given the routing logits ℓ, the system draws Gumbel noise g_i for each expert, with

$$g_i \sim \text{Gumbel}(0, 1).$$

it is then made by soft routing distribution which is given by

$$y_i = \frac{\exp\left((\ell_i + g_i)/\tau\right)}{\sum_{j=1}^{3} \exp\left((\ell_j + g_j)/\tau\right)},$$

where τ is the temperature parameter which controls sharpness of the distribution. In forward pass, a hard one-hot vector is obtained by replacing y with one_hot($\arg\max_i y_i$), gradients are computed backward through the continuous variables y_i. A fixed temperature is used with the parameter of temperature set to $\tau = 1.0$, and it gives a constant balance between the searching and the converging behavior.

The complete training and routing algorithm is defined in Algorithm 1.

Algorithm 1 Latency-Aware Differentiable Routing (Forward & Loss)

Input: Image batch $X \in \mathbb{R}^{B\times3\times224\times224}$, labels Y_{true}
Input: Frozen experts $E = \{e_1, e_2, e_3\}$, latency vector $C \in \mathbb{R}^3$, Gater G
Input: Hyperparameter λ, temperature $\tau = 1.0$

// 1. Gater Processing (Low Resolution)
$\tilde{X} \leftarrow \text{Resize}(X, (96, 96))$ ▷ Downsample input
$\ell \leftarrow G(\tilde{X})$ ▷ Get logits from SqueezeNet-Light Gater

// 2. Differentiable Routing (Gumbel-Softmax)
$g \sim \text{Gumbel}(0, 1)$
$y_{soft} \leftarrow \text{Softmax}((\ell + g)/\tau)$
$k^* \leftarrow \text{argmax}(y_{soft})$
$y_{hard} \leftarrow \text{OneHot}(k^*)$
$z \leftarrow y_{hard} - y_{soft}.\text{detach}() + y_{soft}$ ▷ Straight-through estimator

// 3. Expert Execution (Train Dense)
for $i = 1$ **to** 3 **do**
    $\hat{y}_i \leftarrow e_i(X)$ ▷ Forward pass on full-resolution image
end for
$\hat{Y}_{final} \leftarrow \sum_{i=1}^{3} z_i \cdot \hat{y}_i$ ▷ Select expert output

// 4. Latency-Aware Loss Calculation
$L_{class}^{raw} \leftarrow \text{CrossEntropy}(\hat{Y}_{final}, Y_{true})$
$L_{class} \leftarrow L_{class}^{raw}/3.0$ ▷ Scale classification loss
$lat_{expected} \leftarrow \sum_{i=1}^{3} y_{soft,i} \cdot C_i$
$L_{latency} \leftarrow \text{Mean}((lat_{expected} - \min(C))/(\max(C) - \min(C)))$ ▷ Normalize [0,1]
$L_{total} \leftarrow \lambda L_{class} + (1 - \lambda) L_{latency}$
Return $L_{total}, \hat{Y}_{final}$

3.5 Composite Loss Function

Training of the routing network is based on the reduction of an objective of latency-aware composition, where the classification quality and the expected inference cost are brought to the same numerical level before the trade-off weighting is introduced. Let $y = (y_1, y_2, y_3)$ denote the Gumbel–Softmax soft routing probabilities produced by the Gater for a given input and let $f_i(x)$ be the prediction of expert i. The raw classification loss for the selected expert is the cross-entropy $L_{\text{class}}^{\text{raw}} = \text{CE}(f_{i^*}(x), y_{\text{true}})$, computed in the usual way; the implementation then rescales this raw loss to match the magnitude of the latency term by dividing by a constant factor s. Concretely, the code uses

$$s = 3.0$$

and defines the normalized classification loss as $L_{\text{class}} = L_{\text{class}}^{\text{raw}}/s$. This explicit scaling places L_{class} and the normalized latency term on comparable numerical ranges, preventing the optimization from being dominated by whichever term has a larger raw magnitude and making the hyperparameter λ operate as an interpretable trade-off coefficient in practice.

The latency term is computed as the expected expert latency under the soft routing distribution, using the measured per-expert costs $C = [2.14,\ 10.59,\ 7.56]$ ms. The expected latency per example is

$$E[C] = \sum_{i=1}^{3} y_i\, C_i.$$

Normalization to the unit interval of expected latency has performed using an affine mapping determined by the minimum and range of the observed expert latencies: $\hat{C}_{\text{norm}} = (E[C] - C_{\min})/(C_{\max} - C_{\min})$, where $C_{\min} = \min(C) = 2.14$ ms and $C_{\max} = \max(C) = 10.59$ ms (so the denominator equals 8.45 ms). Averaging over the batch yields the normalized latency loss $L_{\text{latency}} = \mathbb{E}_{\text{batch}}[\hat{C}_{\text{norm}}]$. This normalization also guarantees the numerical stability and that the latency contribution is in the range of $[0, 1]$ so it is commensurate with the scaled classification loss.

The final composite objective used for gradient updates is therefore

$$L_{\text{total}} = \lambda\, L_{\text{class}} + (1 - \lambda)\, L_{\text{latency}},$$

with $L_{\text{class}} = \text{CE}(f_{i^*}(x), y_{\text{true}})/3.0$ and $L_{\text{latency}} = \text{mean}_{\text{batch}}\big((\sum_i y_i C_i - 2.14)/8.45\big)$. The hyperparameter $\lambda \in [0, 1]$ thus directly controls the operating point: larger λ values emphasize accuracy (driving the router toward the robust DeiT-S-16), whereas smaller λ values favor latency minimization (biasing the router toward SqueezeNet). Because both terms are normalized prior to weighting, λ has a stable, interpretable effect across experimental runs and corruption settings.

This normalization design was introduced after observing scale imbalance between raw cross-entropy values and expected latency values; dividing the cross-entropy by a constant factor $s = 3.0$ and linearly normalizing the expected latency to $[0, 1]$ prevents either term from numerically overwhelming the other and allows the router to learn difficulty-sensitive routing policies without external difficulty labels. The latency vector $C = [2.14, 10.59, 7.56]$ ms is taken from the expert benchmarking reported in the project results, and the Gater's per-sample overhead (measured separately) is 0.82 ms.

4 Experimental Setup

4.1 Dataset and Engineered Complexity

All experiments are conducted on the Intel Image Classification dataset, chosen for its structural diversity and suitability for evaluating adaptive routing under varying difficulty. To introduce controlled complexity, each image $x \in \mathbb{R}^{3\times224\times224}$ is transformed through Gaussian blur and Gaussian noise, generating four difficulty regimes: clean, mild, moderate, and severe (Hendrycks & Dietterich, 2019). Blur is applied as $\mathcal{B}_\sigma(x) = x * G_\sigma$, where G_σ is a Gaussian kernel with increasing σ, while noise is applied as $\mathcal{N}_\gamma(x) = x + \eta$ with $\eta \sim \mathcal{N}(0, \gamma^2 I)$. Increasing σ and γ systematically induces degradation. This design produces a monotonic accuracy decay across experts, e.g., SqueezeNet drops from $\sim$ 90% accuracy on clean images to $\sim$ 31–33% under severe

corruption, whereas DeiT-S-16 retains $\sim$ 70–72%. Unlike CIFAR-10, where ICNet-9600 produced an extremely narrow complexity distribution (mean ≈ 0.193, median ≈ 0.195), the engineered severity levels here create a broad difficulty manifold essential for learning meaningful routing policies.

4.2 Implementation Details

The system is implemented in PyTorch and optimized using Adam with learning rate 1×10^{-4} and batch size 32. Expert parameters remain frozen; only Gater parameters θ_g are updated. Training minimizes a composite objective $L_{\text{total}} = \lambda L_{\text{class}} + (1-\lambda) L_{\text{latency}}$, where L_{class} is the cross-entropy loss using the selected expert's output, and $L_{\text{latency}} = \sum_i y_i \hat{C}_i$ uses the soft routing distribution y produced by Gumbel-Softmax (temperature $\tau = 1.0$) and the normalized latency vector $\hat{C}$ derived from raw expert costs $[2.14, 10.59, 7.56]$ ms. GPU latency is measured using synchronized CUDA events to ensure accurate per-sample timing. The Gater receives inputs downsampled to 96×96, resulting in an average overhead of 0.82 ms, while experts always receive the full-resolution input.

4.3 Evaluation Metrics

Classification accuracy and per sample inference latency are used to evaluate the system. Robustness and routing accuracy are determined by computing accuracy on clean and corrupted subsets. The reported latency consists of the Gater overhead and the inference time of the chosen expert, i.e., $T_{\text{total}} = T_{\text{gater}} + C_{i^*}$, where i^* which is the discrete route selected by Straight-Through Gumbel-Softmax. Mean latency $\overline{T}$ is calculated on the entire test set. This performance metric is a direct assessment of the accuracy-efficiency trade-off of various values of λ; for example, at $\lambda = 0.7$ the system achieves 82.71% accuracy at 5.39 ms average latency, reducing computation by $\sim$25% compared to running DeiT-S-16 alone (7.56 ms).

5 Results

This section shows the quantitative analysis of the suggested Gater based Mixture of Experts system. Findings are in the form of baseline expert performance, system behavior as a whole and a detailed dissecture of routing decisions as a function of corruption severities. All values computed below are simply rebuilt directly based on experimental summary tables.

5.1 Baseline Expert Performance and Gater Overhead

In order to determine reference operating points, the individual expert models were tested at all the levels of corruption. The resulting accuracy-latency profile, along with the measured overhead of the Gater, is reported in the table 1.

Model	**Avg Accuracy (%)**	**Avg Latency (ms)**
DeiT-S-16	86.72	7.56
MobileViT-S	72.57	10.59
SqueezeNet	62.06	2.14
Gater (router overhead)	–	0.82

Table 1: Baseline Expert Accuracy and Latency

These findings refer to a large difference in computational cost: SqueezeNet is being run in 2.14 ms, DeiT-S-16 is run in 7.56 ms, and MobileViT-S is run in 10.59 ms. This difference gives the gradient of computation that allows the Gater to find the latency-optimal routing plans.

5.2 Unified Performance Across λ: The Adaptive Accuracy–Latency Frontier

The composite loss also enables the router to find operating points that reflect the various accuracy-latency trade-offs. As the system is perturbed by changing the value of λ, the system will follow a dynamic Pareto

frontier. The overall performance of the entire dataset is replicated in table 2 under varying values of the variable of λ.

Model (λ)	**Avg Acc (%)**	**Avg Lat (ms)**	**%→Sq**	**%→Mob**	**%→DeiT**
Gater (0.95)	86.74	8.01	5.7	0.0	94.3
Gater (0.80)	86.72	7.89	10.5	0.0	89.5
Gater (0.90)	86.72	8.33	1.2	0.0	98.8
Gater (1.00)	86.58	8.38	0.0	1.4	98.6
Gater (0.70)	82.71	5.39	54.9	0.0	45.1

Table 2: Overall Gater System Performance (Whole Test Set)

As indicated by the table, at the point of receiving $\lambda = 1.0$, the router is dominated by the selection of DeiT-S-16 as they are the most powerful bases. The smaller the value of the λ at 0.70, the greater latency is minimized as routing is directed by SqueezeNet. The settings with the lowest average latency of 5.39 ms, about 35% lower than the settings with high accuracy, conserve the accuracy of 82.71% with the entire dataset.

5.3 Routing Behavior on Clean and Mild Corruptions

The routing distributions are also analyzed in the conditions of varying levels of corruption. To obtain clean images, the Gater would highly recommend the usage of the low-λ settings of SqueezeNet, but the high-λ numbers would provide certain deterministic routing to DeiT-S-16.

λ	Acc (%)	Lat (ms)	%→Sq	%→Mob	%→DeiT
0.80	94.37	7.57	14.7	0.0	85.3
0.95	94.33	8.05	5.6	0.0	94.4
0.90	94.30	8.26	0.0	0.0	100.0
1.00	94.30	8.38	0.0	0.0	100.0
0.70	90.53	3.07	98.0	0.0	2.0

Table 3: Routing on Clean Images

λ	Acc (%)	Lat (ms)	%→Sq	%→Mob	%→DeiT
0.80	94.43	7.68	12.2	0.0	87.8
0.95	94.43	8.08	5.7	0.0	94.3
0.90	94.40	8.29	0.0	0.0	99.9
1.00	94.40	8.38	0.0	1.4	98.6
0.70	88.00	3.53	90.5	0.0	9.5

Table 4: Routing on Mild Corruptions

A distinct structure is apparent in both regimes, where input is clean or slightly corrupted, classification loss is minimal in SqueezeNet, and the latency benefit holds the routing decision in lower λ in the regimes where the regimes are lower.

5.4 Routing Behavior on Moderate and Severe Corruptions

With moderate and high corruption level, the performance of SqueezeNet reduces drastically. This moves this loss-latency trade off, and the Gater prefers transformer-based models.

Under these circumstances, the router decreases its dependence on SqueezeNet and gradually transitions to DeiT-S-16 with a higher level of corruption even in latency-biased conditions. This corroborates that the routing mechanism is image difficulty sensitive and distributes computation on the same.

λ	Acc (%)	Lat (ms)	%→Sq	%→Mob	%→DeiT
0.90	87.43	8.30	0.0	0.6	99.4
1.00	87.43	8.30	0.0	0.4	99.6
0.95	87.40	8.21	5.6	0.0	94.4
0.80	87.30	7.89	7.9	0.0	92.1
0.70	82.17	7.19	21.7	0.0	78.3

Table 5: Routing on Moderate Corruptions

λ	Acc (%)	Lat (ms)	%→Sq	%→Mob	%→DeiT
0.80	70.80	7.95	7.1	0.0	92.9
0.95	70.80	7.95	5.7	0.0	94.3
0.90	70.77	8.39	0.0	4.4	96.0
1.00	70.17	8.41	0.0	1.2	98.8
0.70	70.13	7.78	9.5	0.0	90.5

Table 6: Routing on Severe Corruptions

5.5 Summary of Observed Routing Dynamics

The overall analysis of all the tables shows that the router has always learned a challenge-conscious strategy: it uses the low latency of SqueezeNet to learn clean and slightly corrupted inputs, and the DeiT-S-16 architecture in cases when the severity of corruption rises. Lower values of λ reveal an aggressive latency-saving mode, particularly $\lambda = 0.70$, which routes 98% of clean inputs and 90.5% of mildly corrupted inputs to SqueezeNet, lowering total latency to as little as 3.07 ms. Another extreme is that high value of the λ takes us to deterministic use of transformers.

This system thus provides a controllable and interpretable accuracy-latency trade-off, which indicates that differentiable routing can find operating points that any single expert network would have been unable to access.

6 Discussion

This can be seen through the results of the experiment which show a stark difference in heuristic complexity estimation and end-to-end differentiable routing. The predictor VIS_entropy of Phase 1 had an R^2 of just 0.112 and reduced to a degenerate routing policy that never activated the big-capacity expert as an example of the severe drawback of uncertainty-based heuristics: uncertainty does not equal difficulty, and it does not favor the computational advantage of calling upon the slower expert. The decisions of this nature do not encode decision-theoretic cost structure that is at the heart of dynamic model routing (Bolukbasi et al., 2017).

Conversely, the differentiable routing formulation is effective due to the fact that the goal that the router has to achieve is directly incorporated into the training process. GumbelSoftmax Straight-Through estimator (Jang et al., 2017) is a variant of the routing decision, which is converted to a differentiable portion of the computational graph, so the Gater can learn the trade off between classification loss and latency cost of various types of inputs. The routing policy is based on minimizing the composite loss as opposed to external labels of complexity or heuristic signals. The Gater does not just learn to see visually hard examples but to understand examples on which a slower model would make a significant contribution to the expected classification loss- a property that is naive to successful adaptive routing.

Corruption-dependent routing behavior analysis shows that the Gater internalizes such trade-off. At $\lambda = 0.7$, almost all clean samples ($\approx 98\%$) are sent to SqueezeNet, whereas severely corrupted samples ($\approx 90.5\%$) are sent to DeiT-S-16. This monotonic dependence on the degree of corruption of routing choices is a consequence of neither being labeled as noise or blur but learned implicitly since the loss of lightweight experts increases

with the degree of corruption and exceeds the latency cost of using a transformer. Thereby the system evolves a routing strategy that is economically rational: it spends little computational effort on easy inputs, and only in the case where the marginal cost of robustness is sufficiently high, does it spend extra computational effort (Bengio et al., 2015).

One key question in adaptive computation is a question of whether the costs of the routing mechanism are offsetting the payoffs of selective expert invocation (Hor et al., 2024). This concern is directly dealt with by the measured Gater overhead of 0.82 ms. The routing overhead is not significant in comparison with the available computational gradients because the slowest expert (MobileViT-S) needs 10.59 ms and the mid-size DeiT-S-16 needs 7.56 ms. Besides, since the Gater remains the same irrespective of the expert selected, its cost does not rise with sample challenge, fulfilling the most important condition of implementing it practically on resource outflanked edge devices in practice (Zhang et al., 2023; Jiang et al., 2018).

The unified performance results confirm this. The $\lambda = 0.7$ operating point delivers 5.39 ms average latency—approximately a 29% reduction relative to using the transformer alone—while still preserving 82.71% accuracy across clean and corrupted samples. This establishes a demonstrable Pareto improvement that no standalone model can replicate. The system does not merely interpolate between expert behaviors; it identifies an operating point that lies strictly outside the accuracy-latency envelope defined by the experts, confirming that adaptive routing offers computational benefits that are unattainable through static model selection or traditional compression techniques (Zhu et al., 2024; Hossain et al., 2023).

7 Conclusion

This paper shows that an adaptive computation of vision models can be efficiently obtained by using a differentiable Mixture of Experts which approximates routing decisions with the actual inference cost structure. Analysis based on phases indicated that the heuristic difficulty estimators, including entropy predictors and externally trained complexity scorers, are essentially poor in that they either have weak correlates between themselves and the actual routing utility, or lack enough complexity variance in the underlying data. Instead, the reason why the proposed system of Gater works is exactly due to the fact that its routing policy is optimized on an end-to-end basis with a composite objective that explicitly balances classification loss to expected latency.

By integrating a lightweight SqueezeNet-based Gater with a Gumbel–Softmax Straight-Through routing mechanism (Jang et al., 2017), the system discovers a controllable accuracy–latency Pareto frontier that no individual expert model can reach. Under the latency-oriented regime ($\lambda = 0.7$), the router assigns approximately 98% of clean inputs to the 2.14 ms SqueezeNet expert, reducing average latency to 5.39 ms while retaining 82.71% accuracy—a $\sim$ 29% improvement in inference speed relative to relying solely on DeiT-S-16. Conversely, for severely corrupted inputs, the router shifts decisively, sending $\sim 90.5\%$ of such samples to the more robust DeiT-S-16 (Touvron et al., 2021; Bai et al., 2021). This emergent difficulty-aware behavior arises without explicit corruption supervision, confirming that the composite loss enables the system to learn the economic value of additional computation.

The overhead of the Gater observed to be 0.82 ms is sufficient to show that the "cost of thinking" is not so high as to render it computationally inefficient to use selective routing even in tight latency budgets such as those found in edge deployment applications (Jiang et al., 2018). Collectively, these results give strong justification that the concept of differentiable routing, which is based on a principled latency-aware goal, provides a realistic approach to adaptive, efficient and robust visual inference. The resulting system provides the speed of CNNs with a lightweight and the resilience of transformers with hard samples, and it is resilient, a balance that can not be achieved with the static architecture.

References

Yutong Bai, Jieru Mei, Alan L Yuille, and Cihang Xie. Are transformers more robust than cnns. *Advances in Neural Information Processing Systems*, 34:26831–26843, 2021.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

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

Minghui Chen, Zhiqiang Wang, and Feng Zheng. Benchmarks for corruption invariant person re-identification. volume 34, pp. 20342–20354, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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*, pp. 1039–1048, 2017.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

Yong Guo, David Stutz, and Bernt Schiele. Improving robustness of vision transformers by reducing sensitivity to patch corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4108–4118, 2023.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. volume 28, 2015.

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.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. 2015.

Soheil Hor, Ying Qian, Mert Pilanci, and Amin Arbabian. Adaptive inference: Theoretical limits and unexplored opportunities. *arXiv preprint arXiv:2402.04359*, 2024.

Md Belal Hossain, Muhammad Kafiul Islam, and Sheikh Iqbal Ahamed. Computational complexity reduction techniques for deep learning. In *IEEE International Conference on Artificial Intelligence*, pp. 1–6. IEEE, 2023.

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.

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 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

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*, pp. 2704–2713, 2018.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Zhipeng Jiang, Tianqi Chen, Lianmin Zhang, Yida Wang, and Mu Li. Efficient deep learning inference on edge devices. *Proceedings of Machine Learning and Systems*, 1:1–13, 2018.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

Duhong Liu, Wei Yi, and Ruihao Li. A survey of model compression techniques. *Frontiers in Robotics and AI*, 12, 2025.

Sicong Liu and Yingyan Chen. Efficient neural networks for edge devices. *Computers and Electrical Engineering*, 94:107257, 2021.

Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International conference on learning representations*, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. In *International Conference on Learning Representations*, 2022a.

Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. 2022b.

Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.

Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas. Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. *Pattern Recognition*, 105:107346, 2020.

Sayak Paul and Pin-Yu Chen. Vision transformers are robust learners. *arXiv preprint arXiv:2105.07581*, 2021.

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. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8583–8595, 2021.

Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. Why should we add early exits to neural networks. *Cognitive Computation*, 12:954–966, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

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*, pp. 2464–2469. IEEE, 2016.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. *arXiv preprint arXiv:2204.07118*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pp. 409–424, 2018.

Jiachen Yu, Xin Wang, Tao Wang, Sheng Gu, and Dacheng Tao. Improving robustness of vision transformers by reducing sensitivity to patch corruptions. *IEEE Transactions on Multimedia*, 2024.

Chenchen Zhang, Yuqi Yang, Huayou Liu, Jun Wang, and Chengzhong Xu. Edgenn: Efficient neural network inference for cpu-gpu integrated edge devices. In *IEEE International Conference on Computer Design*, pp. 233–242. IEEE, 2023.

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Understanding the robustness in vision transformers. In *International Conference on Machine Learning*, pp. 27378–27394. PMLR, 2022.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577, 2024.