This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Multi-path Routing For Conditional Information Gain Trellis using Cross-Entropy Search and Reinforcement Learning

UFUK CAN BICICI[1], LALE AKARUN[2]
[1]Bogazici Universtiy, Bebek, 34342 Besiktas/Istanbul, Turkiye(e-mail: can.bicici@boun.edu.tr)
[2]Bogazici Universtiy, Bebek, 34342 Besiktas/Istanbul, Turkiye(e-mail: akarun@boun.edu.tr)
Corresponding author: Ufuk Can Bicici (e-mail: can.bicici@boun.edu.tr).

ABSTRACT Convolutional neural networks have made significant strides in solving computer-vision tasks at the expense of high computational demands. This complexity hinders efficient processing, particularly on devices with limited computational resources such as edge devices. One way to overcome this limitation is conditional computing, which optimizes inference by selectively utilizing parts of the network depending on the characteristics of the input. A recent conditional execution method is Conditional Information Gain Trellis (CIGT), which routes samples based on an information gain-based router mechanism. The original CIGT model was designed to route a single sample along a single path in a trellis structure. In this study, advanced inference strategies that allow inputs to traverse multiple paths are proposed to improve the performance of the vanilla CIGT model. These strategies aim to find a middle ground between improved model performance and increased computational demands. For this purpose, two techniques were proposed: A Cross-Entropy Search-based threshold optimization algorithm and a Reinforcement Learning-based routing strategy. The first method treats multi-path routing in CIGT as a black-box optimization problem and the second interprets it as a Markov Decision Process with a Q-Learning-based supervised regression algorithm designed as the solution. It has been shown that both of these methods provide significant performance improvements compared to the original CIGT model, with an adjustable increase in the computation. Experiments were conducted on two image datasets, with additional statistical tests and analyses to inspect the behaviors of the proposed algorithms. The novel methods designed in this study for multi-path routing show potential for both the original CIGT model and similar conditional computation approaches that use specific routing mechanisms for selecting network parts based on samples.

INDEX TERMS Conditional Execution, Deep Neural Networks, Convolutional Neural Networks

I. INTRODUCTION

Deep convolutional neural networks (CNNs) have significantly advanced image classification tasks owing to their large parameter count; however, their high computational requirements limit their deployment on edge devices. Conditional computing, introduced by Bengio [1], addresses this problem by activating only parts of the model based on an input sample. Approaches vary from sampling smaller structures within a larger network [2]–[5], utilizing supernets where subsets are selected for routing [6]–[8], incorporating pre-defined sub-structures or employing early exits for efficient processing [9], [10]. Additionally, Mixture of Experts (MoE) systems route samples to different ‘‘expert’’ networks, presenting a collaborative ensemble approach, as seen in [11]–[14]. These methodologies offer scalable and resource-efficient solutions for deploying complex neural networks to devices with limited computational capabilities.

Building on the principles of conditional computing, ‘‘Conditional Information Gain Networks’’ (CIGNs) have been developed to further refine sample routing through network models [15], [16]. The tree structure allows parts of the model to be bypassed. Within the CIGNs, non-leaf nodes serve as routers. They employed information gain to drive routing decisions and cluster semantically similar classes. This method facilitates the early grouping of similar classes at the router level. It assigns the task of separating closely related classes to the more detailed classifiers in the leaf nodes.

The tree structure of the CIGN model sometimes leads to misrouting and sending samples to sub-trees unsuitable for their classification. Addressing the limitations of the CIGN

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

model, particularly its propensity for misrouting, the ''Conditional Information Gain Trellis'' (CIGT) model was proposed to enable more reliable sample routing [17]. Unlike a tree, CIGT is structured as a trellis, allowing multiple pathways between nodes. This configuration offers misrouted samples the opportunity to realign with the appropriate computational units in the subsequent layers. Inspired by the information gain routing mechanism described in [15], CIGT was designed as a Directed Acyclic Graph (DAG). It aims to establish expert pathways for managing the diverse semantic groups in a dataset. Each layer of Trellis contains N parallel groups, each dedicated to capturing essential data representations.

Although conditional computing is based on the idea of grouping similar samples and routing them to a subset of the computational units, some samples are between groups and are difficult to classify. These difficult samples may benefit from being routed to multiple computational units, which may cooperate in their classification. This approach is known as multi-path routing.

This study introduces two novel multi-path routing techniques. The first technique involves setting the probability thresholds in each layer's routing unit. Samples with routing probabilities above these thresholds follow multiple paths. This leverages the specialized capabilities of various root-to-leaf networks as a mixture of experts. The classification outputs from these paths are averaged to increase the decision accuracy. The problem of finding optimal routing thresholds is addressed using the Cross-Entropy Search algorithm [18]. A carefully designed loss function balances the accuracy with computational efficiency.

Second, a dynamic, sample-specific multiple-path routing mechanism is proposed and formulated as a Reinforcement Learning (RL) problem within a Markov Decision Process (MDP) framework. This method enables routing decisions that are dependent on the sample, aiming for optimal accuracy with minimal computational overheads. For this purpose, a supervised Q-learning approach tailored to the finite, manageable scale of the CIGT network structure has been developed. Both these methods seek to improve the performance of the single-path CIGT models introduced in [17] while trying to keep the extra computation generated by multi-path routing as small as possible. The improved results of these designs are demonstrated through experiments on the Fashion-MNIST (Modified National Institute of Standards and Technology) and CIFAR-10 (Canadian Institute for Advanced Research) datasets.

The remainder of this paper is structured as follows. Section 2 presents a review of the literature on conditional execution in deep network architectures. Section 3 summarizes the CIGT model. In Section 4, the main contributions of this paper, namely the Cross-Entropy Search and supervised Q-learning-based multi-path inference algorithms for CIGT models, are explained. Section 5 presents experiments and discussions and Section 6 concludes the paper.

II. RELATED WORK

A. CONDITIONAL COMPUTING

Conditional computation, as introduced by Bengio [1], [19] aims to reduce the training and evaluation costs by selectively activating parts of the network. The goal of selective activation is to maintain an efficacy similar to that of standard methods. ''Blockout'' by Murdock et al. [2] and Li et al.'s probabilistic element disabling method [20] in multi-modal settings improved the approach. Bengio et al.'s binary mask technique [21] uses the REINFORCE algorithm [22] and refined conditional execution. Wu et al.'s ''Blockdrop'' [3] and Veit's adaptive inference graphs [4], employing Gumbel-Softmax [23], [24] for discrete decisions, enable complete skipping of layers. ''SkipNet'' by Wang et al. [25] combined supervised and reinforcement learning for gating functions. Herrmann et al. [5] focused on channel selection with Gumbel-Softmax, which differs from the layer-level methods in [4], [25]. Krishna et al. [26] extended channel selection techniques to self-supervised learning. Wojcik et al. [10] applied a routing system to visual transformers. They trained routers separately using output distance-based heuristics to meet specific thresholds.

Yu et al. [27] proposed ''Slimmable Neural Networks.'' This technique involves embedding simpler neural networks into a more heavy-weight ''Supernet.'' The system dynamically selects an appropriately sized smaller network based on the complexity of the input, often by masking or indexing out unnecessary channels [28]. Further developments such as the ''Sandwich Rule'' and ''In-Place Distillation'' by Yu et al. [29] have improved the training efficacy.

The techniques discussed thus far involve dynamically adjusting neural network architectures per input by selectively deactivating certain components. A contrasting strategy defines a fixed hierarchical architecture, and uses routing mechanisms to activate specific sections. Ioannou et al. [6] introduced small routing networks within a DAG to allocate weights to different computational pathways, which were then combined via weighted averaging. During inference, only the pathways with the highest weights were activated. Xiong et al. [30] adopted a distinct route by implementing a binary tree-structured network where routing decisions are made based on split losses at each node, with the aim of separating data along a hyperplane without direct supervision signals. Liu et al. [7] also explored routing mechanisms, directing samples through various subnetworks, with the difference of employing Q-learning for both network and routing training, aligning with Ioannou et al.'s method, but with a different learning strategy. Tanno et al.'s ''Adaptive Neural Trees'' use probabilistic routing and grow a neural tree with validation-set feedback [31].

''Branchynet'' by Teerapittayanon et al. [9] introduced the idea of early exits based on entropy, which was further developed by Han et al. [32] through a weight adjustment method for balancing early and late exits. Pacheco et al. [33] and Chiang et al. [34] proposed algorithms for optimizing network partitioning and early exit placement, respectively,

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

for cloud-edge computing environments. Ayyat et al. [35] and Jeon et al. [36] focused on enhancing the early exits for critical class detection and optimizing classifier locations. Yang et al. [37] explored adjusting the image resolution for early exit. They presented a strategy for scaling the resolution based on the sample complexity.

Most conditional computation techniques aim to retain the model performance as high as possible, at least at a level comparable to that of the original full model. A common approach for increasing model performance in deep learning is data augmentation, either via various transformations of the input data, such as Cutout [38], RandAugmnent [39], and Cutmix [40]. Another approach for data augmentation is through synthetic data generation using generative models [41]–[43]. In addition to these, some conditional computation approaches strive to improve the model performance with special routing techniques such as using information gain [15] or knowledge distillation from a larger model [44], [45]. Adding multi-path routing approaches to the CIGT technique can be considered in the latter group.

TABLE 1. Comparison between closely related works in conditional deep learning.

Algorithm	Advantages	Disadvantages
Adaptive Inference Graphs [4]	Layer-wise skipping.	Large Memory footprint. Low computation efficiency.
Channel Selection with Gumbel Softmax [5]	Channel-wise skipping. High model performance.	Large Memory footprint. Low computation efficiency.
Adaptive Neural Trees [31]	Adaptive design of neural tree.	Tree structure: Open to misrouting.
Self-born Wiring for Neural Trees [57]	Automatic design of neural tree. Multi-path inference.	Tree structure: Open to misrouting.
Conditional Information Gain Networks [15], [16]	Information gain based routing. Multi-path inference.	Tree structure: Open to misrouting.

B. MIXTURE OF EXPERTS

In the Hierarchical Mixture of Experts (HMoE) framework [11], gating mechanisms allocate probabilistic weights to experts within their mix, with a Bayesian perspective available in [46]. However, deploying these methods in deep neural networks is challenging, making them rare [47]. As a result, adaptations of the HMoE model suitable for deep learning have only recently emerged.

Eigen et al. integrated the experts model into deep networks by layering expert and gater networks [14]. Shazeer et al. developed a model using sparse gating. This allows for manageable selection and training of experts. They focused on the top-k experts for efficient training and load-balancing [12], while Lee et al. explored advanced ensemble methods for convolutional networks. They showed that shared parameters and ensemble-aware losses yield better results than merely averaging the outputs of independent models [48]. HD-CNN, introduced by Yan et al. [8], cluster large class groups using spectral clustering to form a tree-structured network. Ahmed et al. [49] suggested a similar setup with networks that initially categorized data into broad groups, followed by specialist networks for each group, and later fine-tuned the combined experts, albeit without conditional execution. Chazan et al. applied a deep mixture of experts for speech enhancement. They trained their model using a revised Expectation-Maximization method [50]. Furlanello et al.'s ''Born-Again Neural Networks'' [51] iteratively created ensembles of networks through knowledge distillation, enhancing the multi-path network architecture without using conditional computing. Kirsch et al. [52] approach deep learning with a Mixture of Experts (MoE) model, applying a Viterbi-EM training method and addressing ''mode collapse,'' where too many samples favor a single expert. Efforts to understand MoE systems theoretically have also progressed. Baykal et al. [13] explored sparse activation in neural networks using locality-sensitive hashing, demonstrating the learning capabilities of such networks in a limited setting of Lipschitz functions. Dikkala et al. [53] analyzed the MoE behavior on synthetic data, highlighting the advantages of learned routing over preset functions and the ability of routers to focus on the informative data dimensions for better performance. Transformer models, recognized for handling complex data [54], [55], are also being optimized using MoE approaches for more efficient inference. ''Switch transformers'' by Fedus et al. [56] integrate MoE and trainable routers into Transformers, emphasizing computation efficiency.

C. APPLICATION AREAS FOR CONDITIONAL DEEP LEARNING

Deep learning applications have gained increasing attention in recent years [58]. These models have their power in their redundant representation learning capabilities. However, deploying such models in edge devices or in scenarios where rapid inference is necessary is difficult. Conditional computing, hence CIGT and its multi-path routing improvements, are potentially valuable approaches in such application fields. These fields include smart farming and precision agriculture [43], [59]–[63]. In smart farming, deep learning models are usually placed on drones or robotic devices, or deployed into chips mounted onto tractors and their implements to process data from various sensors. Models need to run in highly constrained environments in such cases; hence, along with other efficient deep learning techniques, conditional computation can have an important potential impact. Another important application area is autonomous driving, where efficient deep neural network models have paramount importance in extracting meaningful data from automobile sensors for safe driving or anomaly detection in traffic [64]–[66]. Finally, Internet of Things (IoT) applications constitute an impactful

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

avenue for efficient deep learning techniques. Algorithms must run on low-resource devices in this domain, such as embedded systems, and execute computationally expensive operations, such as like image classification, object detection, and sensor fusion [67], [68]. Conditional execution of deep learning models, hence CIGT and multi-path routing techniques on this particular model, can be of high value for such applications.

Table 1 summarizes the most closely related conditional computation methods from the literature to the multi-path routed CIGT models, with their perceived advantages and disadvantages.

III. CONDITIONAL INFORMATION GAIN TRELLIS

In this section, the CIGT model is briefly recaptured. An example of a CIGT architecture is shown in Figure 1. It uses information gain-based losses for conditional execution across layers. The aim was to establish expert pathways for different data semantics. Within this structure, the computational units are organized into parallel groups or F units. These units were tasked with learning specific data representations. In addition, there is a singular routing unit H_l, per layer. The router calculates the probability distribution over the F units for the next layer. It effectively creates a hierarchical mixture of experts across the trellis. This approach allows for dynamic adaptation and specialized processing paths for various data inputs. CIGTs employ routing systems that focus on differentiable information gain, facilitating sparse activation, and semantic grouping within the trellis structure. This method enhances the ability of the model to discern fine details among semantically similar samples. This simplifies the classification.

A. CIGT ARCHITECTURE

The CIGT model uses a Trellis structure, where each layer i contains K_i identical parallel units. These units are adaptable to any standard neural network component such as convolutional or dense layers. A CIGT layer, denoted as F_i, comprises units $F_{i,j}$ (with $0 \leq j \leq K_i - 1$). Each unit has fewer parameters to capture the shared low-level features. Initially, an unconditional computation block F_0 processes the inputs. The CIGT architecture in Figure 1 features a three-layered structure named CIGT-[1,2,4] with 1, 2, and 4 computation units per layer, respectively.

For a given layer with $K_i = 2$, the operations are divided into two parallel units. This halves the dimensions for operations such as convolution and dense layers, leading to units named $F_{i,0}$ and $F_{i,1}$. Routing decisions direct the sample through these units, forming a sequential root-to-leaf path $(F_{L,i_L} \circ \cdots \circ F_0)(x)$. This represents a single root-to-leaf expert path that establishes a posterior distribution $p(y|Z,x,\theta)$. Here, Z is a matrix that indicates the chosen experts. The lth row represents selections in layer l, with layer 0 always being single and shared. In the CIGT model, $\mathbf{Z_l}$ is a binary vector. It shows which unit is selected at each layer, with zeros for non-existent units. This indicates the chosen computational path for the sample. θ represents all the neural network parameters. It includes weights for convolutional and dense layers, that influence F unit operations. Routers, or H units, use outputs from F units to create probability distributions $p(\mathbf{Z_{l+1}}|x,\theta,\phi)$ for choosing subsequent F units. This connection creates a dependence on θ. ϕ contains all the parameters for converting H unit inputs into routing distributions. Training H units focuses on maximizing information gain, producing a joint probability distribution for all potential expert paths within the model:

$$p(Z|x,\theta,\phi) = \prod_{l=1}^{L} p(\mathbf{Z_l}|x,\theta,\phi)$$

The main loss function of the CIGT architecture, J_C is the lower bound of the Hierarchical Mixture of Experts model, as stated below:

$$\begin{aligned} J_C(\theta,\phi) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{Z}\log p(y_i|Z,x_i,\theta)\prod_{l=1}^{L}p(\mathbf{Z_l}|x_i,\theta,\phi) \\ &\leq \hat{J}_C(\theta,\phi) := \frac{1}{N}\sum_{i=1}^{n}\log\sum_{Z}p(y_i|Z,x_i,\theta)\prod_{l=1}^{L}p(\mathbf{Z_l}|x_i,\theta,\phi) \end{aligned} \tag{1}$$

which is the result of Jensen's inequality [17]. To train the routing units, information gain losses IG_l for every layer l are also added, resulting in the following overall loss:

$$L_{CIGT} = -J_C(\theta,\phi) - \lambda_{IG}\sum_{l=1}^{L} IG_l + \Omega(\theta,\phi)$$

Here, λ_{IG} controls the intensity of the collective information gain losses and Ω serves as a regularizer of the network parameters.

B. LOCAL INFORMATION GAIN LOSSES

The CIGT model uses router units (H) to direct input across different layers through computational units (F). Each layer has a set number of parallel units, that are selected via probability distributions $p(\mathbf{Z_l}|x,\theta,\phi)$. These distributions, determined by the network (θ) and router (ϕ) parameters, are optimized using information gain losses. This approach is in line with CIGN models [15], [16]. At each layer of the CIGT, the following joint probability distribution is defined:

$$p(y,\mathbf{Z_l},x|\theta,\phi) = p(y,x)p(\mathbf{Z_l}|x,\theta,\phi)$$

Here, $p(y,x)$ represents the empirical distribution of training data. $p(\mathbf{Z_l}|x,\theta,\phi)$ is expressed as a softmax function applied to the potential values of the one-hot vector $\mathbf{Z_l}$:

$$p(\mathbf{Z_l} = \text{onehot}(k)|x,\theta,\phi) = \frac{\exp\{(\mathbf{w_k}^T\mathbf{h_{l-1}}(x) + b_k)/\tau\}}{\sum_{i=1}^{K_l}\exp\{(\mathbf{w_i}^T\mathbf{h_{l-1}}(x) + b_i)/\tau\}}$$

Router decisions are based on the hyperplanes $(\mathbf{w_k}, b_k)$ creating decision boundaries, with τ adjusting the smoothness of the probability distribution.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

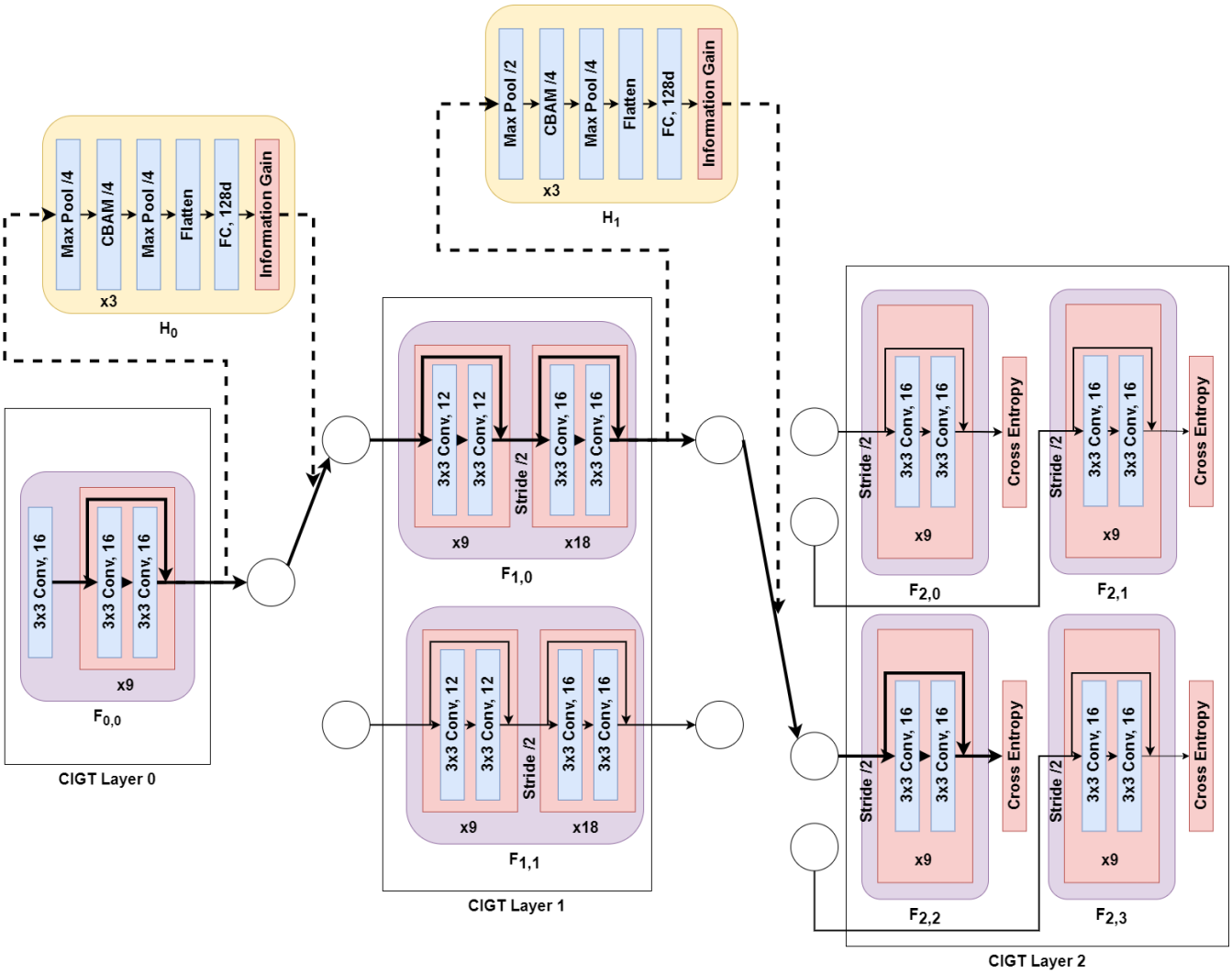

FIGURE 1. **An example of the CIGT architecture. In the terminology of [17], this corresponds to a CIGT-[1,2,4] structure. The router units H_i direct the samples into the computation units of the following layers, namely $F_{i+1,j}$. The image represents the CIGT-[1,2,4] ResNet110 network that has been used in CIFAR-10 experiments.**

The local information gain at layer l is then given as:

$$IG_l = \mathbb{H}\left[p(y)\right] - \mathbb{E}_{p(\mathbf{Z_1}|\theta,\phi)}\left[\mathbb{H}\left[p(y|\mathbf{Z_1},\theta,\phi)\right]\right]$$

Aiming for balanced data distribution among F units, CIGT models use the following information gain definition using the mathematical decomposition of the information gain:

$$IG_l^{balanced} = \mathbb{H}\left[p(y)\right] + \lambda_{balance}\mathbb{H}\left[p(\mathbf{Z_1}|\theta,\phi)\right] - \mathbb{H}\left[p(y,\mathbf{Z_1}|\theta,\phi)\right]$$

The decomposed $IG_l^{balanced}$ term incorporates a balance factor ($\lambda_{balance}$) to ensure even data allocation across computational nodes. The decomposed form addresses "mode collapse" issues more naturally than other approaches [12], [21], [69].

C. TRAINING CIGT MODELS

The derivatives of the lower bound in Eq. (1) with respect to θ and ϕ, $\frac{\partial J_C}{\partial \theta}$ and $\frac{\partial J_C}{\partial \phi}$ were derived in [17].

Approximating the first part of $\frac{\partial L_C}{\partial \theta}$ can be done using Monte Carlo methods by sampling x_i from $p(x)$ and Z_i from $p(Z|x_i,\theta,\phi)$. However, the second part and $\frac{\partial L_C}{\partial \phi}$ calculations are not straightforward. This is because they do not rely on expectations of a known distribution, making Monte Carlo simulations inapplicable.

In the CIGT framework, the terms $\frac{\partial J_C}{\partial \phi}$ and the second part of $\frac{\partial J_C}{\partial \theta}$ are ignored entirely. The reduced gradient is denoted as $\frac{\partial \hat{J}_C}{\partial \theta}$. The parameters of the H units are exclusively updated based on the information gain losses at each block,

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

given by $-\lambda_{IG}\sum_{l=1}^{L}\frac{\partial}{\partial\phi}IG_l$. Similarly, the parameters of the computation units (F) are updated using the information gain objectives. In each CIGT layer l, the router units utilize the outputs of the computational units. They construct and sample from the probability distributions $p(\mathbf{Z_{1+1}}|\hat{x}_l,\theta,\phi)$ where $\hat{x}_l$ represents the output of each corresponding F unit for a particular sample in the current CIGT layer. Based on the one-hot vector sample $\mathbf{Z_{l+1}}$, the corresponding F units in the next block are activated. This process is equivalent to sampling a root-to-leaf path for each sample. The sample is evaluated sub-sequentially, and the corresponding error signal is back-propagated along the traversed path, as in a conventional deep neural network. Given that the gradients of the main loss are partially utilized and only information gain losses train the router nodes, this method is termed "Approximate Training" in [17]. Additionally, [17] allows routers to undergo uniform sampling for a certain number of epochs before introducing information-gain objectives. This initial phase ensures that each route is trained with every possible sample. Such pretraining helps root-to-leaf experts acquire insight into the entire dataset before focusing on their coarse class groups.

The CIGT architecture does not make assumptions regarding the format of the input data or network type. Although it has been used with image data, it can also be used with other modalities. The CIGT architecture makes mild assumptions regarding how the categories in the data are structured. Owing to its semantic grouping ability of classes, CIGT best fits situations where the data has enough classes and these classes have some sort of explicit or implicit semantic grouping among themselves.

IV. MULTI-PATH ROUTING TECHNIQUES FOR CONDITIONAL INFORMATION GAIN TRELLIS

The trellis model used in CIGT groups semantically similar samples and routes them to a path that specializes in characterizing them. However, there may still be samples that consistently get misrouted at every CIGT layer owing to uncertain decisions. Routing such samples to multiple paths can help recover such cases due to the ensemble effect. Introducing a Sparse Mixture of Experts (sMoE) inference mode allows the invocation of more than one root-to-leaf path. In this study, two methods are introduced for multi-path routing.

A. MULTI-PATH ROUTING WITH CROSS-ENTROPY SEARCH

1) Cross-Entropy Search

Cross-Entropy Search is a derivative-free optimization algorithm that simplifies the process of determining the optimal value of a function. It only requires that the function be easily evaluated within its domain [18]. This involves optimizing a score function $S(\mathbf{x})$ over a search space $\mathbf{x}\in X$, aiming to find the optimal score $\gamma^* = S(\mathbf{x}^*)$. It utilizes a family of probability density functions $f(\mathbf{x},\mathbf{w})$, where $\mathbf{w}\in W$ is a vector of adjustable parameters. These functions are used to sample points within search space X. The basic insight of the Cross-Entropy Search method is that when a set of sample points $X_1, X_2, \ldots, X_N$ is sampled from an initial density function $f(\mathbf{x},\mathbf{u})$, the number of acceptable samples that is close enough to the optimum value γ^* is low. To efficiently learn the optimal parameters $\hat{\mathbf{w}}^*$ despite the rarity of events, where $s(\mathbf{x}) > \gamma_t$, a two-step iterative strategy is employed [18]. Initially, N samples from $f(\mathbf{x},\mathbf{u})$ are evaluated and ranked. The top $(1-\rho)$ percentile identified by γ_1 is selected for further optimization. This selection leads to a preliminary set of parameters $\mathbf{w_1}$ via the maximum-likelihood optimization. Utilizing $f(\mathbf{x},\mathbf{w_1})$ to generate new samples, the process of iteratively refining the parameter set continues. This cycle continues until either a set number of iterations are completed or the improvement in the $(1-\rho)$ percentile scores stabilizes. Stabilization indicates the approach to a solution.

2) Cross-Entropy Search for CIGT Threshold Determination

The Cross-Entropy Search method is used to identify the optimal threshold values for each connection between the parent and child nodes in the CIGT. For instance, in a CIGT-[1,2,4] setup, 10 unique threshold parameters are necessary.

In the CIGT model, the Cross-Entropy Search space is defined within the $[0,1]$ interval, which is appropriate for the probability thresholds. The search space's dimension matches the number of thresholds used. Specifically, for a layer with K_l computation units, thresholds for connections to these units are searched within the $[0,1/K_l]$ range. A critical step in the Cross-Entropy method involves selecting a suitable density function $f(\mathbf{x},\mathbf{w})$, for sampling these CIGT routing probability thresholds. The probability density function family should be continuous and multi-modal.

As $f(\mathbf{x},\mathbf{w})$ density families, Gaussian Mixture Models (GMMs) are preferred in this study. However, GMMs extend across the entire real number range, which is not sensible for probability thresholds. This is addressed by transforming the GMM samples using a sigmoid function to confine them within a finite interval. Namely, vector $\mathbf{x}$ is sampled from a GMM as:

$$\mathbf{x}\sim\sum_{n=1}^{K}\pi_n N(\mathbf{x}|\mu_n,\Sigma_n) \tag{2}$$

Assuming that the search space is D-dimensional, for every dimension of $\mathbf{x}$, the following sigmoid-based transformation is defined, which takes $\mathbf{x}$ into a pre-defined finite interval:

$$y_i = u_i(x_i) = (b_i - a_i)\frac{1}{1+\exp\{-x_i\}} + a_i \tag{3}$$

For each dimension, the interval endpoints (a_i, b_i) were used as bounds for the probability thresholds. Then, a vector-valued function is applied to the GMM vector $\mathbf{x}$. This function transforms $\mathbf{x}$´s real-line elements into these specific intervals as follows:

$$\mathbf{y} = u(\mathbf{x}) = (u_1(x_1), u_2(x_2), \ldots, u_D(x_D)) \tag{4}$$

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

Here, $\mathbf{y}$ represents the threshold set for the Sparse Mixture of Experts routing mechanism. To integrate $\mathbf{y}$ into the Cross-Entropy Search, its probability density function (PDF) must be determined. Given a random variable $\mathbf{x}$ with a known PDF $f_X(\mathbf{x})$ undergoing a reversible transformation $\mathbf{y} = u(\mathbf{x})$ (with $v(u(\mathbf{x})) = \mathbf{x}$ for reversibility), the PDF of $\mathbf{y}$ is derived as:

$$f_Y(\mathbf{y}) = |J| f_X(v_1(\mathbf{y}), v_2(\mathbf{y}), \ldots, v_D(\mathbf{y})) \tag{5}$$

$|J|$ is the determinant of the Jacobian matrix, with entries $J_{i,j} = \dfrac{\partial v_i(y_1, y_2, \ldots, y_D)}{\partial y_j}$. In the current case, the inverse functions v_i and their derivatives with respect to y_i are defined as:

$$\begin{aligned} v_i(y_i) &= \log\left(\frac{y_i - a_i}{b_i - y_i}\right) \\ \frac{dv_i}{dy_i}(y_i) &= \frac{a_i - b_i}{(y_i - a_i)(y_i - b_i)} \end{aligned} \tag{6}$$

Finally, observing that the determinant of the diagonal Jacobian matrix J is equal to the product of its diagonal elements, the full expression for the PDF of the $\mathbf{y}$ variables can be written as:

$$\begin{aligned} f_Y(\mathbf{y}) = &\left(\prod_{i=1}^{D} \frac{a_i - b_i}{(y_i - a_i)(y_i - b_i)}\right) \times \\ &\left(\sum_{k=1}^{K} \pi_k N\left(\log(q_1), \log(q_2), \ldots, \log(q_D) \,|\, \mu_k, \Sigma_k\right)\right) \end{aligned} \tag{7}$$

where q_i is defined as $q_i = \dfrac{y_i - a_i}{b_i - y_i}$, $1 \le i \le D$.

The logarithm of the probability density function $f_Y(\mathbf{y})$ leads to simple maximization during Cross-Entropy Search iterations. The Jacobian term does not depend on the GMM parameters; therefore, it can be omitted. The remaining logarithm term with the actual GMM PDF can be optimized efficiently using the EM algorithm by converting the $\mathbf{y}$ variables back into the original $\mathbf{x}$ variables.

Before performing the Cross-Entropy optimization to find the best probability threshold values, a preprocessing step is executed. For simplicity, a CIGT-[1,2,4] structure is considered. In this structure, the network produces three sets of routing probabilities for a dataset of N samples. These include an $N \times 2$ matrix $A_{0,0}$ for the initial routing from nodes $F_{0,0}$ to $F_{1,0}$ and $F_{1,1}$. There is also an $N \times 4$ matrix $A_{1,0}$ for routes from $F_{1,0}$ to four nodes at the next level. Similarly, an $N \times 4$ matrix $A_{1,1}$ exists for the routing from $F_{1,1}$. These $A_{i,j}$ matrices, which potentially contain raw activations, are converted to routing probabilities using a row-wise softmax operation: $\hat{A}_{i,j} = \mathrm{softmax}(A_{i,:})_j$. It is noted that routing probabilities often lean towards certainty, with one path's probability nearing 1 and the others approaching 0. Such an extremity in probabilities poses challenges for Cross-Entropy optimization. This is because the search space is narrowly concentrated around very small or large values. A more even distribution of routing probabilities, without altering the routing decisions, would make the Cross-Entropy optimization more manageable. To address this, the routing probability entropy for each decision node n and sample i is calculated as follows:

$$E_{n,i}(T) = -\sum_{k=1}^{K} \left(\frac{a_{i,k}^n / T}{\sum_{j=1}^{K} a_{i,j}^n / T}\right) \log\left(\frac{a_{i,k}^n / T}{\sum_{j=1}^{K} a_{i,j}^n / T}\right) \tag{8}$$

In this equation, $a_{i,k}^n \in A_n$ represents the activation of the kth child node for sample i in the upcoming layer of the CIGT model. These activations are processed by softmax layers within the CIGT framework to determine the actual routing probabilities. The temperature parameter T is crucial. T specifically facilitates the Cross-Entropy method for fine-tuning probability thresholds. It is thus a post-training adjustment.

Subsequently, the algorithm computes the variance of $E_{n,i}(T)$ for every sample across the dataset concerning node n:

$$Var_n(T) = \frac{1}{N} \sum_{i=1}^{N} (E_{n,i}(T) - \mathbb{E}[E_{n,i}(T)])^2 \tag{9}$$

By adjusting temperature T to maximize the variance of routing entropies, $T^* = \arg\max_T Var_n(T)$, less deterministic and more uniformly distributed routing probabilities are achieved. This adjustment facilitates the subsequent Cross-Entropy optimization. Initially, routing decisions across the dataset were highly certain. However, after optimizing for T, the spread of entropies increases. This results in more evenly distributed routing probabilities. These probabilities avoid the extremes of 0 and 1.

For the score function, a balancing term is used between the extra accuracy to be gained with multi-path routing and the extra calculation, measured in Multiplication and Accumulation (MAC) units:

$$L_{opt}(\mathbf{t}) = \lambda L_{class}(\mathbf{t}) + (\lambda - 1)\mathbb{E}_{p(x)}[L_{MAC}(x, \mathbf{t})] \tag{10}$$

$\lambda \in [0, 1]$ determines the balance between the accuracy and extra computational burden. Under this setting with the Cross-Entropy Search learned multi-path routing, the enhanced CIGT algorithm is called s-CIGT ((s)parse Mixture of Experts CIGT).

B. MULTI-PATH ROUTING WITH REINFORCEMENT LEARNING

In Cross-Entropy Search-based multi-path routing, thresholds are considered global, meaning that the same set of optimal thresholds is applied to every sample. The second algorithm explores a new idea, inspired by Q-learning from a Markov Decision Process (MDP), to enable per-sample thresholds.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

Algorithm 1: The Cross-Entropy Search Algorithm for s-CIGT

Decision Nodes of CIGT : F_n
Validation Dataset : D
Number of total iterations : T
Number of iterations without improvement: M
Initial parameters : $\mathbf{w_1}$
Quantile : ρ
Score function : $s(.)$
Probability Density Function : $f(\mathbf{y}|\mathbf{w_1})$, Sigmoid GMM as in Equation 7

Result: Optimum routing thresholds $\mathbf{t}_o$
Calculate $T_n = \arg\max_v Var_n(v)$ for every F_n;
Apply T_n to every routing probability before start;
while $i < T$ **do**
  Sample $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N \sim f(\mathbf{y}|\mathbf{w_i})$;
  $S_i = []$;
  for $\mathbf{t}_j \in \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_N\}$ **do**
    Evaluate the sampled threshold: $s(\mathbf{t}_j) = \mathbb{E}_{(x,y)\sim D}[\lambda L_{classification}(x, y, \mathbf{t}_j) + (\lambda - 1)L_{MAC}(x, \mathbf{t}_j)]$;
    $S_i.add(s(\mathbf{t}_j))$
  end
  Sort samples according to the scores: $s(\mathbf{t}_1), s(\mathbf{t}_2), \ldots, s(\mathbf{t}_N)$;
  Determine (1-ρ) quantile performance γ_i;
  Solve the optimization problem: $\mathbf{w}_{i+1} = \arg\max_{\mathbf{v}} \frac{1}{N}\sum_{j=1}^{N} \mathbb{I}[s(\mathbf{t}_j) > \gamma_i] \log f(\mathbf{t}_j|\mathbf{v})$;
  Break if γ_i has not been improved in the last M iterations;
end

Theoretically, this approach is expected to yield better results. In the following, the interpretation of CIGT routing as an MDP and the derived Q-learning algorithm is explained.

1) Interpreting CIGT routing as a Markov Decision Process

CIGT routing is interpreted as an MDP by considering all the possible routes that have been taken in the CIGT structure so far. In general terms, the relevant steps are listed as follows:

- At t^{th} level of the CIGT, it is assumed that starting from the root node, all paths that have been executed are stored in a set F: $F = \{F_{t-1,i_t^1} \circ \cdots \circ F_{1,i_1^1} \circ F_{0,0}(x), \ldots, F_{t-1,i_t^K} \circ \cdots \circ F_{1,i_1^K} \circ F_{0,0}(x)\}$. This indicates that there have been K such paths from the root node have been executed, and F_{l,i_a^b} means b^{th} path executes $(i_a^b)^{th}$ computational unit at $(l+1)^{th}$ CIGT layer.
- The result of each execution path, $\mathbf{s}_r = r(x)$, $r \in F$, is stored in a set S. These are accumulated into a single state vector $\mathbf{s}_t = \frac{1}{|F|}\sum_{r\in F} \mathbf{s}_r$. This constitutes the state-space vector for t^{th} level.
- A policy network for level t samples an action $a_t \sim P_t(\mathbf{s}_t)$. The action space is equal to the number of computation units in the next CIGT layer: $a_t \in \{0, 1, \ldots, D_{t+1} - 1\}$.
- The router at CIGT level t is executed for each $\mathbf{s}_r \in F$: $H_t(\mathbf{s}_r)$. Selecting action a_t would lead to the execution of $a_t + 1$ units with the highest routing probabilities at the next level, for every executed path in F.
- The prediction $\hat{y} = \arg\max_y \left(\frac{1}{|F|}\sum_{r\in F} p(y|x, l = r)\right)$ is calculated in the final CIGT layer. The final reward term will be $R_L = \lambda I[\hat{y} = y] + (\lambda - 1)L_{MAC}(x, F)$ where $I[.]$ is the indicator function. In the inner layers of the CIGT, 0 is used as the reward. λ provides a balance between the extra accuracy and computation.

The entire process is depicted for a hypothetical CIGT-[1,2,4] architecture in Figure 2. It should be noted that this procedure is tractable and feasible owing to the finite nature of the CIGT models. In [17], at most 3 layered CIGTs were used, with 1, 2, and 4 computation units at every layer, respectively. However, for deeper CIGT architectures, it is possible to modify the MDP into a structure similar to a beam search. This modification aims to limit the number of executed paths $|F|$ at each layer and avoid exponential growth. The procedure is described in Algorithm 2.

Algorithm 2: The Markov Decision Process for CIGT Multi-path Routing

Sample : (x,y)
Computation Units : $F_{0,0}, F_{1,0}, \ldots$
Routing Units : $H_0, \ldots H_{L-1}$
Policy Networks : $P_0, \ldots P_{L-1}$
MDP Hyperparameters : λ
Result: Multi-path Routing Reward: R_L
Keep the record of executed routes in set F, initialize F with the root unit $F_{0,0}$: $F = \{F_{0,0}\}$;
Set L to the number of non-leaf CIGT levels;
for *t in [0:L]* **do**
  $F_{next} = \{\}$;
  $S = \{\}$;
  Keep routing probabilities to the next CIGT level in A, for every current route $r \in F$: $A = []$;
  for *r in F* **do**
    Execute the route and store its result: $r(x) \rightarrow \mathbf{s}_r$;
    $S = S \cup \mathbf{s}_r$;
    Execute the router unit, and sort the next level nodes based on routing probabilities: $A[r] = sort(H_t(\mathbf{s}_r))$;
  end
  Set the state vector, with an accumulation operation: $\mathbf{s}_t = accum(S)$;
  Sample action a_t from the policy network: $a_t \sim P_t(\mathbf{s}_t)$;
  for *r in F* **do**
    Select next-level nodes, with Python-style indexing: $F_{next} = F_{next} \cup \{n \in A[r][: (a_t + 1)]\}$;
  end
  $F_{next} \rightarrow F$;
end
Calculate the final reward: $\hat{y} = \arg\max_y \left(\frac{1}{|F|}\sum_{r\in F} p(y|x, l = r)\right)$;
$R_L = \lambda I[\hat{y} = y] + (\lambda - 1)L_{MAC}(x, F)$;

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

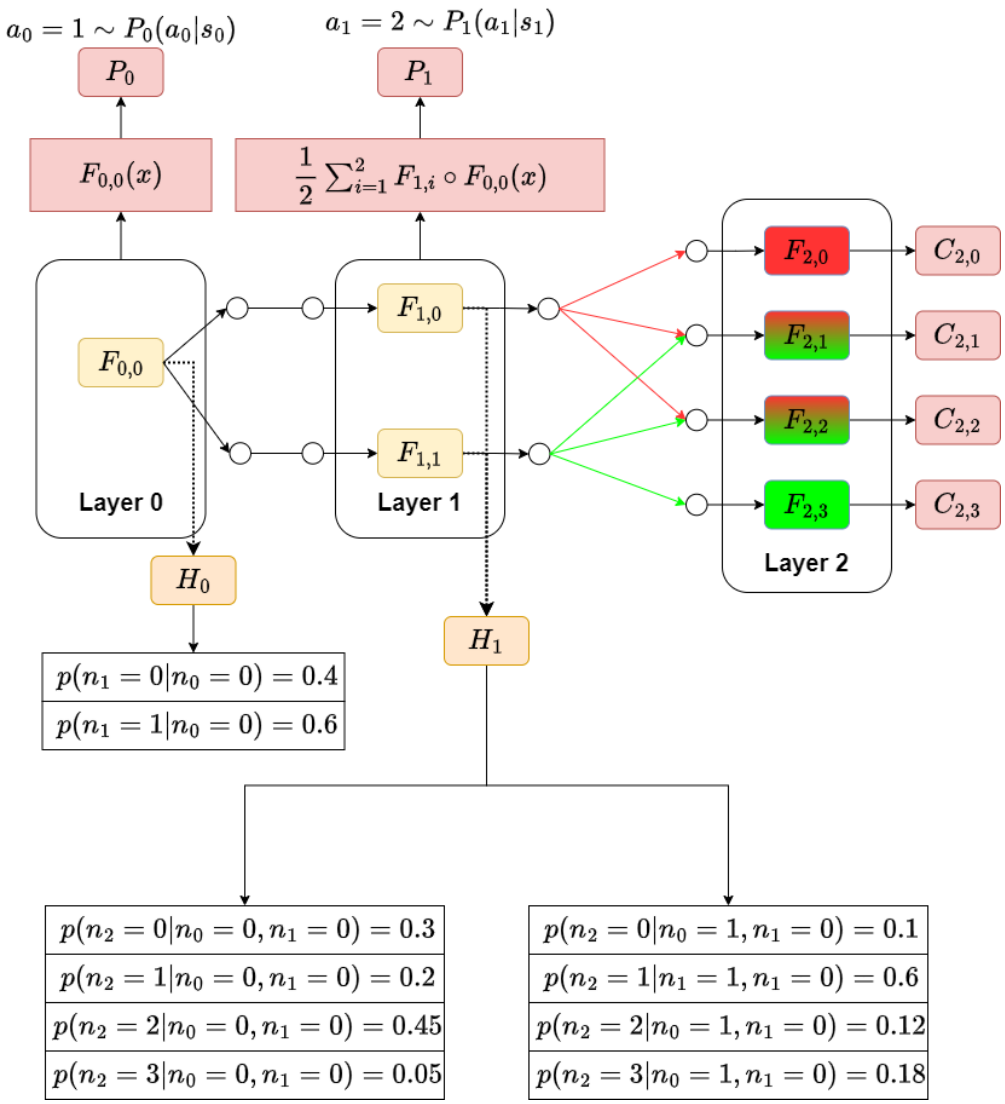

FIGURE 2. **The Markov Decision Process in a hypothetical CIGT-[1,2,4] structure. The action trajectory $a^0 = 1$ and $a^1 = 2$ is sampled. In Layer 2, 6 units are executed. The units with gradient color are executed by both units in the previous CIGT layer.**

2) Supervised Q-Networks Based Routing

The CIGT framework allows for modeling a finite-horizon MDP, where the episode lengths match the depth of the CIGT architecture. Typically, CIGT networks, in theory, might suggest an exponential growth in potential paths. However, in practice, one often works with a maximum depth of three [17]. Consequently, the path possibilities at each level of the CIGT-[1,2,4] model were limited to $1, 2, 8$. This limitation enables precise calculation and listing of reward tables. It sets up a Q-learning environment in which the optimal Q-tables are accurately determined at each step. This simplifies the problem into a regression task. This formulation contrasts with the more complex gaming scenarios. Such scenarios require dynamic experience-replay tables for deep Q-networks, as mentioned in [70]. There, the regression targets continuously shift; however, in the described algorithm, they remain fixed during training.

Using CIGT-[1,2,4] as an example, without loss of generality, it is started by solving from the final layer. The initial regression problem is then addressed, with the progression to earlier layers occurring sequentially. This approach is fitted to any CIGT configuration, regardless of depth or breadth. However, challenges might be faced by longer CIGT structures. This is because of the exponential increase in the size of the Q-tables.

For a sample x, let $\hat{y}(x, a_0, a_1)$ represent the predicted outcome after choosing actions a_0 and a_1, following the paths determined by these choices in our CIGT-based MDP model.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

The optimal Q-table is defined as follows:

$$Q_2(CIGT_2(x,a_0),a_1) = \lambda I[\hat{y}(x,a_0,a_1)=y] + (\lambda-1)L_{MAC}(CIGT_2(x,a_0),a_1)$$

Here, $CIGT_2(x,a_0)$ denotes the state of x after traversing all routes indicated by action a_0, consolidating into the state space for the CIGT level. The term $I[\hat{y}(x,a_0,a_1)=y]$ represents the reward for correct predictions, and $L_{MAC}(CIGT_2(x,a_0),a_1)$ accounts for the computational cost penalty for following the (a_0,a_1) action path.

To estimate the optimal Q-table, a neural network P_2 is used. This network takes the state vectors $CIGT_2(x,a_0)$ as the input. It outputs four predictions based on the CIGT-[1,2,4] structure. Each ith output corresponds to $Q_2(CIGT_2(x,a_0),a_1=i)$.

Then the following regression problem is solved:

$$L_{Q_2} = \mathbb{E}_{a_1\sim\pi_1^*,x}\left[\left(P_2(CIGT_2(x,a_0),a_1)-Q_2(CIGT_2(x,a_0),a_1)\right)^2\right]$$

Note that because the optimal Q table is known, the optimal policy can be computed as follows, and samples can be drawn from it:

$$\pi_1^*(a_1=i|CIGT_2(x,a_0)) = \frac{\exp\{Q_2(CIGT_2(x,a_0),a_1=i)\}}{\sum_{j=1}^{|A^2|}\exp\{Q_2(CIGT_2(x,a_0),a_1=j)\}}$$

In the first layer, the optimal Q-table is calculated using the Bellman equation using the already determined optimal Q-tables from the next layer:

$$Q_1(CIGT_1(x),a_0) = \max_{a_1} Q_2(CIGT_2(x,a_0),a_1)$$

This equation omits an explicit reward term, as all intermediate rewards are considered 0 except in the final CIGT layer. It should also be noted that the state transitions in the model are deterministic.

The inductive process described can be extended to each CIGT layer. After obtaining the optimal Q-table for the first layer, the problem is rewritten as a regression problem. The table Q_1 is estimated using a neural network P_1 and the following loss is minimized:

$$L_{Q_1} = \mathbb{E}_{a_0\sim\pi_0^*,x}\left[\left(P_1(CIGT_1(x),a_0)-Q_1(CIGT_1(x),a_0)\right)^2\right]$$

Again, the optimal policy is built to sample actions, as follows:

$$\pi_1^*(a_0=i|CIGT_1(x)) = \frac{\exp\{Q_1(CIGT_1(x),a_0=i)\}}{\sum_{j=1}^{|A^1|}\exp\{Q_1(CIGT_1(x),a_0=j)\}}$$

In general, the MDP-based routing loss for a generic CIGT structure can be written as:

$$L_Q = \sum_{i=1}^{d} L_{Q_i} \tag{11}$$

Where d is the CIGT depth.

V. EXPERIMENTS

Experiments were carried out on the CIFAR-10 [71] and Fashion-MNIST [72] datasets, on which vanilla CIGT models have already reported results [17].

A. EXPERIMENT DESIGN

A LeNet-based model was used for the Fashion-MNIST and a ResNet110-based model was used for the CIFAR-10 datasets, as in [17]. For supervised Q-learning, Q-table regressor networks are integrated after each CIGT computation layer. The algorithms, both for the Cross-Entropy Search and supervised Q-learning, are applied to a pre-trained network. These algorithms utilize a heavily augmented version of the original training set used in vanilla CIGT training in [17]. The pre-trained networks generate feature representations that serve as inputs for the Q-table regression networks, and only the Q-networks are refined during training. For Cross-Entropy Search, there is no need to add any new network elements, because it is a post-processing algorithm.

The Cutout enhanced versions of the original training sets from [17] were used to train the multi-path routing systems. This strategy aims to lower the pre-trained network accuracy in the training set. If the networks already perform well in the training set, they may not find it necessary to learn additional multi-path routing strategies. This is because single-path routing based on information gain routers may suffice for achieving high accuracy.

For the experiments, the average accuracy, number of parameters used by the network, and the average number of Multiplication and Accumulations (MAC) are reported.

B. CIFAR-10 EXPERIMENTS

In the CIFAR-10 experiments, the ResNet110-based CIGT architecture was utilized as a basis, as in [17], similar to previous research [4], [5]. The multi-path routing experiments were conducted on CIGT-[1,2,4] architectures, that is, three CIGT layers, with 1, 2, and 4 computation units at every layer. For the initial CIGT layer, which consisted of a single unit, 9 ResNet blocks were used. Each of these blocks features two convolutional layers with 3×3 kernels and incorporates a shortcut connection that adds the input of the block to its output without using any bottleneck layers [73]. All the blocks in this section have 16 feature maps. The second CIGT layer is composed of 2 units, each containing 27 ResNet blocks. The first 9 blocks had 12 feature maps each, followed by a reduction in the feature map dimensions after applying a 2×2 stride. The remaining 18 blocks each contained 16 feature maps. The last CIGT layer had 4 units, each with 18

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

TABLE 2. CIFAR-10 results with Cross-Entropy Search and Q-learning based models with accuracy and average number of MAC counts. The extra computation sourced by both the extra routes and Q-learning routers is included in the MAC measurement. All CIGT models have a [1, 2, 4] architecture. SP and MP in SeBoW-A and SeBoW-B architectures mean Single-Path and Multi-Path. s-CIGT stands for Sparse Mixture of Experts.

Method	Avg Acc.	Avg # MAC	# Params
CIGT [17]	93.81%	78.8×10^6	0.64M
s-CIGT $\lambda = 1.0$	**94.11%**	83.3×10^6	0.64M
s-CIGT. $\lambda = 0.99$	94.09%	81.24×10^6	0.64M
s-CIGT. $\lambda = 0.9$	94.05%	79.5×10^6	0.64M
Q-CIGT. $\lambda = 1.0$	94.25%	104.0×10^6	0.78M
Q-CIGT. $\lambda = 0.99$	**94.27%**	91.3×10^6	0.78M
Q-CIGT. $\lambda = 0.9$	94.15%	80.2×10^6	0.78M
AIG-110 [4]	94.24%	153.3×10^6	1.74M
CSGS-110 [5]	**94.36%**	121.5×10^6	1.74M
SeBoW-A (SP) [57]	93.41%	$\approx 73 \times 10^6$	0.7M
SeBoW-A (MP) [57]	93.45%	$\approx 76 \times 10^6$	1.0M
SeBoW-B (SP) [57]	93.93%	$\geq 100 \times 10^6$	1.6M
SeBoW-B (MP) [57]	94.00%	$\geq 150 \times 10^6$	2.7M

ResNet blocks, where a 2×2 stride was introduced at the beginning of these units. For routing decisions, Convolutional Block Attention Modules (CBAM) were used, as introduced in [74]. Samples were processed through 3 CBAM layers with a reduction ratio of 4 and 4×4 average pooling after each routing unit. This step was followed by the calculation of the information gain for the sample. The training procedure for the CIGT models was the same as that indicated in [17].

For the application of the Sparse Mixture of Experts approach with Cross-Entropy Search, the samples from the training set were further augmented with the Cutout augmentation technique. Sigmoid Gaussian Mixtures with 3 components were used for $f(\mathbf{x}, \mathbf{w})$. The Cross-Entropy Search algorithm was trained for 10 iterations. 10000 threshold samples were drawn from $f(\mathbf{x}, \mathbf{w})$ and a quantile parameter of $\rho = 0.05$ is used. The Cutout ratio was maintained at 0.25.

For supervised Q-learning, in the CIFAR-10 experiments, intermediate features from the CIGT network were processed through a 2×2 max pooling step. This is followed by three CBAM layers [74]. The outputs from these CBAM layers went through global average pooling. These were then linearly mapped to a dimension of size 32. Outputs from the two middle layers of the CIGT are fed into a Long Short-Term Memory (LSTM) network [75]. This allows temporal sequence processing. The LSTM has 32 hidden units and, depending on the CIGT configuration produces 2 or 4 action outputs for the CIGT-[1,2,4] structures. To train the Q-Networks, the Adam Optimizer [76] was used. It had an initial learning rate of 0.0001, which decreased linearly with each iteration. The training employed a weight decay rate of 0.00005 and lasted over 20 epochs.

Table 2 presents the results of multi-path routing on the CIFAR-10 dataset. All results are the average of 10 test runs. The Cross-Entropy Search-based s-CIGT, at a balance level of $\lambda = 1.0$, which completely ignores extra computation, achieves 0.3% extra accuracy compared with the single-path routing vanilla CIGT. This comes at the cost of an additional 5.7% of MAC. A Bootstrap t-test [77] gives a p-value smaller than 0.05 against the vanilla CIGT results, for s-CIGT. Simultaneously, the best performing Q-learning-based CIGT at level $\lambda = 0.99$ achieves 94.27% accuracy. The accuracy difference between these best Q-learning and Cross-Entropy Search-based multi-path routing schemes is significant according to the Bootstrap t-test. However, Q-learning tends to require additional computations. At level $\lambda = 0.99$, 15.8% extra MACs were produced compared to the vanilla CIGT.

It can be seen that the Q-Learning-based multi-path learning can outperform the Cross-Entropy-based s-CIGTs with higher accuracy. It was also apparent that at higher levels of the λ balance parameter, the Q-Networks tend to execute extra paths more heavily compared to the Cross-Entropy Search.

The s-CIGT and Q-CIGT algorithms were compared with competitor conditional computation models. These include Adaptive Inference Graphs (AIG) [4], Channel Selection with Gumbel Softmax (CSGS) [5] and Self-born Wiring for Neural Trees (SeBoW) [57]. Both the AIG and CSGS were modified to work on an equivalent ResNet-110 model. The SeBoW model designs a neural tree based on the VGG-13 components. However, the outcome of their self-wiring process produced a deep neural network with comparable complexity [57]. It can be seen that the best-performing Q-CIGT can outperform SeBoW and produce comparable accuracy with AIG-110, but is slightly outperformed by the CSGS-110 model.

It should be noted that AIG can bypass individual ResNet blocks. CSGS takes this further by enabling the omission of specific feature maps through Gumbel-Softmax sampling. On the other hand, CIGT opts to skip whole computation blocks. From that point of view, the search spaces for both AIG and CSGS models are more diverse than those for the CIGT models. Although this limits the possibilities for reducing the size of the model, it offers improved inference efficiency. For instance, AIG-110 and CSGS-110 utilize 82% and 65% of the full, unconditional ResNet-110 model's computational resources, respectively. Q-CIGT $\lambda = 0.99$ operates at only 48.8% of the full ResNet-110 computational budget. In addition, CIGT models tend to be more compact. Both AIG-110 and CSGS-110 leverage the original thick model and apply conditional computation only at inference time. Therefore, they use the original heavyweight ResNet-110 memory footprint, whereas CIGT models are lightweight by design. The SeBoW models, which build a neural tree from VGG-13 blocks, are outperformed by both single-path and multi-path CIGTs in their single-path and multi-path execution modes.

C. FASHION-MNIST EXPERIMENTS

For the Fashion-MNIST tests, CIGT-[1,2,4] LeNet-based models were used, as described in [17]. These models consisted of three convolutional layers and three dense layers. The convolutional layers featured 32, 64, and 128 filters, respectively. Each contains 5×5 kernels, followed by 2×2 max-pooling kernels. The fully connected layers had sizes of 1024,

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

TABLE 3. Fashion-MNIST results with Cross-Entropy Search and Q-learning based models with accuracy and average number of MAC counts. The extra computation sourced by both the extra routes and Q-learning routers is included in the MAC measurement. All CIGT models have a [1, 2, 4] architecture. The s-CIGT stands for Sparse Mixture of Experts.

Method	Avg Acc.	Avg # MAC	# Params
CIGT [17]	**92.52**%	2.7×10^6	0.29M
s-CIGT $\lambda = 1.0$	92.65%	3.2×10^6	0.29M
s-CIGT $\lambda = 0.99$	92.63%	2.8×10^6	0.29M
s-CIGT $\lambda = 0.9$	92.53%	2.7×10^6	0.29M
Q-CIGT $\lambda = 1.0$	**92.74**%	3.7×10^6	0.36M
Q-CIGT $\lambda = 0.99$	**92.74**%	3.7×10^6	0.36M
Q-CIGT $\lambda = 0.9$	92.71%	3.2×10^6	0.36M

512, and 10. The first convolutional layer included 32 feature maps. The second layer, with 64 feature maps, is split into two paths, each having 32 feature maps. For the third convolutional layer, the feature map count was adjusted to 32 for each of the four CIGT computation units. The dimensions of the dense layers were revised to 128 and 64. These adjustments, along with the final classification layer, were integrated into the four computation units of the final block. These networks were used as pre-trained feature extractors. They were not trained further for the multi-path routing experiments.

In the Fashion-MNIST experiments, the Sparse Mixture of Experts inference was used for the CIGT models. The thresholds were learned via Cross-Entropy Search from a highly augmented training set. Sigmoid Gaussian Mixtures with 3 components were used for the Cross-Entropy algorithm. The optimization loop in Algorithm 1 was iterated 10 times. At every iteration, 10000 thresholds were sampled. The quantile variable ρ was set to 0.05. The training set used to train the backbone CIGT model was employed again. However, Cutout augmentation was added to the images during Cross-Entropy optimization. This was performed at a cut ratio of 0.25 [38] to artificially lower the training accuracy, which was approximately 98%.

The results in Table 3 show that multi-path routing improves the results of the CIGT. The results are the average of 10 test runs.

In the Q-learning-based multi-path routing experiments, the CIGT-[1,2,4] networks produce 2 and 4 respective action outputs. For the LeNet variant of CIGT, a singular CBAM layer follows the output. This leads to an LSTM module with a 32-dimensional output for the Q-table regression. Q-Networks were trained using an Adam Optimizer with an initial learning rate of 0.0001. This rate decreases linearly with each training iteration. A weight decay rate of 0.00005 was applied and the training lasted for 20 epochs.

Similar to the approach with Cross-Entropy Search, these reinforcement learning models were trained using the same dataset as for the backbone networks. Cutout optimization was employed to prevent the backbone from generating overly accurate representations.

The results for Q-learning-based multi-path routing in Fashion-MNIST did not match the relative success seen with CIFAR-10. The Q-learning strategy slightly improved the accuracy compared with the single-path, vanilla CIGTs. While the Bootstrap t-test shows that the Q-CIGT's accuracy improvement over the vanilla CIGT is significant, it was not possible to find this significance against Cross-Entropy-based s-CIGT results.

D. DISCUSSION OF RESULTS

Both Cross-Entropy Search and Reinforcement Learning have been applied to multi-path routing in the CIGT model. Both methods yielded improved performance. The improvement in the case of supervised Q-learning on CIFAR-10 is higher than that of the Cross-Entropy Search results. This improvement was less prominent for Fashion-MNIST. This is attributed to two facts: The smaller size of the backbone model that has been used in Fashion-MNIST experiments and the trellis architecture that already enables a more powerful routing capability in this model with over-confident routing decisions. However, it was observed that both approaches produced significant accuracy improvements over vanilla CIGT.

The model performance results of Reinforcement Learning, Cross-Entropy Search, and Vanilla routing-based CIGT architectures were further analyzed, in a comparative fashion. Specifically, for the ResNet110 and CIFAR-10 dataset, the average accuracy of the Reinforcement Learning method (Q-CIGT) was 94.24% across balance levels $\lambda = 1.0, \lambda = 0.99$, and $\lambda = 0.9$. This was compared to 94.08% for the Cross-Entropy-based method (s-CIGT) and 93.81% for the vanilla CIGT. The best Q-CIGT performance was obtained at $\lambda = 0.99$ with 94.27% accuracy. The best performance of s-CIGT was 94.11%, in comparison. The performance gap between the Reinforcement Learning and Cross-Entropy Search methods was significant, as confirmed by a Bootstrap t-test yielding a p-value of 2×10^{-6}. For the Fashion-MNIST dataset using LeNet, Q-CIGT performed slightly better than s-CIGT did. The average accuracies were 92.73% and 92.60% across all balance levels $\lambda = 1.0, \lambda = 0.99$, and $\lambda = 0.9$. The highest Q-CIGT accuracy was 92.74% at $\lambda = 0.99$. The highest accuracy of s-CIGT was 92.65% at $\lambda = 1.0$. However, the Bootstrap t-test showed no significant difference between the best-performing Q-CIGT and s-CIGT models at a significance level of $p = 0.05$. This was largely caused by variance in the Q-CIGT results. Nonetheless, both models significantly surpassed the vanilla CIGT performance, with very small p-values, in both datasets. These results are presented in Table 4.

In terms of computational efficiency, Reinforcement Learning routing generally results in more additional computation than Cross-Entropy Search. For the ResNet110 model using the CIFAR-10 dataset, the Cross-Entropy Search adds an average of 3.2% more computations. This is measured in terms of Multiplication and Accumulation (MAC) across the three balance levels. The top s-CIGT model at level $\lambda = 1.0$, which disregards penalties for extra computation, generates 5.7% more computations than the standard CIGT. The Re-

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

TABLE 4. Significance comparison and resulting p-values between model performances of Reinforcement Learning (Q-CIGT), Cross-Entropy Search-Based (s-CIGT), and Vanilla routing (CIGT) approaches, using Bootstrap t-test [77].

ResNet110 CIFAR-10	s-CIGT $\lambda = 1.0$	CIGT
Q-CIGT $\lambda = 0.99$	$p = 2 \times 10^{-6}$	$p = 1 \times 10^{-6}$
s-CIGT $\lambda = 1.0$	-	$p = 2 \times 10^{-5}$
LeNet Variant Fashion-MNIST	**s-CIGT** $\lambda = 1.0$	**CIGT**
Q-CIGT $\lambda = 0.99$	$p = 5.6 \times 10^{-2}$	$p = 0.0$
s-CIGT $\lambda = 1.0$	-	$p = 0.0$

TABLE 5. Average increase in the MAC per sample, for performing Reinforcement Learning (Q-CIGT) and Cross-Entropy Search-Based (s-CIGT) routing approaches, compared to Vanilla routing (CIGT)

ResNet110 CIFAR-10	vs. CIGT
Q-CIGT $\lambda = 0.99$	15.9%
s-CIGT $\lambda = 1.0$	5.7%
LeNet Variant Fashion-MNIST	**vs. CIGT**
Q-CIGT $\lambda = 0.99$	37.0%
s-CIGT $\lambda = 1.0$	18.5%

inforcement Learning-based method increases computation by an average of 16.5% across all balance levels. The best-performing Reinforcement Learning model at $\lambda = 0.99$ generates a 15.9% higher computational load (Table 5).

For the Fashion-MNIST dataset and LeNet variant architecture, the additional computations for both methods exceed those of the ResNet110 and CIFAR-10 setups. Cross-Entropy Search results in 7.4% more computations across all the balance levels. The highest-performing s-CIGT model at $\lambda = 1.0$ causes an 18.5% increase in average MACs. The Reinforcement Learning approach leads to a 30.8% increase in computation across balance levels, with the best model at $\lambda = 0.99$ adding 37% more MACs per sample (Table 5).

The experiments reveal that Reinforcement Learning-based routing enhances model performance over Cross-Entropy Search. This enhancement comes at the cost of increased computation loads. This added computation from Reinforcement Learning is mainly because neural networks serve as policy networks. The addition comes from both their execution and overfitting behavior by attempting to maximize the number of routes to maintain high accuracy. In the Q-learning approach, as λ increases, the computational penalty term L_{MAC}'s effect decreases. Optimal Q-table entry differences, caused only by that penalty, become very small. This allows neural networks to more easily find local optima that open as many paths as possible, thereby optimizing the maximum accuracy. Conversely, Cross-Entropy Search fine-tunes the computation costs by directly adjusting the probability thresholds. This approach offers more precise control over additional computations. Moreover, it does not require the use of additional neural units. This is a post-processing approach based on the trained network outputs.

In terms of model performance, Reinforcement Learning is expected to yield a higher accuracy because it theoretically enables each sample to select the most efficient routing. Conversely, the Cross-Entropy Search sets the global thresholds based on the training dataset. This advantage of Reinforcement Learning is notably apparent in the ResNet110 model and CIFAR-10 dataset. However, with the smaller LeNet variant and Fashion-MNIST dataset, the performance improvement from Reinforcement Learning compared to Cross-Entropy Search is less pronounced.

The discrepancy between the CIFAR-10 and Fashion-MNIST performance results is analyzed in more detail: By closely inspecting the router unit outputs in the two models, a key difference is observed. The LeNet Variant trained with Fashion-MNIST produced overly confident routing results. These results may potentially limit improvements in model performance. The reader is referred to the Supplementary Material for detailed analysis.

One of the advantages of CIGT is its interpretability: CIGT tends to route semantically similar samples along the same path. However, some samples are difficult to classify and get misrouted. Multi-path routing routes these difficult samples into multiple paths. As the number of paths through which a sample is routed increases, the model's decision-making process becomes more intricate, involving multiple expert opinions, thereby reducing the interpretability of the model. A more detailed analysis with visual examples regarding interpretability and multi-path relationship is presented in the Supplementary Material.

During the adaptation of the multi-path routing approach to the vanilla CIGT method in [17], numerous challenges have been encountered both theoretically and practically. These are as follows:

- Bayesian Optimization has been used in previous work to set optimal probability thresholds for routing in a tree structure rather than a trellis [16]. Bayesian Optimization relies on sequentially updating a surrogate model (often a Gaussian Process) with new data points based on an acquisition function. [16] used the Expected Improvement for this purpose. Users cannot explicitly define the data-generating process for the samples. Cross-Entropy Search, allows one to directly define how samples are generated using a specific probability distribution as a data generator.
- The direct adaptation of the threshold optimization as a gradient-based procedure was investigated. The main obstacle in this regard is the lack of a definable gradient used in path routing decisions in the form of $p(Z = k|x) \geq t_k$. Consequently, the multi-path routing challenge has been treated as an MDP, leading to the development of the Reinforcement Learning-based Q-CIGT algorithm.
- In the initial trial with a Reinforcement Learning strategy, the REINFORCE algorithm with baselines was

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

employed [22] using the same policy network structure outlined for the Q-learning method. Unfortunately, this method struggles with convergence issues and high variability, resulting in inadequate generalization. The problem was reframed as a supervised regression task, with the Q-tables serving as direct targets for regression. This approach led to the creation of the Q-CIGT model.

- In the implementation of Cross-Entropy Search, 10000 thresholds were sampled. This sampling robustly identifies the optimal $(1-\rho)$-quantile for the maximum-likelihood optimization of $f(\mathbf{x}, \mathbf{w})$. However, this necessitates expensive evaluations of the training set for every threshold. A naive implementation would involve 10000 neural network epochs on the training set for each iteration of the Cross-Entropy Search, which is extremely costly. Consequently, the neural network was evaluated for a single time on a highly augmented training set. The routing probabilities and logits of the leaf node classification layers were recorded. For each set of sampled thresholds, network execution was simulated using the recorded buffers. This approach avoids executing the full network end-to-end and results in a significantly more efficient optimization process.

VI. CONCLUSION

In this study, two novel multi-path routing algorithms have been presented, that are based on the Conditional Information Gain Trellis conditional computation model, first introduced in [17]. The first algorithm is based on a Cross-Entropy Search, which finds the optimal probability thresholds using a validation set, using a loss function that balances extra accuracy and computation. Whenever a router unit in the CIGT structure generates probabilities larger than these thresholds, a sample is routed into multiple routes, resulting in a Sparse Mixture of Experts mode of inference [16]. The second algorithm is based on a supervised Q-learning model, which is made possible by the finite horizontal structure of the CIGT models facilitated by its practical use cases. This algorithm can determine the individual routing behaviors for every sample. Experiments show that both methods improve the vanilla Resnet110-based CIGT model for the CIFAR-10 dataset. On the other hand, while there is also an improvement for the LeNet variant-CIGT model on the Fashion-MNIST dataset, these improvements are smaller, possibly because of over-confident router behaviors.

In the development of the Q-CIGT model, the core CIGT network has been used as a static feature extractor for policy networks without updating it through gradients. The concept of fully integrating the training of the CIGT with the policy networks—a process involving optimal hyperparameter settings or a novel training sequence that alternates which network is updated— is a promising avenue for future research. In addition, since the rise of large language models in the last couple of years, efficient training and inference with these models have become a very active field of research. The mixture of experts layers are now commonly used in numerous state-of-the-art language models. The adaptation of the CIGT architecture and the multi-path routing algorithms introduced in this article for such transformer-based models is also an interesting direction for future work.

REFERENCES

[1] Y. Bengio, "Deep learning of representations: Looking forward," in *Proceedings of the Statistical Language and Speech Processing*, A.-H. Dediu, C. Martín-Vide, R. Mitkov, and B. Truthe, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–37.

[2] C. Murdock, Z. Li, H. Zhou, and T. Duerig, "Blockout: Dynamic model selection for hierarchical deep networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2583–2591.

[3] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8817–8826.

[4] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 730–741, Mar 2020.

[5] C. Herrmann, R. S. Bowen, and R. Zabih, "Channel selection using gumbel softmax," in *European Conference on Computer Vision*. Springer, 2020, pp. 241–257.

[6] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, and A. Criminisi, "Decision forests, convolutional networks and the models in-between," 2016, arXiv:1603.01250 [cs].

[7] L. Liu and J. Deng, "Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018*. AAAI press, 2018, pp. 3675–3682.

[8] Z. Yan, H. Zhang, R. Piramuthu, V. Jagadeesh, D. DeCoste, W. Di, and Y. Yu, "Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2740–2748.

[9] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR)*, 2016, pp. 2464–2469.

[10] B. Wójcik, A. Devoto, K. Pustelnik, P. Minervini, and S. Scardapane, "Adaptive computation modules: Granular conditional computation for efficient inference," 2023, arXiv:2312.10193 [cs].

[11] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural Comput.*, vol. 6, no. 2, pp. 181–214, Mar. 1994.

[12] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[13] C. Baykal, N. Dikkala, R. Panigrahy, C. Rashtchian, and X. Wang, "A theoretical view on sparsely activated networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 071–30 084, 2022.

[14] D. Eigen, M. Ranzato, and I. Sutskever, "Learning factored representations in a deep mixture of experts," in *Proceedings of the International Conference on Learning Representations (ICLR), Workshop Track*, 2014.

[15] U. C. Bicici, C. Keskin, and L. Akarun, "Conditional information gain networks," in *Proceedings of the 24th International Conference on Pattern Recognition (ICPR)*, 2018, pp. 1390–1395.

[16] U. C. Bicici and L. Akarun, "Conditional information gain networks as sparse mixture of experts," *Pattern Recognition*, vol. 120, p. 108151, 2021.

[17] U. C. Bicici, T. H. S. Meral, and L. Akarun, "Conditional information gain trellis," 2024, arXiv:2402.08345 [cs].

[18] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of operations research*, vol. 134, pp. 19–67, 2005.

[19] Y. Bengio, N. Leonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, arXiv:1308.3432 [cs].

[20] F. Li, N. Neverova, C. Wolf, and G. Taylor, "Modout: Learning to fuse modalities via stochastic regularization," *Journal of Computational Vision and Imaging Systems*, vol. 2, no. 1, 2016.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

[21] E. Bengio, P. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," in *Proceedings of the International Conference on Learning Representations (ICLR), Workshop Track*, 2016.

[22] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[23] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[24] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[25] X. Wang, F. Yu, Z.-Y. Dou, T. Darrell, and J. E. Gonzalez, "Skipnet: Learning dynamic routing in convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018, pp. 409–424.

[26] T. Krishna, A. K. Rai, Y. A. Djilali, A. F. Smeaton, K. McGuinness, and N. E. O'Connor, "Dynamic channel selection in self-supervised learning," 2022, arXiv:2207.12065 [cs].

[27] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, "Slimmable neural networks," 2018, arXiv:1812.08928 [cs].

[28] C. Li, G. Wang, B. Wang, X. Liang, Z. Li, and X. Chang, "Ds-net++: Dynamic weight slicing for efficient inference in cnns and vision transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4430–4446, 2022.

[29] J. Yu and T. S. Huang, "Universally slimmable networks and improved training techniques," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1803–1811.

[30] C. Xiong, X. Zhao, D. Tang, K. Jayashree, S. Yan, and T. Kim, "Conditional convolutional neural network for modality-aware face recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 3667–3675.

[31] R. Tanno, K. Arulkumaran, D. Alexander, A. Criminisi, and A. Nori, "Adaptive neural trees," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6166–6175.

[32] Y. Han, Y. Pu, Z. Lai, C. Wang, S. Song, J. Cao, W. Huang, C. Deng, and G. Huang, "Learning to weight samples for dynamic early-exiting networks," in *European Conference on Computer Vision*. Springer, 2022, pp. 362–378.

[33] R. G. Pacheco and R. S. Couto, "Inference time optimization using branchynet partitioning," in *2020 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2020, pp. 1–6.

[34] C.-H. Chiang, P. Liu, D.-W. Wang, D.-Y. Hong, and J.-J. Wu, "Optimal branch location for cost-effective inference on branchynet," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 5071–5080.

[35] M. Ayyat, T. Nadeem, and B. Krawczyk, "Classynet: Class-aware early exit neural networks for edge devices," *IEEE Internet of Things Journal*, 2023.

[36] J.-Y. Jeon, X. T. Nguyen, S. Ryu, and H.-J. Lee, "Usdn: A unified sample-wise dynamic network with mixed-precision and early-exit," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 646–654.

[37] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, and G. Huang, "Resolution adaptive networks for efficient inference," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2369–2378.

[38] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, arXiv:1708.04552 [cs].

[39] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 702–703.

[40] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[41] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic data augmentation using gan for improved liver lesion classification," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, 2018, pp. 289–293.

[42] A. Figueira and B. Vaz, "Survey on synthetic data generation, evaluation methods and gans," *Mathematics*, vol. 10, no. 15, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/15/2733

[43] Y. Akkem, S. K. Biswas, and A. Varanasi, "A comprehensive review of synthetic data generation in smart farming by using variational autoencoder and generative adversarial network," *Engineering Applications of Artificial Intelligence*, vol. 131, p. 107881, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197624000393

[44] T.-W. Chin, A. S. Morcos, and D. Marculescu, "Joslim: Joint widths and weights optimization for slimmable neural networks," in *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part III 21*. Springer, 2021, pp. 119–134.

[45] Z. Jiang, C. Li, X. Chang, L. Chen, J. Zhu, and Y. Yang, "Dynamic slimmable denoising network," *IEEE Transactions on Image Processing*, vol. 32, pp. 1583–1598, 2023.

[46] C. M. Bishop and M. Svensen, "Bayesian hierarchical mixtures of experts," 2012, arXiv:1212.2447 [cs].

[47] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, arXiv:1503.02531 [cs].

[48] S. Lee, S. Purushwalkam, M. Cogswell, D. J. Crandall, and D. Batra, "Why M heads are better than one: Training a diverse ensemble of deep networks," 2015, arXiv:1511.06314 [cs].

[49] K. Ahmed, M. H. Baig, and L. Torresani, "Network of experts for large-scale image categorization," in *Computer Vision - Proceedings of the ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 516–532.

[50] S. E. Chazan, J. Goldberger, and S. Gannot, "Speech enhancement using a deep mixture of experts," 2017, arXiv:1703.09302 [cs].

[51] T. Furlanello, Z. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1607–1616.

[52] L. Kirsch, J. Kunze, and D. Barber, "Modular networks: Learning to decompose neural computation," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 2414–2423.

[53] N. Dikkala, N. Ghosh, R. Meka, R. Panigrahy, N. Vyas, and X. Wang, "On the benefits of learning to route in mixture-of-experts models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 9376–9396.

[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[55] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[56] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *The Journal of Machine Learning Research*, vol. 23, no. 1, pp. 5232–5270, 2022.

[57] Y. Chen, F. Mao, J. Song, X. Wang, H. Wang, and M. Song, "Self-born wiring for neural trees," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5047–5056.

[58] C. M. Bishop and H. Bishop, *Deep Learning - Foundations and Concepts*, 1st ed., S. Cham, Ed., 2023.

[59] P. E. David, P. R. Chelliah, and P. Anandhakumar, "Chapter nine - reshaping agriculture using intelligent edge computing," in *Applying Computational Intelligence for Social Good*, ser. Advances in Computers, P. E. David and P. Anandhakumar, Eds. Elsevier, 2024, vol. 132, pp. 167–204. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065245823000736

[60] Y. Akkem, S. K. Biswas, and A. Varanasi, "Smart farming using artificial intelligence: A review," *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105899, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197623000830

[61] K. Vidhya, S. George, P. Suresh, D. Brindha, and T. J. Jebaseeli, "Agricultural farm production model for smart crop yield recommendations using machine learning techniques," *Engineering Proceedings*, vol. 59, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2673-4591/59/1/20

[62] Y. Akkem, S. K. Biswas, and A. Varanasi, "Streamlit application for advanced ensemble learning methods in crop recommendation system – a review and implementation," *Indian Journal of Science and Technology*, vol. 16, pp. 4688–4702, 2023.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3394805

[63] S. Balaji, N. K. K. Raju, S. Tarun, R. Karthikeyan, T. Shankar, and R. Santhakumar, "Precision agriculture crop recommendation system using iot and machine learning," in *2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*. IEEE, 2023, pp. 1–6.
[64] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, "A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7366–7390, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157822000970
[65] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of field robotics*, vol. 37, no. 3, pp. 362–386, 2020.
[66] D. Bogdoll, M. Nitsche, and J. Zollner, "Anomaly detection in autonomous driving: A survey," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2022, pp. 4487–4498. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPRW56347.2022.00495
[67] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys And Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.
[68] Q. Tang, J. Liang, and F. Zhu, "A comparative review on multi-modal sensors fusion based on deep learning," *Signal Processing*, vol. 213, p. 109165, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168423002396
[69] S. Baek, K. I. Kim, and T.-K. Kim, "Deep convolutional decision jungle for image classification," 2017, arXiv:1706.02003 [cs].
[70] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[71] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," in *Master's thesis, Department of Computer Science, University of Toronto [dataset]*, 2009.
[72] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms [dataset]," 2017. [Online]. Available: http://arxiv.org/abs/1708.07747
[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
[74] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.
[75] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[76] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*. Ieee, 2018, pp. 1–2.
[77] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, ser. Monographs on Statistics and Applied Probability. Boca Raton, Florida, USA: Chapman & Hall/CRC, 1993, no. 57.



UFUK CAN BICICI received his B.S. and Master degrees from Marmara University in 2009 and from Bogazici University in 2012, in Computer Science. He is currently a Phd candidate in Bogazici University, in Computer Science. His research topics are deep learning and conditional deep neural networks.



LALE AKARUN Lale Akarun is a professor of Computer Engineering at Bogazici University. She received the PhD degree in Electrical Engineering from the Polytechnic School of Engineering of NYU, in 1992. She has been a faculty member at Bogazici University, Istanbul since 1993. She has served as Department Head of Computer Engineering (2010-2012) and Vice Rector for Research (2012-2016). Her research areas are image processing, computer vision and computer graphics. She has supervised 70 graduate theses and published more than 200 scholarly papers in scientific journals and refereed conferences. She has conducted research projects in biometrics, face recognition, hand gesture recognition, human-computer interaction, and sign language recognition. She is a senior member of IEEE Computer Society and Biometrics Council. She has served as the vice president of the International Association of Pattern Recognition (IAPR) since 2018

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see https://creativecommons.org/licenses/by-nc-nd/4