Exploiting Class-Granular Classifications to Improve Inference Efficiency with Dynamic DNNs

Benjamin Sanati
University of Southampton
bes1g19@soton.ac.uk

Lei Xun
University of Southampton
l.xun@soton.ac.uk

Mingyu Hu
University of Southampton
mh1u20@soton.ac.uk

Hengrui Zhao
University of Southampton
hz20u22@soton.ac.uk

Jonathon Hare
University of Southampton
jsh2@ecs.soton.ac.uk

Geoff Merrett
University of Southampton
gvm@ecs.soton.ac.uk

***Abstract*—Recent advancements in deep learning have achieved exceptional performance on the multi-class classification problem. However, these techniques are often computationally expensive and incur undesired latency overheads. To mitigate these effects, DyNNs were proposed as an architectural modification to vanilla DNNs, allowing for the dynamic adjustment of compute at run-time.**

Although DyNNs successfully alter computational overheads at run-time based on problem complexity, they assume that each problem has only one acceptable solution - one semantic complexity. However, semantically speaking, each image can be associated with a range of acceptable classifications. In the context of a multi-class classification problem, a set of solutions exist within the semantic hierarchy, exemplified by sets like {poodle, dog, animal}.

This paper highlights the importance of hierarchical learning by investigating the impact of dynamic granularity adjustments within the context of classification. This investigation is performed within the framework of a novel early-exiting architecture named Gran-HBN.

Gran-HBN is used for a thorough analysis of the accuracy-specificity trade-off and to show that the classification uncertainty decreases for both coarse and fine classifications as we delve deeper into the model. Additionally, overlaps in both fine label and coarse label classification uncertainties are identified at each exit. This indicates the need for a nuanced approach, promoting a combination of hierarchical classifications at each exit.

This leads to the development of AG-HBN, a proof-of-concept architecture designed to dynamically adjust the classification granularity based on the complexity of an input image. AG-HBN is used to navigate the trade-off between accuracy and specificity, enabling earlier branches to achieve late-level classification accuracy while utilizing fewer computational resources. AG-HBN is shown to capitalize on the accuracy-specificity trade-off for individual images, enhancing classification flexibility, information density and preserving/improving the granular complexity at each branch compared to prior models [1]–[3].

I. Introduction

Dynamic neural networks (DyNNs) effectively exploit the trade-off between computational resources and accuracy, making them well-suited for deployment on resource-constrained devices.

One implementation of dynamic networks involves networks that achieve dynamic depth through early-exiting mechanisms [4]. These architectures introduce a trade-off between model size and accuracy, which can be utilized during inference.

Nevertheless, deploying DyNN models on resource-constrained devices in time-sensitive environments presents numerous challenges. Such a model would require low latency, low computational requirements, and high accuracy classifications. However, when exiting at an early branch in an early-exiting DyNN, the improvement in latency comes at the cost of a reduction in classification accuracy.

Upon closer examination, it is evident that classifications often have a hierarchy of acceptable semantic outcomes. Therefore, implementing DyNNs with dynamic classification granularities leverage the hierarchy of acceptable semantic results during inference, enabling the preservation of late-exit classification accuracies at earlier exits. This offers a more desirable trade-off between latency and specificity, instead of the less desirable latency-classification accuracy trade-off made in [5].

To implement such a model, we present Granular Hierarchical-BranchyNet (Gran-HBN): a model capable of adjusting classification granularity over a two-level semantic hierarchy during inference. Within the investigation, we introduce and define three key terms:

- **Information density**: the amount of classification information encapsulated within a branch
- **Granular complexity**: the degree of semantic classification complexity a branch is capable of
- **Classification flexibility**: the model's ability to dynamically adjust its hierarchical classification levels at a branch

Gran-HBN allows for an investigation into the importance of hierarchical learning via the following architectural attributes:

- Enhanced information density and classification flexibility, achieved via a fine-tolerance hyperparameter τ
- Control over the trade-off between latency and specificity with granular exits
- Control over the trade-off between latency and classification accuracy (for a given semantic complexity) with early-exits

Subsequently, we present Autonomous Granular-HBN (AG-HBN): a model that autonomously selects a classification granularity during inference, exemplifying an architecture that automatically tunes the classification complexity based on the input image.

II. Background and Related Prior Work

In contrast to static NNs, DyNNs utilize adaptive inference to efficiently adapt either the model architecture or the network parameters based on the input sample [4], [6]. Sample-wise DyNNs with dynamic architectures reconfigure their model architectures during the inference process, optimizing computational resource allocation according to the input sample [4].

A common dynamic architecture used to acquire dynamic depth is early-exiting [5], [7], [8]. Early-exiting hinges on two ideas: (i) that feature extraction complexity scales with layer depth, and (ii) that not all input samples require equally complex feature extraction for classification [4]. It follows that less complex input samples can be classified with earlier exits, while more complex input samples use later exits for sufficiently fine features to be extracted and classified [2]–[5], [9], [10].

Early-exiting is thus realized with the use of a backbone network that incorporates intermediate classifiers and employs confidence-based criteria to achieve adaptive early-exiting during inference [4], [5], [9].

A. BranchyNet - An Early-Exiting Architecture

BranchyNet is an early-exiting architecture consisting of a backbone network with intermediate classifiers. BranchyNet is trained by solving the joint optimization problem of the weighted loss of all exit points. As such, each exit regularizes the other, improving the test accuracy by reducing over-fitting. Branches also provide additional gradient signals during back propagation resulting in a reduction of the vanishing gradient effect [5].

1) Training Procedure: To train BranchyNet, a joint optimization problem is solved on the loss of the network [5]. The forward propagation step is implemented by (i) acquiring the prediction and loss at each exit in the network, then (ii) calculating the overall loss of the network.

The prediction of the network at the n-th exit, $\hat{\mathbf{y}}_n$, is used to calculate the loss at the n-th exit via the multi-class cross-entropy, as shown in equation (1).

$$\begin{aligned} \hat{\mathbf{y}}_n &= \text{softmax}(f_n(\mathbf{x};\theta)) \\ L_n(\hat{\mathbf{y}}_n, \mathbf{y};\theta) &= -\frac{1}{|C|}\sum_{c\in C} y_c \log \hat{y}_{n,\,c} \end{aligned} \tag{1}$$

Now that both the prediction and the loss at each of the N exits are acquired, the overall loss can be derived by performing a weighted sum of the loss functions at each exit, as shown in equation (2).

$$L_{branchynet}(\hat{\mathbf{y}}, \mathbf{y};\theta) = \sum_{n=1}^{N} w_n L_n(\hat{\mathbf{y}}_n, \mathbf{y};\theta) \tag{2}$$

The gradient of the network loss, $\nabla L_{branchynet}(\hat{\mathbf{y}}_n, \mathbf{y}_n;\theta)$, can then be used in gradient descent to update the parameters.

2) Fast-Inference: When the network is trained, BranchyNet reduces the computational cost of inference by early-exiting. The exit taken is the earliest exit which complies with its confidence based criteria. In the case of BranchyNet, the entropy [11] is employed as the exiting determinant, where an exit is taken if the entropy, shown in (3), at the exit is less than or equal to a threshold, Th.

$$H(\hat{\mathbf{y}}_n) = -\sum_{c\in C} y_c \ln \hat{y}_c \tag{3}$$

3) Design: When designing a BranchyNet network, the main design considerations include:

- Locations of branch points
  - Depend on the complexity of the dataset: more complex datasets require the branches to be placed later in the network
  - [9] proposes an algorithm which finds the optimal branch location by formulating an NP-complete optimization problem for the optimal branch location and finding the optimal solution
- Structure of a branch
  - Earlier exits must require less computation than later exits
  - Earlier branches should have more layers than later branches
  - The inclusion of fully-connected (FC) layers facilitate the combination of local and global features, resulting in more discriminative features
- Hyper-parameter sensitivity
  - The weights used in the joint loss, w_n, should give more weighting to earlier branches, improving the accuracy of the later branches due to added regularization
  - The entropy threshold, Th, should be selected such that a desired trade-off between computation and accuracy is acquired

These methods were used in [5] to acquire the network architecture in figure 1.

B. Class Hierarchy Utilization

Current state-of-the-art CNNs regarding image classification revolve around the use of deep CNN architectures with flat (non-hierarchical) classifiers. As the size of the dataset and the number of object categories increases, the variance in the required feature extraction complexity for classification also increases [2], [3], [12], [13].

DyNNs present the opportunity to leverage semantic hierarchies during classification, while leveraging early-exiting branches to reduce the computational load during inference. To aid this, many image classification datasets are constructed from classes with semantic hierarchies which can be used to produce semantic hierarchical classification trees. Examples

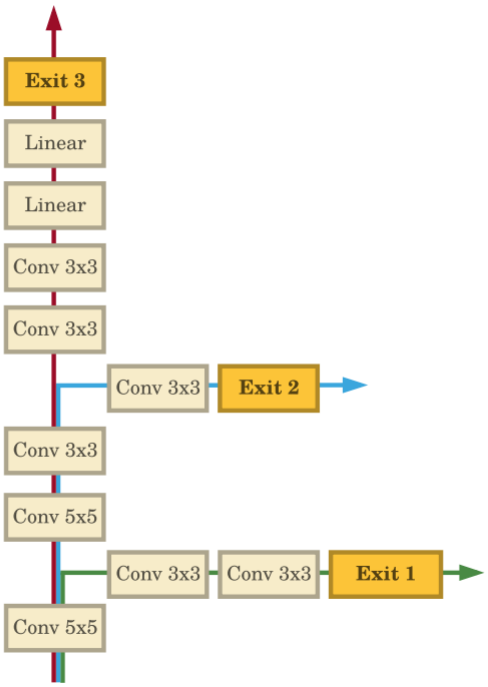

Fig. 1. A B-AlexNet example architecture. B-AlexNet is a BranchyNet implementation which uses the AlexNet model [1] as the backbone network, and introduces intermediate classifiers that allow for adaptive early-exiting during inference.
Source: [5]

include the CIFAR-100 dataset [14] and the ImageNet dataset [15] that is constructed from WordNet [16].

Several studies have successfully utilized the hierarchical semantic structure of such datasets [2], [3], [10], [13], [17]–[19]. These studies observed an improvement in the representation power of the NN, however, sacrifice the information density and granularity complexity of early branch classifications, losing fine label information due to coarse predictions.

Hierarchical classification was achieved with early-exiting by allowing the network to classify coarse classes early in the network and fine classes later in the network [2], [3], [18]. This implies that coarse classifications exhibit lower uncertainty (lower entropy) compared to their fine classification counterparts, a claim that is affirmed in figure 5. In response, subsequent loss functions, such as those proposed by [20], have been developed to harness hierarchical relationships during training.

While surveying hierarchical classification problems, [21] presented a unifying framework to classify these types of problems as either mandatory or non-mandatory leaf node predictors, coupled with either local or global classifiers.

C. Discussion

Hierarchical methods pose the potential to improve representation power by exploiting semantic class hierarchies during image classification. Thus far, the precedent has been to implement early-exiting DyNNs in a hierarchically equivalent manner to the semantic class hierarchy [2], [3], [10], [13], [17]–[19]. In this setup, coarse classes are classified early in the network and fine classes are classified later in the network. However, sample complexity fluctuates across the data population, indicating that this methodology is not optimal.

In light of this, this paper will present two models: Gran-HBN, designed to facilitate an exploration into the significance of hierarchical learning, and AG-HBN, serving as a proof-of-concept model to demonstrate its feasibility. The exploration will involve identifying potential relationships between the granularity of semantic classifications and the complexity of images. Additionally, it will identify the advantages and drawbacks associated with the use of hierarchical exits over flat ones, while highlighting potential future research directions regarding hierarchical classifiers.

III. Gran-HBN

Gran-HBN is an early exiting architecture designed to facilitate dynamic adjustments in classification granularity during inference. This allows for an investigation into granular classifications by increasing the information density and classification flexibility of a model at run-time, while maintaining each exits granular complexity.

A. Architecture

Granular Hierarchical-BranchyNet (Gran-HBN) performs adaptable classifications in granularity for a 2-level semantic hierarchy classification problem by splitting each "super branch" of a BranchyNet model [5] into two "granular branches", as shown in figure 2. Each of the granular branches is responsible for classifying distinct levels of the semantic hierarchy, performed during inference with the use of the fine-tolerance hyper-parameter, τ.

B. Training

Gran-HBN is trained by solving a joint optimization problem on the loss of the network, which is a weighted sum of the loss at each exit point. Due to the granularity branches of Gran-HBN introducing more than one exit at each branch, some alterations to the BranchyNet loss are required to derive the Gran-HBN loss.

The forward propagation step is performed by acquiring the prediction probabilities, $\hat{\mathbf{y}}_n^{(g)}$, for each exit, $g = \{\text{coarse, fine}\}$, on a branch, n, as shown in (4).

$$\hat{\mathbf{y}}_n^{(g)} = \text{softmax}(f_n^{(g)}(\mathbf{x};\ \theta)) \tag{4}$$

The loss at each exit can then be determined using the multi-class cross-entropy, as detailed in (5).

$$L_n^{(g)}(\hat{\mathbf{y}}_n^{(g)}, \mathbf{y}; \theta) = -\frac{1}{|C|} \sum_{c \in C} y_c \log(\hat{y}_{n,\,c}^{(g)}) \tag{5}$$

Then the sum of the losses at each exit can be taken to determine the overall loss of the branch (6).

$$L_n(\hat{\mathbf{y}}_n, \mathbf{y}; \theta) = \sum_{g \in G} L_n^{(g)}(\hat{\mathbf{y}}_n^{(g)}, \mathbf{y}; \theta) \tag{6}$$

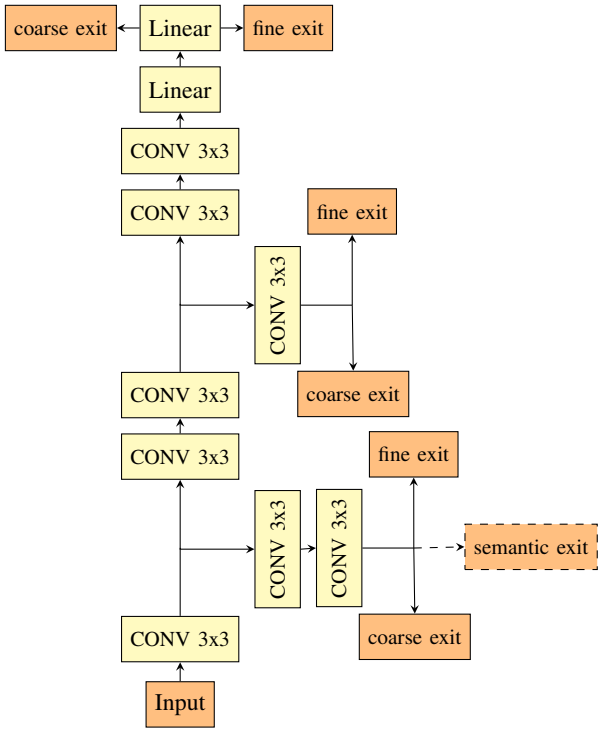

Fig. 2. The Gran-HBN model architecture. This uses the CIFAR-100 AlexNet model as a backbone with intermediate classifiers, similar to [5]. The fine-tolerance hyper-parameter is used to enable adaptable classifications in granularity at run-time. The dashed addition represents the architectural change required to implement AG-HBN, detailed in section IV.

The overall network loss can then be acquired with (7), where a weighted sum of the losses at each branch is taken. The network loss introduces a joint optimization problem which is used in gradient descent to update the parameters, θ.

$$L_{\text{Gran-HBN}}(\hat{\mathbf{y}}_n, \mathbf{y}_n; \theta) = \sum_n w_n L_n(\hat{\mathbf{y}}_n, \mathbf{y}_n; \theta)$$
$$\min_\theta L_{\text{Gran-HBN}}(\hat{\mathbf{y}}_n, \mathbf{y}_n; \theta) \tag{7}$$

C. Inference

During inference, the computational cost can be reduced by utilizing early-exiting with confidence-based criteria. Similar to BranchyNet [5], the threshold hyper-parameter is used as an exiting determinant.

The design of the network introduces the potential to control the number of fine or coarse exits taken at any branch. As such, the fine-tolerance hyper-parameter, τ, is introduced to control the behaviour during inference. The fine-tolerance determines the classification granularity by determining which granular exit (coarse or fine exit) should be taken. The desired behaviour of τ is as follows:

- As $\tau \to -\infty$, the number of fine-exits should tend to 0
- As $\tau \to +\infty$, the number of coarse-exits should tend to 0

Algorithm 1 achieves this behaviour for a 2-level class hierarchy, by providing an extra condition to the exiting criteria on lines 6 and 13. Assuming the criteria for the threshold is met by both exits of differing classification granularity, the coarse, g_c, and fine, g_f, exit entropy's are compared and the exit with a lower entropy is taken. To provide classification flexibility, the fine-tolerance hyper-parameter is subtracted by the fine classification entropy. As such, if $\tau \to -\infty$, the coarse entropy cannot be less than the fine entropy, and the fine classification exit is taken. If $\tau \to +\infty$, the fine entropy cannot be less than the coarse entropy, and the coarse classification exit is taken.

Algorithm 1 Gran-HBN fast-inference pseudocode. Note that g_f denotes a fine label and g_c denotes a coarse label.

1: **procedure** GRAN_HBN_FAST_INFERENCE(**x**, Th, τ))
2: **for** `n=1..N-1` **do** ▷ For each branch
3: $\mathbf{z}_n^{(g)} \leftarrow f_n^{(g)}(\mathbf{x})$
4: $\hat{\mathbf{y}}_n^{(g)} \leftarrow \texttt{softmax}(\mathbf{z}_n^{(g)})$
5: $e_n^{(g)} \leftarrow \text{entropy}(\hat{\mathbf{y}}_n^{(g)})$
6: **if** $(e_n^{(g_f)} < Th)$ **and** $(e_n^{(g_f)} - \tau < e_n^{(g_c)})$ **then**
7: **return** $\arg\max \hat{\mathbf{y}}_n^{(g_f)}$
8: **else if** $(e_n^{(g_c)} < Th)$ **then**
9: **return** $\arg\max \hat{\mathbf{y}}_n^{(g_c)}$
10: **end if**
11: **end for**
12: ▷ Always exit on last branch if no early-exit is taken
13: **if** $(e_N^{(g_f)} - \tau < e_N^{(g_c)})$ **then**
14: **return** $\arg\max \hat{\mathbf{y}}_N^{(g_f)}$
15: **else**
16: **return** $\arg\max \hat{\mathbf{y}}_N^{(g_c)}$
17: **end if**
18: **end procedure**

IV. AG-HBN

To introduce a trivial implementation of a model capable of dynamically adjusting its hierarchical classification during run-time, we present AG-HBN. It is important to note that the aim of this model is not to serve as an optimized solution but rather as a proof-of-concept, demonstrating the feasibility of such a dynamic classification approach.

AG-HBN incorporates an additional exit at the first branch, used to discern whether a fine or coarse classification should be pursued based on the input image (see figure 2).

A. Training

The training procedure for the model is initially equivalent to that of the Gran-HBN model. The changes in training occur after the model has been trained on the branch loss from equation (2). At this point the model is "post-trained" to optimize the semantic loss, L_{sem}.

B. Post-Training

When post-training, all model weights are frozen excluding the additional semantic exit at the first branch. The output of the semantic exit, $\hat{\mathbf{y}}_{\text{sem}}$, is a classification for either a fine or coarse exit: $\{\text{coarse, fine}\} = \{0, 1\}$.

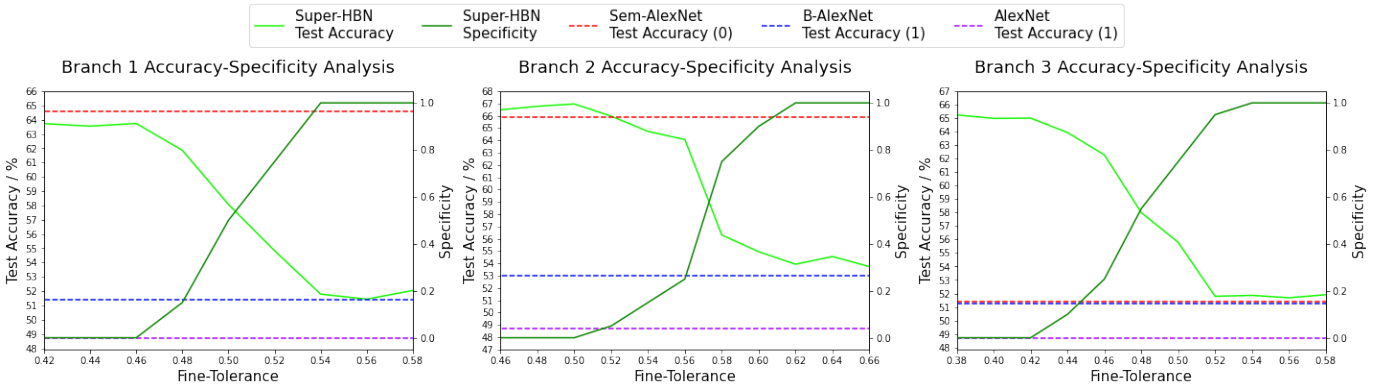

Fig. 3. The effect of varying the fine-tolerance hyper-parameter on the test-accuracy and specificity of each branch. The graph is the result of forcing the models-under-test to exit all test samples at branch 1, 2 or 3. We notice that the fine-tolerance has an operating region, where the variations in specificity occur. Outside of the fine-tolerance region, the classification specificity is fixed at either 0 or 1.

The objective during post-training is to minimize the semantic loss function L_{sem}. The semantic loss function constitutes of two primary components: the estimated optimal granularity, $\overline{\mathbf{y}}_{\text{sem}}$, and the granular entropy, H_{gran}.

To derive $\overline{\mathbf{y}}_{\text{sem}}$, we define an indicator function in equation (8), regarding whether a correct classification was made for a given branch n, batch image i, classification granularity g, and fine class weighting w_f. The indicator function is subsequently used to get the estimated optimal granularity.

$$\begin{aligned} \mathbb{1}_{n,i}^{(g)} &= \begin{cases} 1 & \text{if } \hat{y}_{n,i}^{(g)} = y_i^{(g)} \\ 0 & \text{otherwise} \end{cases} \\ \overline{\mathbf{y}}_{\text{sem},i} &= \begin{cases} 1 & \text{if } \sum_n \mathbb{1}_{n,i}^{(g_f)} \cdot w_f > \sum_n \mathbb{1}_{n,i}^{(g_c)} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{8}$$

To obtain H_{gran}, we first calculate the entropy at each classification exit using equation 9, for a given semantic classification granularity g and class c.

$$\begin{aligned} \sigma_{ic}^{(g)} &= \frac{e^{z_{ic}^{(g)}}}{\sum_k e^{z_{ik}^{(g)}}} \\ H_i^{(g)} &= -\sum_{c \in C} \sigma_{ic}^{(g)} \log(\sigma_{ic}^{(g)}) \end{aligned} \tag{9}$$

Subsequently, we can sum the average entropies $\overline{H}^{(g_f)}$ and $\overline{H}^{(g_c)}$ over the batch, applying a weighting, w_f, to the fine entropies based on the preference for fine classifications, as shown in 10.

$$H_{\text{gran}} = \frac{w_f}{N} \sum_{i=1}^{N} H_i^{(g_f)} + \frac{1}{N} \sum_{i=1}^{N} H_i^{(g_c)} \tag{10}$$

The semantic loss, L_{sem}, is thus the weighted sum of the binary cross-entropy, $\text{BCE}(\overline{y}_{\text{sem}}, \hat{y}_{\text{sem}})$, and the granular entropy H_{gran}.

$$L_{\text{sem}}(\overline{\mathbf{y}}_{\text{sem}}, \hat{\mathbf{y}}_{\text{sem}}; \theta_{\text{sem}}) = \alpha \cdot \text{BCE}(\overline{\mathbf{y}}_{\text{sem}}, \hat{\mathbf{y}}_{\text{sem}}) + H_{\text{gran}} \tag{11}$$

Intuitively, the first term of the loss function is used to identify the optimal specificity for classification, whereas the second term is used to increase the confidence in the classifications.

The parallels between bootstrapping in reinforcement learning (RL) and the post-training method outlined above are apparent. The estimated optimal classification granularity, $\overline{\mathbf{y}}_{\text{sem}}$, is an estimate of the optimal classification specificity obtained after the forward pass. This estimate is subsequently used to update the current optimal classification granularity, $\hat{\mathbf{y}}_{\text{sem}}$, of the model.

C. Inference

Similar to Gran-HBN, the computational cost is reduced by utilizing early-exiting branches with confidence-based criteria.

In contrast, AG-HBN employs a binary semantic exit classification, $\hat{\mathbf{y}}_{\text{sem}} = \{\text{coarse, fine}\} = \{0, 1\}$, enabling it to dynamically choose between a fine or coarse exit at any branch. This removes the necessity for the fine-tolerance hyperparameter. To accomodate the modification, additional conditions are required. In lines 6 and 13, the fine condition outlined in 12 is added, while in line 8, the coarse condition from 12 is included.

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^{N} \hat{y}_{\text{sem},i} &\geq 0.5 \qquad \text{fine condition} \\ \frac{1}{N} \sum_{i=1}^{N} \hat{y}_{\text{sem},i} &< 0.5 \qquad \text{coarse condition} \end{aligned} \tag{12}$$

V. Experimentation

The Gran-HBN and AG-HBN models were trained and evaluated on the CIFAR-100 data set over a two-level semantic class hierarchy [14]. To provide a comparative analysis, prior models from [1]–[3], [5] were also trained on the CIFAR-100 data set. To ensure a fair comparative analysis, each model's architecture is constructed upon the foundations laid by AlexNet and B-AlexNet, with subsequent models building

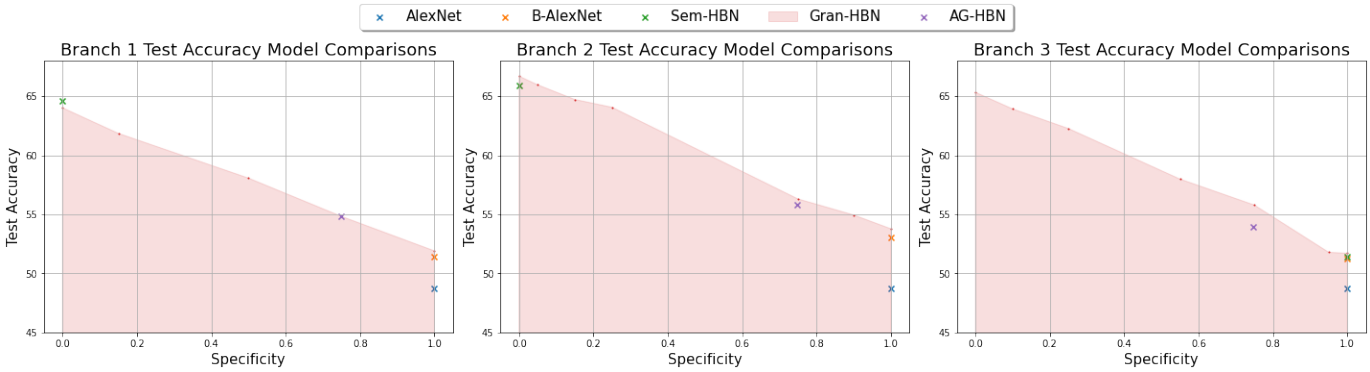

Fig. 4. Comparative analysis of the performance of Gran-HBN and AG-HBN in relation to the benchmark models [1]–[3], [5] at each individual branch.

upon them. Mini-batch gradient descent with an Adam optimizer is used during training with Kaiming initialization [22] of each models parameters. The metadata consisting of the hyper-parameter values used during training is given in table I.

	AlexNet	B-AlexNet/Sem-AlexNet	Gran-HBN	AG-HBN
η	1.5×10^{-4}	1.5×10^{-4}	1.5×10^{-4}	1.5×10^{-4}
λ	5×10^{-4}	5×10^{-4}	5×10^{-4}	5×10^{-4}
w_n	N/A	[1.0, 0.8, 0.6]	[1.0, 0.8, 0.6]	[1.0, 0.8, 0.6]
w_f	N/A	N/A	1.5	1.02

TABLE I
MODEL TRAINING METADATA: HERE, η IS THE LEARNING RATE, λ IS THE WEIGHT DECAY, w_n IS THE WEIGHTED BRANCH LOSS, AND w_f IS THE FINE ENTROPY WEIGHTING. B-ALEXNET IS FROM [5], AND SEM-ALEXNET IS THE SEMANTIC HIERARCHICAL B-ALEXNET FROM [2], [3].

Each model is trained with a batch size of 64 for 120 epochs (in the case of AG-HBN, it has 100 epochs of training and 20 epochs of post-training). It should be noted that model testing is conducted with a batch size of 512, with the exception of AG-HBN, which undergoes testing with a batch size of 1.

The model was tested on an x86 i7-8565U CPU at 1.8Ghz, fixed at 70% of its maximum clock speed (1.26Ghz). This was done to ensure a fixed processing rate, mitigating hotspot formation in the CPU and producing a stable environment for accurate model latency measurements to be obtained.

VI. ANALYSIS

During inference, the threshold and fine-tolerance hyper-parameters of the Gran-HBN model are used to modulate the computational load and classification specificity, respectively. We define the classification specificity, δ, over a batch of images in equation 13, with $|\hat{y}_i^{(g_f)}|$ representing the number of fine classifications made in batch i.

$$\delta = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i^{(g_f)}| \tag{13}$$

The effects of the threshold hyper-parameter have already been investigated in prior research [5], therefore, section VI-A will be used to investigate the effects of the fine-tolerance hyper-parameter on the accuracy/specificity trade-off.

Following this analysis, section VI-B presents a comparison of the Gran-HBN and AG-HBN model performance against benchmark models [1]–[3], [5].

Finally, in section VI-C, we conduct an investigation of granular classification entropies at each exit of Gran-HBN, with the aim to provide novel insights into the dynamics of hierarchical learning.

A. Accuracy/Specificity Trade-Off

Figure 3 shows the effects of varying the Gran-HBN fine-tolerance hyperparameter, τ, at each branch. When observing the plots, we identify an operating region for each branch, noted in table II.

Branch	Operating Region	
	Range	Size
1	0.46 to 0.54	0.08
2	0.50 to 0.62	0.12
3	0.42 to 0.54	0.12

TABLE II
THE OPERATING REGIONS FOR EACH BRANCH OF THE GRAN-HBN MODEL

Within the operating region, there is a trade-off between test accuracy and specificity. The optimal value of the fine-tolerance is situational, and as such, the fine-tolerance can be varied within the operating region to acquire the desired trade-off during inference.

To get an intuitive understanding of figure 3, we must understand the intuition behind the x-axis. The fine-tolerance serves as a measure of the entropy difference between fine and coarse classifications. The size of the operating region, thus acts as a gauge for the spread of hierarchical classification uncertainty across the testset. In this context, hierarchical classification variance refers to the degree of additional uncertainty associated with a fine classification compared to a coarse classification for a single image.

If the operating region spans over a large range of fine-tolerances, it implies that the branch exhibits large variations in hierarchical classification variance over the testset and vice

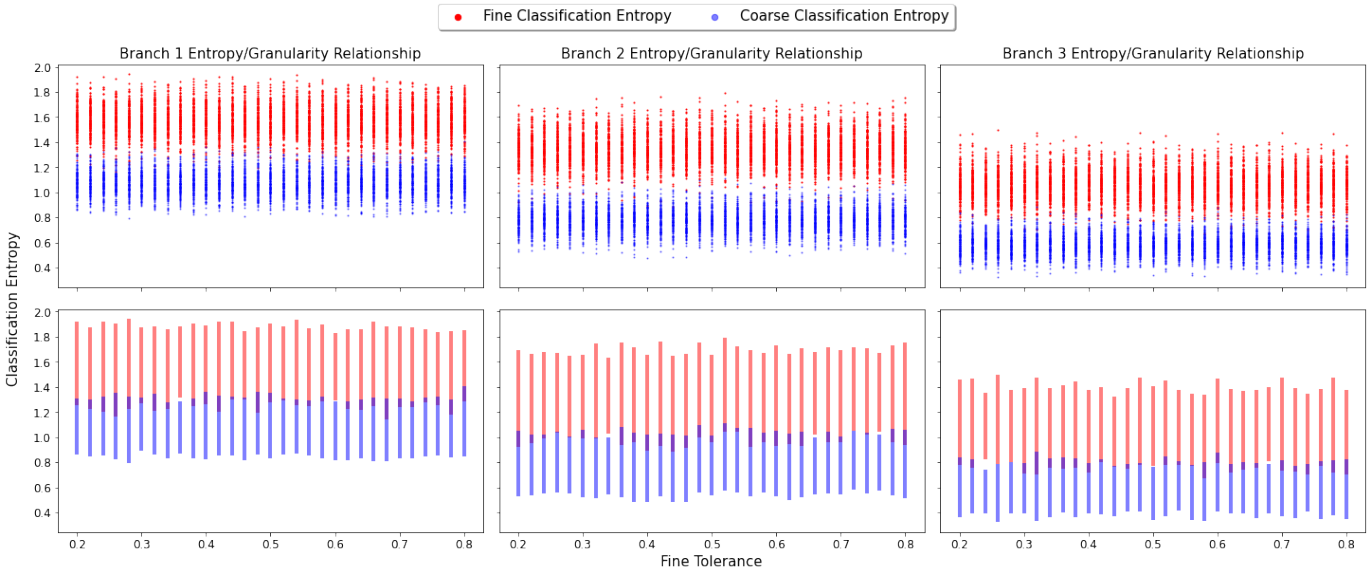

Fig. 5. Empirical findings regarding fine and coarse classification entropies across branches, facilitating the assessment of the distribution and range of classification uncertainties associated to distinct classification granularities within each branch. The top row showcases scatter plots for each branch, indicating the individual classification entropy results. The bottom row depicts the variations in classification entropy for each semantic heirarchy.

versa. Thus, the blueprint for an optimal branch involves a narrow operating region that begins and ends at a low fine-tolerance value, indicating a small hierarchical classification variance and a marginal increase in uncertainty for increased specificity.

Now that the blueprint for an optimal branch is established, we can investigate the effects of the different branches. Table II shows that branch 1 has the smallest operating region, indicating reduced hierarchical classification variance over the testset. Conversely, branch 3's operating region is initiated at the earliest fine-tolerance value, despite it having the joint largest operating region with branch 2. This indicates that the degree of hierarchical classification variance increases with feature extraction complexity, resulting in some fine label classifications being made with greater relative confidence to their coarse label classification counterpart.

These observations showcase the importance of incorporating classification flexibility in, particularly late, branches of DyNN early-exiting models. Such flexibility allows for navigation of the operating region, resulting in occasional confident fine label classifications to be made while still promoting more frequent, confident coarse label classifications.

It should be noted that the size of the operating region increases as the batch size of the test set is reduced. This is likely due to the model's heightened sensitivity to the individual complexities of each image within the smaller batch. Thus, the large batch size was used during testing to provide a more generalized perspective of the models behaviour.

B. Model Performance Comparison

Figure 4 provides a comparative analysis of each models performance at each branch. The benefits of Gran-HBN are evident in these plots, due to its classification flexibility allowing each branch to navigate the accuracy-specificity landscape for optimal trade-offs at run-time. AG-HBN automates this process, albeit with a noted loss in accuracy compared to the Gran-HBN model. This discrepancy can be mitigated by performing the same number of training epochs as the other models and then incorporating the additional post-training epochs. The authors chose not to pursue this adjustment, deeming it an unfair comparison.

In contrast to benchmark models, Gran-HBN consistently matches or surpasses their performance at extreme specificities, offering the additional advantage of classification flexibility at each branch.

C. Granular Classification Uncertainty

Figure 5 allows for a more detailed analysis of the hierarchical classification entropies and also provides compelling empirical evidence for the necessity of incorporating a diverse array of classification hierarchies within each branch.

The primary characteristic is a reduction in both fine and coarse classification entropies at each subsequent branch. This indicates a direct correlation between classification uncertainty and the complexity of feature extraction, leading to increased classification confidence in later branches, aligning with the findings from prior works [2]–[5].

Notably, an overlap in fine and coarse classification uncertainty is evident at each branch, signifying a presence of fine label classifications with coarse classification confidence. To optimize performance, a range of hierarchical classifications should be made for each batch. Coarse label classifications can be leveraged to achieve accuracies characteristic of later branches and more advanced models. Simultaneously, fine label classifications can be leveraged with comparable coarse

classification confidence, thus enhancing branch classification specificity while preserving classification confidence.

VII. Conclusion and Future Work

This paper presents two models: Gran-HBN, designed for adaptable classifications in granularity during inference, and AG-HBN, a trivial proof-of-concept demonstrating the feasibility of a model autonomously adjusting classification granularity during inference.

Gran-HBN facilitates an analysis of the accuracy-specificity trade-off and explores the necessity of hierarchical classifications during inference. The obvserved overlap in fine and coarse classification uncertainty at each branch suggests the existence of fine label classifications with coarse classification confidence. Thus, optimal performance involves employing a range of hierarchical classifications at each branch, leveraging both coarse and fine label classifications to enhance specificity while preserving accuracy and having negligible impact on latency.

Furthurmore, Gran-HBN consistently matches or surpasses benchmark model performance at the specifity extremities, while offering additional benefits such as increased classification flexibility at each branch, increased early branch information density, and increased early branch granular complexity. AG-HBN demonstrates this capability without the need for a fine-tolerance hyperparameter, however, achieves this at the cost of an additional post-training phase and larger model.

Future work should increase the scalability of the Gran-HBN architecture, enabling the incorporation of more classification hierarchies without the necessity of additional exits. Furthermore, efforts should be directed towards more sophisticated implementations of AG-HBN, eliminating the reliance on post-training and an additional semantic exit. The authors believe that adopting a RL approach towards the training of this model represents the optimal strategy for addressing these challenges.

Acknowledgment

The authors acknowledge the use of the IRIDIS High-Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] X. Zhu and M. Bain, "B-CNN: branch convolutional neural network for hierarchical classification," *arXiv preprint arXiv:1709.09890*, 2017.

[3] B. Kolisnik, I. Hogan, and F. Zulkernine, "Condition-CNN: A hierarchical multi-label fashion image classification model," *Expert Systems with Applications*, vol. 182, p. 115195, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421006291

[4] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

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

[6] U. Acar, A. Ihler, R. Mettu, and O. Sumer, "Adaptive Inference on General Graphical Models," *arXiv e-prints*, 06 2012.

[7] G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," *arXiv preprint arXiv:1703.09844*, 2017.

[8] T. Bolukbasi, J. Wang, O. Dekel, and V. Saligrama, "Adaptive neural networks for efficient inference," in *International Conference on Machine Learning*. PMLR, 2017, pp. 527–536.

[9] 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)*, 2021, pp. 5071–5080.

[10] L. Bertinetto, R. Mueller, K. Tertikas, S. Samangooei, and N. A. Lord, "Making better mistakes: Leveraging class hierarchies with deep networks," *arXiv e-prints*, pp. 12 506–12 515, 2020.

[11] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[12] R. Cerri, R. C. Barros, and A. C. de Carvalho, "Hierarchical multi-label classification using local neural networks," *Journal of Computer and System Sciences*, vol. 80, no. 1, pp. 39–56, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000013000718

[13] 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 the IEEE international conference on computer vision*, 2015, pp. 2740–2748.

[14] A. Krizhevsky, "Learning multiple layers of features from tiny images," MIT, Tech. Rep., 2009.

[15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[16] G. A. Miller, *WordNet: An electronic lexical database*. MIT press, 1998.

[17] L. He, D. Song, and L. Zheng, "Hierarchical Image Classification with A Literally Toy Dataset," *arXiv preprint arXiv:2111.00892*, 2021.

[18] R. La Grassa, I. Gallo, and N. Landro, "Learn class hierarchy using convolutional neural networks," *Applied Intelligence*, vol. 51, no. 10, pp. 6622–6632, 2021.

[19] J. Wehrmann, R. Cerri, and R. Barros, "Hierarchical Multi-Label Classification Networks," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 5075–5084.

[20] P. Goyal and S. Ghosh, "Hierarchical Class-Based Curriculum Loss," *arXiv preprint arXiv:2006.03629*, 2020.

[21] C. Silla and A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, pp. 31–72, 01 2011.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.