Early Exiting in Deep Neural Networks via Dirichlet-based Uncertainty Quantification

Feng Xia
Princeton University
feng.xia@princeton.edu

Jake C. Snell
Princeton University
jsnell@princeton.edu

Thomas L. Griffiths
Princeton University
tomg@princeton.edu

Abstract

Deep neural networks are renowned for their accuracy across a spectrum of machine learning tasks but often suffer from prolonged inference time due to their depth. Early exiting strategies have been proposed to mitigate this by allowing predictions to output at intermediate layers if total uncertainty falls below a threshold. However, we observe that using total uncertainty as an exiting criterion does not consistently reflect true model uncertainty, causing traditional methods to prevent early exits for ambiguous data even when model uncertainty is low. To address this limitation, we propose a Dirichlet-based framework to directly quantify model uncertainty. Models trained with our approach demonstrate more balanced handling of both ambiguous and unambiguous data, enabling a higher proportion of ambiguous samples to exit early for more efficient inference.

1 Introduction

Deep neural networks (DNNs) have shown tremendous success across various domains: convolutional neural networks for image classification [14, 12, 25, 28, 6], transformers for language modeling [30, 4, 3], recurrent neural networks for sequential data processing [21, 5, 26], just to name a few. While this remarkable success is primarily credited to their depth, it also leads to substantial computational costs in both training and inference phases [27]. Fast inference in DNNs is important for various reasons. First, as networks continue to grow deeper, optimizing computation becomes crucial to minimize waste and enhance efficiency. Second, fast inference is essential for deploying models in resource-constrained environments such as mobile devices and embedded systems. Third, real-world applications require fast inference to deliver a seamless user experience by enabling real-time responses.

The sequential nature of DNNs limits their potential for parallelization [32]. To address this, early exiting is an approach aimed at reducing the average inference time without parallelization [13]. It allows certain samples to exit early through intermediate layers of the network when the network exhibits sufficient confidence. Traditional exiting criteria use low total uncertainty as a proxy for high model confidence [29]. Since total uncertainty can be further decomposed into model uncertainty and data uncertainty, this measure does not accurately capture true model uncertainty, and therefore fails to allow samples with high data uncertainty to exit early when model uncertainty is low.

We suggest that it is crucial to distinguish model uncertainty from total uncertainty and utilize the former as the exiting criterion instead. To this end, we propose a Dirichlet-based framework that posits neural networks output a Dirichlet distribution of probability vectors instead of a single

38th Workshop on Fine-Tuning in Machine Learning (NeurIPS 2024).

probability vector. Under this assumption, model uncertainty can be captured by the spread of the distribution. We observe that the traditional cross entropy loss is degenerate within this framework and therefore introduce a new training objective based on Kullback-Leibler divergence. Furthermore, we generate an artificial dataset containing samples with high data uncertainty to explicitly train the model for better handling of such cases.

Our findings show that models trained using our approach are more balanced in handling both high-complexity and low-complexity samples, while the early exiting baseline shows a significant performance gap between these two types of data. As a result, our approach enables a larger proportion of ambiguous samples to exit early, outperforming the baseline. Our approach provides a simple and easy-to-implement improvement to exiting criteria that preserves the model architectures and enables models to perform effectively not only on low-complexity but also high-complexity data. By differentiating between model uncertainty and total uncertainty, we also expose an interpretable representation of uncertainty, which may be inherently useful for applications where uncertainty is important for human decision making.

2 Background

2.1 Early exiting in deep neural networks

Early exiting is an approach aimed at reducing the average inference time of deep neural networks by allowing "easy" samples to terminate early at intermediate exits, while "difficult" samples can utilize the networks fully [20]. It usually requires answers to three main questions: 1) where to place the intermediate exits (architectural design), 2) how to train the intermediate exits (training objective), and 3) how to decide when to terminate at an intermediate exit (exiting criterion). Our study focuses on the third question.

2.2 BranchyNet

BranchyNet [29] is one of the pioneering and most frequently referenced architectures for early exiting in deep neural networks, making it a robust baseline for our studies.

Architecture. BranchyNet modifies the standard neural network by adding exit branches throughout the network, allowing samples to exit early when their predictions have high confidence. The exit branches can range from simple fully connected layers to more complex configurations such as convolutional layers and multilayer perceptrons.

Training. During training, all branches in BranchyNet are jointly trained with the main network using softmax cross entropy loss. Let $\boldsymbol{y}$ be the one-hot encoded ground truth label vector, and $\hat{\boldsymbol{y}}$ be the predicted softmax probability vector. Then the softmax cross entropy loss is defined as:

$$L(\hat{\boldsymbol{y}}, \boldsymbol{y}) = -\sum_{c \in C} y_c \log \hat{y}_c \tag{1}$$

where C is the set of all possible labels. The training objective of BranchyNet is to minimize the weighted sum of losses at each exit branch:

$$L_{\text{BranchyNet}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \sum_{n=1}^{N} w_n L(\hat{\boldsymbol{y}}_{\text{exit}_n}, \boldsymbol{y}) \tag{2}$$

where N is the total number of branches and w_n are hyperparameters that represent weights of each branch. Earlier branches receive higher weights in BranchyNet.

Exiting criterion. During inference, a sample begins with the lowest-level exit and iterates to the final exit of the network. In BranchyNet, the exit criterion is based on the entropy. The entropy for a predicted probability vector $\hat{\boldsymbol{y}}$ is defined as:

$$\mathcal{H}(\hat{\boldsymbol{y}}) = -\sum_{c \in C} \hat{y}_c \log \hat{y}_c \tag{3}$$

Once the entropy value of a predicted probability vector at an intermediate exit falls below a predetermined threshold T, the prediction is returned and the sample does not undergo further processing by subsequent layers.

2.3 Aleatoric uncertainty and epistemic uncertainty

In machine learning, total uncertainty can be broken down into two dimensions: aleatoric uncertainty and epistemic uncertainty [10, 15, 7]. Aleatoric uncertainty, also known as data uncertainty, usually refers to the irreducible uncertainty which arises from the natural complexity of data, such as class overlap and label noise. Epistemic uncertainty, also known as model uncertainty, on the other hand, is the uncertainty caused by a lack of training data or insufficient model complexity. In particular, let x be the input to the network and μ be a bottleneck representation such that $p(y|x) = \int p(y|\mu)p(\mu|x)\, d\mu$. Then the total uncertainty can be decomposed as:

$$\mathcal{H}(y|x) = \underbrace{\mathbb{E}_{\mu\sim p(\mu|x)}\left[\mathcal{H}(y|\mu)\right]}_{\text{aleatoric (data) uncertainty}} + \underbrace{\mathcal{I}(y;\mu|x)}_{\text{epistemic (model) uncertainty}} \tag{4}$$

Many early exiting models rely on total uncertainty (i.e., entropy) as the exiting criterion. The underlying assumption of these models is that the predictions of a neural network should converge to a probability vector that favors a single class through successive layers. Therefore, they implicitly assume total uncertainty will monotonically decrease towards zero throughout the network. Based on this premise, high model confidence can be approximated by low entropy. Such an assumption is generally accurate for samples with low data uncertainty (e.g., an unambiguous image as shown in Figure 1). However, for samples with high data uncertainty (e.g., an ambiguous image as shown in Figure 2), a network's predictions may converge to a probability vector that assigns comparable probabilities to multiple classes. Thus, early exiting mechanisms based on total uncertainty may consistently route ambiguous samples to the final exit due to their overall high uncertainty, even when the network's predictions have stabilized at earlier stages.

Figure 1: An unambiguous rabbit

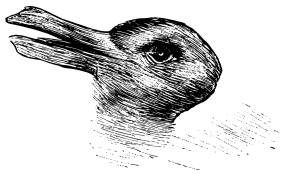


Figure 2: An ambiguous rabbit-duck illusion [31]

Table 1 summarizes four possible combinations of data uncertainty and model uncertainty, with examples for each. Among these four cases, only case 1 has low total uncertainty, allowing early exits for samples from this category when using total uncertainty measure. Since data uncertainty is irreducible, we argue that our objective is to return an intermediate prediction as soon as the model uncertainty becomes low, regardless of data uncertainty. Such an approach will enable samples from both case 1 and case 2 to exit early in the network.

Table 1: Four possible combinations of data uncertainty and model uncertainty.

	Model Uncertainty	Data Uncertainty	Total Uncertainty	Examples
Case 1	low	low	low	unambiguous in-distribution data
Case 2	low	high	high	ambiguous in-distribution data
Case 3	high	low	high	unambiguous out-of-distribution data
Case 4	high	high	high	ambiguous out-of-distribution data

3 Approach

3.1 Basics of the Dirichlet distribution

To effectively distinguish model uncertainty from total uncertainty, the assumption that neural networks produce a single point estimation is inadequate. Instead, we need networks' output to be

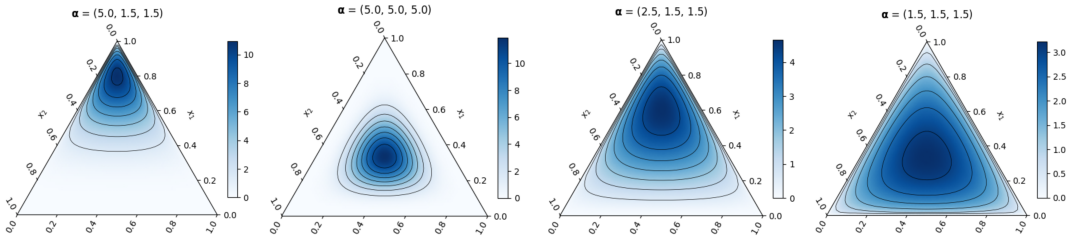
able to capture uncertainty in its prediction. The Dirichlet distribution provides a natural solution, as it is a distribution over predicted probability vectors.

The Dirichlet distribution of probability vectors over the set of all possible classes C is parameterized by $|C|$ concentration parameters $\alpha_1, \alpha_2, \ldots, \alpha_{|C|} > 0$, and is defined as:

$$\text{Dir}(\hat{\boldsymbol{y}} \mid \boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\prod_{c \in C} \Gamma(\alpha_c)} \prod_{c \in C} \hat{y}_c^{\alpha_c - 1}, \qquad \alpha_0 = \sum_{c \in C} \alpha_c \tag{5}$$

where $\hat{\boldsymbol{y}}$ represents a probability vector over C and $\Gamma(\cdot)$ is the gamma function.

The concentration parameters $\alpha_1, \alpha_2, \ldots, \alpha_{|C|}$ control the distribution of probability vectors: larger values of α_c indicate that probability vectors drawn from this distribution will assign a higher probability to class c. When all α_c are large (Figure 3b), the distribution concentrates around the uniform probability vector at the center of the simplex. Conversely, with smaller α_c values (Figure 3d), the distribution over probability vectors becomes more dispersed.



(a) Low model uncertainty, low data uncertainty (b) Low model uncertainty, high data uncertainty (c) High model uncertainty, low data uncertainty (d) High model uncertainty, high data uncertainty

Figure 3: Four Dirichlet simplexes under different combinations of model and data uncertainty. Visualizations were created using the `mpltern` software package [8].

By assuming that neural networks output a Dirichlet distribution of class probability vectors rather than a single class probability vector, the spread of the Dirichlet distribution can effectively represent model uncertainty. To visualize, Figure 3 presents four possible scenarios of such a Dirichlet simplex over three classes. The spread of the Dirichlet distribution can be quantified by its differential entropy:

$$\mathcal{H}\left[\text{Dir}(\boldsymbol{\alpha})\right] = \log \frac{\prod_{c \in C} \Gamma(\alpha_c)}{\Gamma(\alpha_0)} + (\alpha_0 - |C|)\psi(\alpha_0) - \sum_{c \in C} (\alpha_c - 1)\psi(\alpha_c) \tag{6}$$

where $\psi(\cdot)$ is the digamma function. A higher differential entropy indicates a more dispersed distribution, reflecting greater model uncertainty in the predicted class probabilities.

3.2 Our Dirichlet-based framework

We will next show the validity of the assumption that neural networks output a Dirichlet distribution over class probability vectors. Specifically, we will show that the network's output before the softmax function captures a Dirichlet distribution. Mathematically, given an input x and a set of training data $\mathcal{D}$, the predicted probability of any class c can be written as:

$$\hat{y}_c = P(\hat{y} = c \mid x, \mathcal{D}) = \int p(\hat{y} = c \mid \boldsymbol{\alpha}) p(\boldsymbol{\alpha} \mid x, \mathcal{D}) \, d\boldsymbol{\alpha} \tag{7}$$

where $\hat{y}$ is the predicted class and $\boldsymbol{\alpha}$ denotes the set of parameters of the Dirichlet distribution that the network outputs. By further assuming that the trained network's weights $\boldsymbol{w}$ sufficiently captures the entire training dataset $\mathcal{D}$ and that the network outputs a deterministic Dirichlet distribution parameterized by $\boldsymbol{\alpha}^\star$ (i.e. $\boldsymbol{\alpha}^\star = f(x, \boldsymbol{w})$, where f is the function that the network computes), the

predicted probability then simplifies to:

$$
\begin{aligned}
P(\hat{y} = c \mid x, \mathcal{D}) &= \int p(\hat{y} = c \mid \boldsymbol{\alpha}) p(\boldsymbol{\alpha} \mid x, \boldsymbol{w}) \, d\boldsymbol{\alpha} \\
&= \int p(\hat{y} = c \mid \hat{\boldsymbol{y}}) p(\hat{\boldsymbol{y}} \mid \boldsymbol{\alpha}^{\star}) \, d\hat{\boldsymbol{y}} \\
&= \int \hat{y}_c \cdot \text{Dir}(\hat{\boldsymbol{y}} \mid \boldsymbol{\alpha}^{\star}) \, d\hat{\boldsymbol{y}} \\
&= \mathbb{E}_{\text{Dir}(\hat{\boldsymbol{y}} \mid \boldsymbol{\alpha}^{\star})}[\hat{y}_c] \\
&= \frac{\alpha_c^{\star}}{\sum_{k \in C} \alpha_k^{\star}}
\end{aligned}
\tag{8}
$$

where the last equation comes from the definition of expectation of a Dirichlet distribution. We recall that this predicted probability is also given by the network's softmax output:

$$
P(\hat{y} = c \mid x, \mathcal{D}) = \frac{e^{z_c(x)}}{\sum_{k \in C} e^{z_k(x)}} \tag{9}
$$

where $z_k(x), \forall k \in C$ are the network's outputs before softmax. By equating the two, we have:

$$
\frac{\alpha_c^{\star}}{\sum_{k \in C} \alpha_k^{\star}} = \frac{e^{z_c(x)}}{\sum_{k \in C} e^{z_k(x)}} \tag{10}
$$

Under this formulation, the neural network is essentially computing the parameters of the Dirichlet distribution. More precisely, it computes $\log(\boldsymbol{\alpha})$.

Because both narrowly and widely scattered Dirichlet distributions can yield the same expected values, our Dirichlet framework allows two network outputs that result in identical probability vectors after softmax to differ in model uncertainty, even when their total uncertainty remains the same.

3.3 Training

Training a neural network within the Dirichlet framework using the standard softmax cross entropy loss is degenerate because it only encourages the expected value of the output Dirichlet to become closer to the ground truth probability vector. It is insensitive to arbitrary scaling of $\boldsymbol{\alpha}$ which controls the spread of the Dirichlet, and it therefore disregards model uncertainty in its prediction. In order to train the network to yield a confident prediction around the target probability vector, it is necessary to modify the loss function.

Let $\mathcal{D}_{\text{unamb}}$ represent the set of unambiguous training data, where each sample is associated with exactly one class. Drawing inspiration from Dirichlet Prior Networks for out-of-distribution sample detection [15, 16], we define a new loss based on Kullback-Leibler divergence:

$$
L_{\text{unamb}}(\hat{\boldsymbol{\alpha}}) = \frac{1}{|\mathcal{D}_{\text{unamb}}|} \sum_{\mathcal{D}_{\text{unamb}}} \text{KL}\left[\text{Dir}(\hat{\boldsymbol{y}} | \hat{\boldsymbol{\alpha}}) \,||\, \text{Dir}(\hat{\boldsymbol{y}} | \boldsymbol{\alpha}_{\text{unamb}})\right] \tag{11}
$$

where $\boldsymbol{\alpha}_{\text{unamb}}$ is the set of parameters of the target Dirichlet for an unambiguous sample, which we define as:

$$
\boldsymbol{\alpha}_{\text{unamb}}^{(c)} = \begin{cases} \beta + 1 & \text{if } c = k \\ 1 & \text{otherwise} \end{cases}, \quad \text{for } c \in C \tag{12}
$$

where C is the set of all possible classes and k is the ground truth class of the sample. This loss encourages the network to output a sharp Dirichlet distribution centered around the ground truth label's probability vector. The parameter β controls the sharpness—larger values correspond to greater sharpness.

As we wish the network to not only perform well on unambiguous data but also on ambiguous data, we include a set of ambiguous data during training. The goal remains consistent, but now the target vector will represent class overlaps, assigning probabilities to multiple classes. For example, if an ambiguous sample may equally belong to any class in the set $K \subseteq C$, the target Dirichlet for this sample is:

$$
\boldsymbol{\alpha}_{\text{amb}}^{(c)} = \begin{cases} \beta + 1 & \text{if } c \in K \\ 1 & \text{otherwise} \end{cases}, \quad \text{for } c \in C \tag{13}
$$

The training objective is to learn the target Dirichlet for each ambiguous sample:

$$L_{\text{amb}}(\hat{\boldsymbol{\alpha}}) = \frac{1}{|\mathcal{D}_{\text{amb}}|} \sum_{\mathcal{D}_{\text{amb}}} \text{KL}\left[\text{Dir}(\hat{\boldsymbol{y}} \mid \hat{\boldsymbol{\alpha}}) \,||\, \text{Dir}(\hat{\boldsymbol{y}} \mid \boldsymbol{\alpha}_{\text{amb}})\right] \tag{14}$$

The final loss is the sum of the two losses:

$$L(\hat{\boldsymbol{\alpha}}) = L_{\text{unamb}}(\hat{\boldsymbol{\alpha}}) + L_{\text{amb}}(\hat{\boldsymbol{\alpha}}) \tag{15}$$

3.4 Exiting criterion

We use differential entropy of the Dirichlet distribution as a measure of model uncertainty to replace traditional entropy which measures total uncertainty to determine whether to exit at an intermediate branch. As in BranchyNet, exiting thresholds can be manually set to meet a specified runtime or accuracy constraint.

4 Experiments

4.1 Datasets

CIFAR-10. CIFAR-10 is a widely used dataset in machine learning for image classification benchmarking [11]. It contains 60,000 color images of size 32x32 from 10 classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It is divided into 50,000 training images and 10,000 testing images.

Unambiguous dataset. We use the original CIFAR-10 training dataset as the unambiguous training dataset as each image in the set corresponds exactly to one class.

Ambiguous dataset. Given the challenge of obtaining a large set of naturally ambiguous images, we generate ambiguous images artificially by blending images from distinct classes in the CIFAR-10 training dataset. The blending process is described in Appendix C. A blended image represents the overlaps of the classes from which it is derived.

We generated 50,000 blended images to match the size of CIFAR-10 training dataset. Images were blended from six groups of common overlapping classes: 10,000 images of "cat vs. dog", 10,000 images of "deer vs. horse", 10,000 images of "bird vs. airplane", 10,000 images of "automobile vs. truck", 5,000 images of animals (bird, cat, deer, dog, frog, horse), and 5,000 images of vehicles (airplane, automobile, ship, truck).

4.2 Models

In our experiments, we compare several models. Our goal is to explore the effects of (a) using the Dirichlet distribution to capture uncertainty, and (b) the effect of training with ambiguous data.

AlexNet. AlexNet is a convolutional neural network used widely in image classification [12]. It is consisted of five convolutional layers followed by three fully connected layers. To align with the early exiting architecture we used in our experiments, we modified the original AlexNet to only include one fully connected layer following five convolutional layers (see Figure 6 in Appendix A).

B-AlexNet. B-AlexNet is a BranchyNet adapted from AlexNet by adding a fully connected layer as a side branch to each of the five convolutional layers [29] (see Figure 7 in Appendix A). B-AlexNet was trained on the unambiguous dataset according to Section 2.2. For simplicity, we set training weights w_n to be 1 for all branches. B-AlexNet served as the baseline model in our experiments.

Dir-B-AlexNet. Dir-B-AlexNet shares the same architecture as B-AlexNet, but it was trained using the objective rooted in the Dirichlet framework. Specifically, it was trained solely on the unambiguous dataset and utilized the loss function outlined in Equation 11 in Section 3.3.

Amb-Dir-B-AlexNet. Similar to Dir-B-AlexNet, Amb-Dir-B-AlexNet has the same architecture as B-AlexNet and was trained under the Dirichlet objective. However, it was trained on both the unambiguous dataset and the ambiguous dataset, utilizing the summed loss function outlined in Equation 15 in Section 3.3.

5 Results

We analyzed the performance of our Dirichlet-based approach (Dir-B-AlexNet and Amb-Dir-B-AlexNet) compared to the traditional early exiting baseline (B-AlexNet) in handling ambiguous data versus unambiguous data.

5.1 Evaluation datasets

In order to better understand how the various models handle different types of input samples, we measure their performance on four evaluation datasets.

Baseline unambiguous dataset. The original CIFAR-10 testing dataset served as the baseline unambiguous dataset in our evaluation.

Ambiguous dataset: Challenging Test. Challenging Test consists of 866 images in CIFAR-10 testing dataset found challenging by AlexNet during inference, indicated by entropy values over 0.5. We further confirmed its ambiguity by observing that all branches in all three networks achieved an accuracy around 50%.

Ambiguous dataset: In-distribution Blend. In-distribution Blend contains 10,000 images blended from CIFAR-10 testing dataset, using the same blending distribution as the ambiguous training dataset for training Amb-Dir-B-AlexNet.

Ambiguous dataset: Out-of-distribution Blend. Out-of-distribution Blend contains 10,000 images blended from CIFAR-10 testing dataset, using different combinations of categories relative to the ambiguous training dataset. Specifically, it includes 2,500 images of "bird vs. frog", 2,500 images of "ship vs. truck", 2,500 images of "automobile vs. airplane" and 2,500 images of "deer vs. dog".

5.2 Evaluation Metrics

We evaluated the methods using two key metrics: (a) difference in average uncertainty between ambiguous and unambiguous data and (b) the percentage of ambiguous samples that exit early.

5.2.1 Difference in average uncertainty between ambiguous and unambiguous data

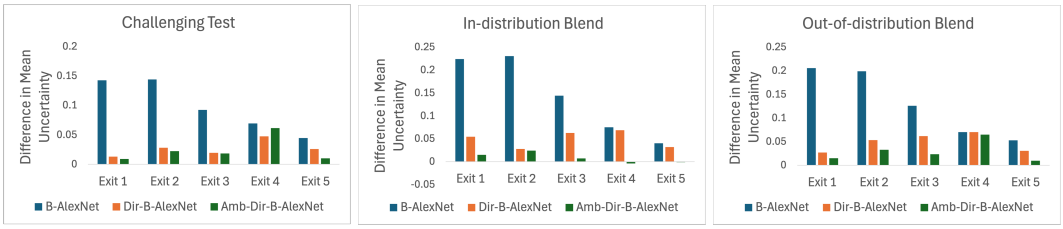
This metric measures the difference in average uncertainty of ambiguous data vs. unambiguous data. A smaller difference is preferred, as it indicates that the network is more balanced in its handling of both types of data. To compute this metric, we first calculated the mean uncertainty for each network (entropy for B-AlexNet, differential entropy for both Dir-B-AlexNet and Amb-Dir-B-AlexNet) on the baseline unambiguous dataset. Then for each network, we computed the mean uncertainty across Challenging Test, In-distribution Blend, and Out-of-distribution Blend. Given the difference in scale for entropy and differential entropy, we utilized min-max normalization to standardize the results, ensuring all values are scaled to a range of $[0, 1]$. Finally, we calculated the difference by subtracting the normalized mean uncertainty of unambiguous data from that of ambiguous data.

Results. Figure 4 presents a comparison of three networks' differences in mean uncertainty between the unambiguous baseline and three ambiguous datasets across five exits. Amb-Dir-B-AlexNet consistently demonstrates the smallest difference, followed closely by Dir-B-AlexNet. B-AlexNet exhibits the largest difference across all exits for all datasets. Notably, Amb-Dir-B-AlexNet exhibits a minimal difference at exits 1 to 3 and even shows a negative difference at exits 4 and 5 on In-distribution Blend. This indicates that it is equally and even more confident in its predictions for this group of ambiguous data compared to unambiguous data. This aligns with our expectation as it was specifically trained to learn this distribution of ambiguous data.

5.2.2 Percentage of ambiguous samples early exiting

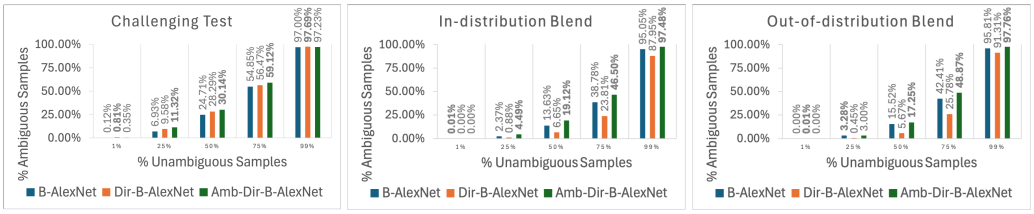
The second metric calculates the percentage of ambiguous samples that can exit early through the first branch, when thresholds are set to allow $1, 25, 50, 75, 99\%$ of unambiguous samples to exit from this branch. Thresholds were found by sweeping across a range of possible values. This metric directly assesses the practical utility of our approach in reducing inference time when handling ambiguous samples, with a higher percentage indicating a better performance.

Results. Results are shown in Figure 5. Amb-Dir-B-AlexNet again outperforms the other two models, evidenced by its consistently higher percentage values across all thresholds and datasets. To our surprise, however, despite a small advantage on Challenging Test, Dir-B-AlexNet is outperformed by B-AlexNet on In-distribution Blend and Out-of-distribution Blend. The reason for this discrepancy is not immediately clear and needs further investigation.



(a) Challenging Test (b) In-distribution Blend (c) Out-of-distribution Blend

Figure 4: A model comparison of difference in average uncertainty between ambiguous and unambiguous data.



(a) Challenging Test (b) In-distribution Blend (c) Out-of-distribution Blend

Figure 5: A model comparison of percentage of ambiguous samples exiting from exit 1 when $1, 25, 50, 75, 99\%$ unambiguous samples exit from exit 1.

6 Related work

Early exiting. Many early exiting strategies have been proposed to address the prolonged inference time in deep neural networks. BranchyNet proposes to add side branches to the main network, enabling samples to exit early when entropy values of predictions drop below predetermined thresholds [29]. Differential Branching reformulates early exiting in a differentiable format and eliminates the need for designing an explicit exiting criterion [19]. Adaptive Neural Networks solves a layer-by-layer weighted binary classification problem to trade off between future accuracy and computational cost [2]. PonderNet learns to predict the probability of halting at each step [1]. Conditional Deep Learning identifies the input difficulty via confidence measure and conditionally activate deeper portions of the network [18]. Confident Adaptive Transformers trains a meta consistency classifier for each intermediate exit, allowing samples to exit early while guaranteeing a specifiable degree of consistency with the original model with high confidence [22].

Existing work generally falls into two categories in terms of exiting criteria: 1) using some function of classification entropy as confidence measure, or 2) using a neural network to learn and predict confidence measure or exiting probability. The first approach may not adequately represent true model uncertainty for data with inherent complexity, while the second may lack interpretability and

efficiency. Our approach provides a simple yet effective way to directly measure model uncertainty and improves model performance in handling ambiguous data.

Dirichlet neural networks. As Dirichlet neural networks enable the distinction between different types of uncertainty in a prediction, they have been widely applied in out-of-distribution sample detection and uncertainty estimation. As a form of Dirichlet neural networks, Dirichlet Prior Networks (DPN) separate distributional uncertainty from both data uncertainty and model uncertainty to detect out-of-distribution samples [15, 16]. Further work built upon DPNs to maximize the representational gap between in-distribution and out-of-distribution samples [17]. Evidential Networks also demonstrate success in out-of-distribution detection and adversarial attack via Dirichlet-based uncertainty estimation [24, 23]. Belief matching framework transforms the target label into a random variable based on Bayesian principles and uses Dirichlet distribution as both the conjugate prior and the posterior to improve generalization performance and uncertainty estimation [9]. To the best of our knowledge, our approach is the first to apply Dirichlet neural networks to the early exiting setting.

7 Conclusion

Deep neural networks demonstrate impressive performance on a wide range of machine learning tasks thanks to their depth, but this often comes at the cost of prolonged inference time. To address this, early exiting strategies have been proposed to reduce the average inference time by allowing certain samples to exit early through an intermediate layer of the network. Existing exiting criteria rely on total uncertainty as a measure of model confidence, which can fail to reflect true model uncertainty in the presence of high data uncertainty. In response, we propose a Dirichlet-based framework to directly quantify model uncertainty, thereby enabling ambiguous samples to exit early when model uncertainty is low, even if data uncertainty remains high. Compared to the baseline, models trained with our approach exhibit a more balanced performance in handling ambiguous data vs. unambiguous data, and enable early exits for a larger proportion of ambiguous data.

Our study has several limitations. First and foremost, our experiments focused on a single network architecture and a specific image classification dataset. This limits the ability to apply our conclusions to deep neural networks in general. Further experiments on various network architectures and diverse datasets are crucial to fully determine the applicability and generalizability of our results. Second, the blending technique we used to generate ambiguous datasets for training may raise some questions about its validity: are blended image actually ambiguous? Blended images may contain less information than the combination of two individual images when their values cancel out. Blending can produce overly noisy images. One image may overshadow the other during the blending process, resulting in an unambiguous image instead. Third, it remains unclear why Dir-B-AlexNet is outperformed by B-AlexNet under the second evaluation metric despite a clear advantage in the first metric. Despite these limitations, our work represents a step towards developing more interpretable, highly efficient, and robust early exiting strategies.

Acknowledgments and Disclosure of Funding

This research project was supported by the NOMIS Foundation and by grant N00014-23-1-2510 from the Office of Naval Research.

References

[1] Andrea Banino, Jan Balaguer, and Charles Blundell. “Pondernet: learning to ponder”. In: *arXiv preprint arXiv:2107.05407* (2021).

[2] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. “Adaptive neural networks for efficient inference”. In: *International Conference on Machine Learning*. PMLR. 2017, pages 527–536.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec

Radford, Ilya Sutskever, and Dario Amodei. “Language models are few-shot learners”. In: *Advances in Neural Information Processing Systems*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Volume 33. Curran Associates, Inc., 2020, pages 1877–1901.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*. Edited by Jill Burstein, Christy Doran, and Thamar Solorio. Association for Computational Linguistics, 2019, pages 4171–4186.

[5] Alex Graves. “Long short-term memory”. In: *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin Heidelberg, 2012, pages 37–45.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[7] Eyke Hüllermeier and Willem Waegeman. “Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods”. In: *Machine Learning* 110.3 (Mar. 2021), pages 457–506.

[8] Yuji Ikeda. *Yuzie007/Mpltern: 1.0.4*. Version 1.0.4. Zenodo, Apr. 25, 2024. URL: `https://zenodo.org/doi/10.5281/zenodo.11068993`.

[9] Taejong Joo, Uijung Chung, and Min-Gwan Seo. “Being Bayesian about categorical probability”. In: *Proceedings of the 37th International Conference on Machine Learning*. Edited by Hal Daumé III and Aarti Singh. Volume 119. Proceedings of Machine Learning Research. PMLR, July 2020, pages 4950–4961.

[10] Armen Der Kiureghian and Ove Ditlevsen. “Aleatory or epistemic? Does it matter?” In: *Structural Safety* 31.2 (2009). Risk Acceptance and Risk Communication, pages 105–112.

[11] Alex Krizhevsky. “Learning multiple layers of features from tiny images”. Master’s thesis. University of Toronto, 2009.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems*. Edited by F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger. Volume 25. Curran Associates, Inc., 2012.

[13] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D. Lane. “Adaptive inference through early-exit networks: design, challenges and directions”. In: *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*. 2021, pages 1–6.

[14] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pages 2278–2324.

[15] Andrey Malinin and Mark Gales. “Predictive uncertainty estimation via prior networks”. In: *Advances in Neural Information Processing Systems*. Edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Volume 31. Curran Associates, Inc., 2018.

[16] Andrey Malinin and Mark Gales. “Reverse KL-divergence training of prior networks: improved uncertainty and adversarial robustness”. In: *Advances in Neural Information Processing Systems*. Edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett. Volume 32. Curran Associates, Inc., 2019.

[17] Jay Nandy, Wynne Hsu, and Mong Li Lee. “Towards maximizing the representation gap between in-domain & out-of-distribution examples”. In: *Advances in Neural Information Processing Systems*. Edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Volume 33. Curran Associates, Inc., 2020, pages 9239–9250.

[18] Priyadarshini Panda, Abhronil Sengupta, and Kaushik Roy. “Conditional deep learning for energy-efficient and enhanced pattern recognition”. In: *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2016, pages 475–480.

[19] Simone Scardapane, Danilo Comminiello, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. “Differentiable branching in deep networks for fast inference”. In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pages 4167–4171.

[20] Simone Scardapane, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. "Why should we add early exits to neural networks?" In: *Cognitive Computation* 12.5 (Sept. 2020), pages 954–966.

[21] Mike Schuster and Kuldip K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pages 2673–2681.

[22] Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. "Consistent accelerated inference via confident adaptive transformers". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Edited by Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih. Association for Computational Linguistics, Nov. 2021, pages 4962–4979.

[23] Murat Sensoy, Lance Kaplan, Federico Cerutti, and Maryam Saleki. "Uncertainty-aware deep classifiers using generative models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Volume 34. 04. Apr. 2020, pages 5620–5627.

[24] Murat Sensoy, Lance Kaplan, and Melih Kandemir. "Evidential deep learning to quantify classification uncertainty". In: *Advances in Neural Information Processing Systems*. Edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Volume 31. Curran Associates, Inc., 2018.

[25] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *3rd International Conference on Learning Representations, ICLR 2015*. Edited by Yoshua Bengio and Yann LeCun. 2015.

[26] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks". In: *Advances in Neural Information Processing Systems*. Edited by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger. Volume 27. Curran Associates, Inc., 2014.

[27] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. "Efficient processing of deep neural networks: a tutorial and survey". In: *Proceedings of the IEEE* 105.12 (2017), pages 2295–2329.

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pages 1–9.

[29] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. "BranchyNet: fast inference via early exiting from deep neural networks". In: *2016 23rd International Conference on Pattern Recognition (ICPR)*. 2016, pages 2464–2469.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. Edited by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Volume 30. Curran Associates, Inc., 2017.

[31] Ludwig Wittgenstein. *Philosophical investigations*. Translated by G. E. M. Anscombe, P. M. S. Hacker, and Joachim Schulte. Rev. 4th ed. Malden, MA: Wiley-Blackwell, 2009.

[32] Weizheng Xu, Youtao Zhang, and Xulong Tang. "Parallelizing DNN training on GPUs: challenges and opportunities". In: *Companion Proceedings of the Web Conference 2021*. New York, NY, USA: Association for Computing Machinery, 2021, pages 174–178.

A Network architectures

Figure 6 shows the AlexNet architecture we used in our experiments and Figure 7 shows the BranchyNet variant.

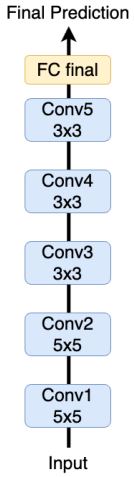


Figure 6: AlexNet

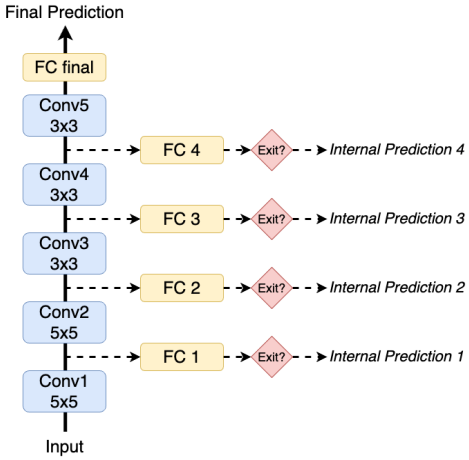


Figure 7: B-AlexNet

B Experimental setup

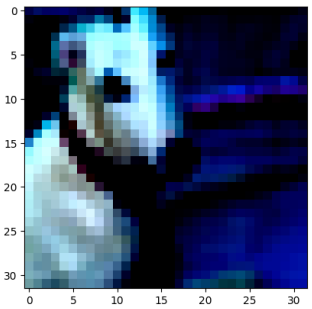
All experiments were run in Python 3.11 using an NVIDIA MIG GPU with 10 GB of GPU memory. During training, B-AlexNet, Dir-B-AlexNet, and Amb-B-AlexNet were initialized with trained weights of AlexNet on CIFAR-10. All models were trained for 100 epochs using the Adam optimizer with a learning rate of 0.001.

C Image blending process

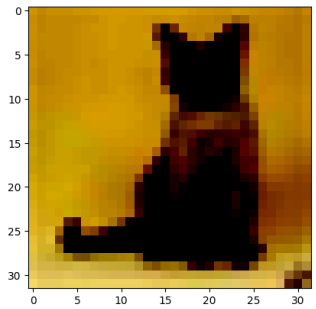
Figure 8 demonstrates the blending process in the creation of artificial ambiguous training data. Given two images A and B, blending A and B results in an image C where each pixel of C is obtained by:

$$C_{i,j} = \frac{1}{2}A_{i,j} + \frac{1}{2}B_{i,j}$$

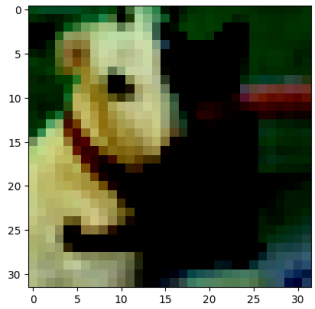
where $A_{i,j}, B_{i,j}, C_{i,j}$ are the pixels of A, B, C on row i and column j, respectively.



(a) An image of “Dog”



(b) An image of “Cat”



(c) A blended image of “Dog” and “Cat”

Figure 8: Blending an image of “Dog” and an image of “Cat” into an ambiguous image of “Cat” and “Dog”