Exploiting epistemic uncertainty at inference time for early-exit power saving

Jack Dymond[a;*], **Sebastian Stein**[a] **and Steve Gunn**[a]

[a]Electronics and Computer Science
University of Southampton
United Kingdom
ORCiD ID: Jack Dymond https://orcid.org/0000-0003-2069-716X

Abstract. Distinguishing epistemic from aleatoric uncertainty is a central idea to out-of-distribution (OOD) detection. By interpreting adversarial and OOD inputs from this perspective, we can collect them into a single *unclassifiable* group. Rejecting such inputs mid-inference will reduce resource usage. To achieve this, we apply k-nearest neighbour (KNN) classifiers to the embedding space of branched neural networks. This introduces a novel means of additional power savings, through an **early-exit reject**.

Our technique works out-of-the-box on any branched neural network and can be competitive on OOD benchmarks, achieving an area under receiver operator characteristic (AUROC) of over 0.9 in most datasets, and scores of 0.95+ when identifying perturbed inputs. A mixed input test set is introduced, we show how OOD inputs can be identified up to 50% of the time, and adversarial inputs up to 85% of the time. In a balanced test environment, this equates to power savings of up to 18% in the OOD scenario and 40% in the adversarial scenario. This allows a more stringent in-distribution (ID) classification policy, leading to accuracy improvements of 15% and 20% on the OOD and adversarial tests, respectively, when compared to conventional exit policies operating under the same conditions.

1 Introduction

With the ever-increasing power of deep learning, it is becoming increasingly feasible to apply its methods to a wider range of application areas. Yet, these systems have a tendency to over-fit, cut corners during optimisation, and make overconfident mistakes [16, 20]. This lack of robustness becomes an issue when applied to safety critical problems, for example, self-driving cars [18]. Because of this, the field of robust deep learning has recently become more active [10].

Here, robustness can refer to a variety of things. We focus on dealing with out-of-distribution (OOD) data and adversarial data, data designed deliberately to fool a classification model. Assuming the OOD input cannot be classified and the adversarial data successfully fools the model, we can refer to these collectively as *epistemically* uncertain inputs, which are inputs outside of the target distribution of the classifier and induce epistemic uncertainty in the classifier. In safety critical scenarios, it could be dangerous for the classifier to process them, for example, if a self-driving car misclassifies the moon as a traffic light and stops the car suddenly. In such cases, rejecting the classification may be more appropriate. See figure 1.

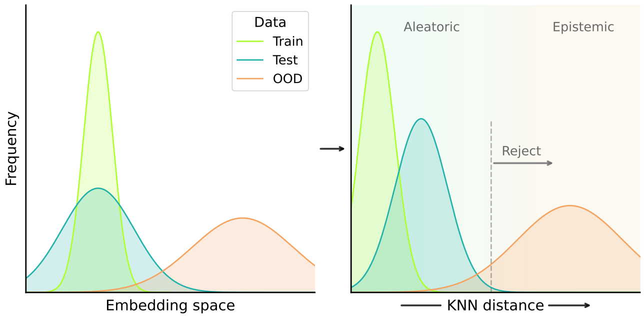

Figure 1. 1-dimensional projections of distributions in the embedding space of neural networks. The expected embedding distributions are shown in the left figure, and expected KNN distances in the right. The embedding representation of OOD data is expected to be clustered away from the target dataset. We label the uncertainty encountered in the KNN distance regions: aleatoric is that which is close to the the training distribution, epistemic is characterised with larger KNN distances.

In these environments, it would be better still if these inputs could be rejected early in the inference process to save power. Branched neural networks are a subset of neural networks with intermediate classification branches, which allow for resource saving through early exiting at inference time [47]. The idea of robustness has seldom been incorporated into the area of early exiting neural networks, and when it has, the work has focused less on handling epistemic uncertainty, and more on the classification performance on clean data [25, 35]. This overlooks an important factor that OOD and adversarial data introduces to the early exiting classifier: resource wastage.

In conventional models, the best case scenario is the input passes through the model and absolute uncertainty is outputted, e.g. [41], or a reject option is given to the classifier, e.g. [15]. In the early exiting field, there is also an opportunity to reject these inputs early and thus save resource usage. This is the core idea we address in this paper.

Achieving this, however, presents the challenge of distinguishing aleatoric uncertainty, that is uncertainty due to the randomness of the target distribution, and epistemic uncertainty, uncertainty arising due to the data being outside of the target distribution. To address this challenge, we use two uncertainty measures. Entropy is used to quantify uncertainty in the *aleatorically* uncertain inputs, and we use a KNN-based classifier to catch epistemically uncertain inputs, allowing for distribution-aware early exiting.

[*] jd5u19@soton.ac.uk

To our knowledge this is the first implementation of OOD adversarial detection in early exit architectures. Furthermore, our method works out-of-the-box, meaning no additional optimisation is required. Our contributions are as such:

- We present a study of robustness in branched neural networks, addressing OOD robustness as well as adversarial robustness.
- We introduce an easy to implement KNN-based OOD detection technique which can be applied out-of-the-box in branched neural networks.
- A novel distribution-aware early exiting system is introduced, which incorporates KNN-based OOD detection alongside conventional early exit classification. Our new early exiting system allows the classifier to save additional power through rejecting inputs it is not able to classify.
- A mixed input test set is introduced for distribution-aware early exiting. Using our method, we can save up to 40% power usage or improve accuracy by up to 20%, when compared to a conventional early exiting policy operating under the same constraints.

2 Related Work

Producing models which give confidence aware output distributions is one method of identifying both adversarial and OOD inputs, and these models have been studied in great detail [12, 37, 3]. Some methods incorporate additional data into their methodology [11, 48], use data augmentation/generation [32, 26, 21, 40, 45, 49], or use probabilistic models to adapt their loss functions [41, 30, 6, 51]. However, all of these methods adapt the training procedure for the underlying network, whereas we aimed for a method to work out-of-the-box, as to not incur additional optimisation costs.

There are a number of methods, like this work, which focus on the inference process. KNN and distance metrics for anomaly detection have been studied previously [19, 5, 27], and recently have seen use in OOD and adversarial detection [39, 1, 43]. Furthermore, the notion of rejecting classification has been studied in depth [15, 23, 4]. However, this line of work often presents a rejection at the end of inference and so functionally is not dissimilar to methods which calibrate their outputs. One such paper uses branched neural networks to classify data hierarchically to assist uncertainty quantification. However, the architecture chosen is not conducive to power savings due to the exits being positioned at the end of the model [2].

Since their inception [44, 47], branched networks have remained an integral part of the dynamic inference research community [50, 24]. More recently, like in this work, the focus has shifted to optimised inference techniques for them [9, 42, 52].

Overconfidence in branched networks has recently been investigated in [35]. However, the authors focus on ID data, and, while this is important, it was outside of the scope of this work. We instead focus on a novel method for resource savings when encountering OOD and adversarial data. Work in [25] most closely aligns with ours. The authors train branched neural networks to be robust to adversarial attacks, some of which are referred to as *slow down* attacks, which is in essence what we describe in this work. However, the authors approach this problem in a different way, instead opting to classify the data, meaning inference still requires the use of the whole network.

Branched networks offer a unique opportunity when encountering OOD and adversarial inputs. Specifically, if they can be detected, inference can be halted to prevent further time wastage. The existing literature has yet to explore the idea of rejecting classification early in the inference process, which we describe as an **early exit reject**.

To explore this idea, we first examine the effect of OOD and adversarial inputs on pre-trained branched networks. We then explore a detection method, which we incorporate into an existing early exiting paradigm. Finally we analyse the results and present our conclusions.

3 Out of Distribution and Adversarial Inputs in Branched Neural Networks

Branched neural networks are a subset of neural networks which have intermediate classification branches. We can denote the backbone as a function f and each classification branch as $f_b(x)$, where $b \in \{1, 2, ..., B\}$, and B is the total number of branches. Each branch will give the softmax output vector: y_b. We train these networks through the joint optimisation of the branches, with a weighted loss function and a target vector $\hat{y}$ with K target classes:

$$\mathcal{L}_{\text{total}}(y, \hat{y}) = \sum_{b \in B} w_b \mathcal{L}_{\text{b}}(y_b, \hat{y}) \quad (1)$$

$$\mathcal{L}_b(y_b, \hat{y}) = \sum_{i=1}^{K} y_{b,i} \log(\hat{y}_i) \quad (2)$$

Here in equation 1, y is the collection of branch outputs and w_b refers to a tunable weighting hyperparameter, which we assume sums to 1. The branch loss in equation 2 is the cross entropy loss.

Using conventional methods, neural networks are often trained to give confident classification outputs in the supervised setting. This presents a challenge when OOD data is encountered. Whilst they might not give results of high confidence, they do not return results which are of low confidence. In branched neural networks, confidence is often quantified for early exiting using entropy [46, 25]. To motivate the introduction of an additional classification method for early exit rejection, we first analyse entropy probability densities when presented with unclassifiable data, shown in figure 2.

In accordance with the benchmarks set in [54], we pass various OOD datasets to our model: Describable Textures Dataset (DTD) [8], CIFAR100 [29], Street View House Numbers (SVHN) [36], and Tiny-ImageNet [31]. We use a branched ResNet18 [22], which has 3 branches spaced equidistantly, and a final exit which we call the 4th branch. This is trained to convergence on CIFAR10 [29], achieving an accuracy of $\sim$ 95%. We believe our method is not specific to the ResNet18 architecture, as existing work uses a number of architectures with little variation in general behaviour [47, 25, 13].

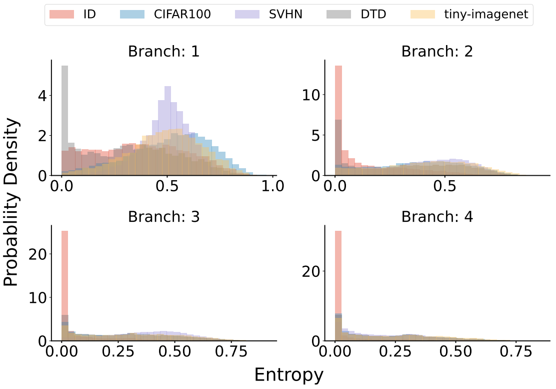

Figure 2. Histograms denoting the entropy probability densities for each branch on in-distribution (ID) data and various OOD datasets.

As indicated by the probability density graph, we find earlier branches generally give outputs of lower confidence, except for DTD, which gives a large quantity of high confidence outputs. As inputs are processed further through the network, in-distribution (ID) data is much more likely to prompt low entropy outputs than OOD data. Importantly, however, it is difficult to distinguish these distributions in the higher entropy ranges. Therefore, entropy alone is insufficient for distinguishing ID inputs from OOD inputs. In an early exiting scenario where entropy thresholds are varied at run-time, such networks will be susceptible to confusing ID data with OOD data.

We also examine adversarial inputs, using the fast gradient sign method (FGSM) to generate adversarial inputs [17]. This method applies a perturbation defined using the sign of the gradient, which is multiplied by a small value ϵ and added to the original input to create an adversarial example. This perturbation, η, can be defined as:

$$\eta = \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\phi, x, y)), \tag{3}$$

where $\mathcal{L}$ refers to the objective function of the model, ϕ the model parameters, and x and y the input and target, respectively. Using this for an adversarial attack generates our perturbed input $\tilde{x}$, defined as:

$$\tilde{x} = x + \eta,$$

In equation 3, we vary ϵ between 0 and 0.3. At 0.3, we find the performance of the model has degraded completely:

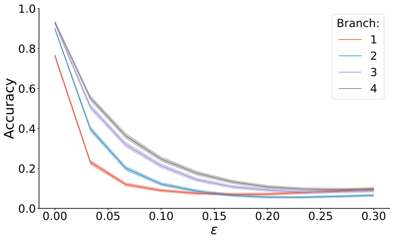

Figure 3. Accuracy degradation of each branch of the network when presented with adversarial data. We vary ϵ between 0 to 0.3, which is denoted on the x axis. Classification accuracy is shown on the y axis.

Accuracy drops prohibitively when applying the perturbation to the inputs, but the later branches handle this marginally better than the earlier branches. To understand the effect of the attacks on entropy, we analyse the entropy distributions in figure 4.

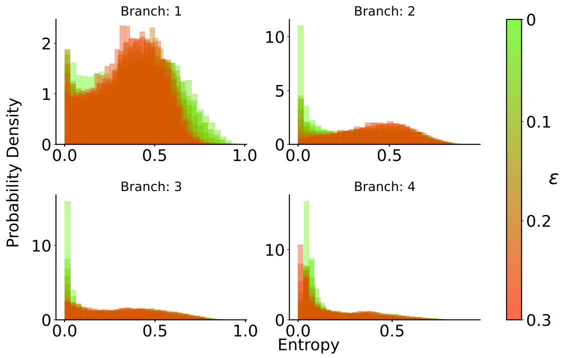

Figure 4. Histograms denoting the entropy probability densities for each branch on adversarial data.

Like with the OOD data, the first branch is less susceptible to giving overconfident outputs, but there is also a lot of overlap with the adversarial inputs. Once again, this effect lessens later in the network.

Hence, we have shown in this section, in conventional early exit policies, entropy alone is insufficient for distinguishing an ID input from an unclassifiable one. We address the challenge of doing so in the next section.

4 Distinguishing Aleatoric and Epistemic Uncertainty in Branched Neural Networks

In many scenarios where machine learning is applied, the uncertainty in the output is as important as the output itself, as it allows the user to understand how to incorporate it into their system. Uncertainty in machine learning can be broadly categorised into two domains: Aleatoric and Epistemic.

Aleatoric uncertainty arises due to the natural randomness of data, whereas epistemic uncertainty arises due to a distributional change in the input data, meaning the input is no longer in the range in which the model was trained. Validation and testing data should fall into this category to some degree. However, in balanced datasets, like the benchmarks seen in the field, this effect is not significant. For epistemic uncertainty we consider OOD datasets, those which are taken from different sources and have different class labels. We also consider adversarial inputs. As we show in this section, inputs which have been sufficiently perturbed have their distribution shifted in the latent space of the model.

When encountering OOD and adversarial inputs we have shown conventional deep learning models are susceptible to outputting similar predictions to those made on ID data. Hence, we wish to develop a second method which detects these unclassifiable inputs and allows us to classify them individually. First, we focus on OOD inputs.

4.1 Out of distribution detection

Following recent work in [43] we employ k-nearest neighbour (KNN) classification in the penultimate layer embedding space. We find this is a particularly appropriate method to use, since it can be applied to any model without the need for additional optimisation.

Consider the distribution in the embedding space, training inputs will be more tightly clustered than test inputs, but a well optimised model should position these distributions roughly about the same mean. OOD inputs, however, should be in a separate cluster. When translating this to KNN distances, the training inputs will have the lowest distances, followed closely by the test inputs, then the OOD inputs should cluster separately from the target dataset.

To test this principle, we train a baseline 4 exit branched neural network on a target dataset, recording the k$^{\text{th}}$ nearest neighbour distances on the target data test set, and a collection of OOD datasets. Results with the CIFAR10 target dataset are shown in figure 5.

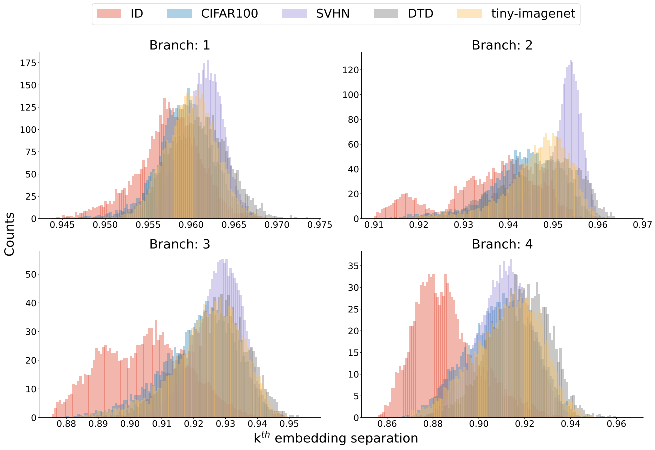

Figure 5. KNN embedding separation for in distribution (ID) data and various out of distribution datasets.

We find that there is clear separation between the datasets as the inputs progress through the model. However, in the early branches, this separation is minimal. Furthermore, in each branch, there is a large amount of overlap. This separation increases further along the inference process, and at the final exit the overlap is minimised. In some datasets, there is sufficient separation to allow for classification in the earlier branches. To examine this further, we analyse the receiver operator characteristic (ROC) of OOD detection on these branches using a KNN classifier. To generate this, we vary the position of the classification boundary between the distributions, defining this boundary according to percentiles of the validation distribution.

As shown in figure 5, the later branches are more effective in separating the OOD inputs from the target dataset. This is reflected in the area under ROC (AUROC) for each branch, shown in table 1. In the final branch, there are competitive results, considering there is no additional optimisation taking place for OOD detection. However, on some datasets, namely SVHN, there are competitive AUROC results on the earlier branches. For most of the datasets tested, we find that identification performance increases further along the network. However, we do find that for the SVHN dataset, performance is maximised in the second branch.

Table 1. Table showing the AUROC for each branch given different OOD datasets. Recent benchmarks are shown, where † denotes the benchmark which is implemented out-of-the-box, like ours. The mean and standard deviation (mean ± std) are taken over 5 runs.

Branch/Benchmark	AUROC			
	Dataset			
	CIFAR100	SVHN	DTD	Tiny-Imagenet
1	0.62±0.01	0.65±0.04	0.69±0.04	0.67±0.04
2	0.74±0.01	**0.95±0.01**	0.81±0.05	0.80±0.01
3	0.84±0.01	0.93±0.01	0.88±0.03	0.88±0.03
4	**0.85±0.01**	0.88±0.02	**0.90±0.02**	**0.89±0.01**
UDG [53]	0.90	0.93	0.94	0.93
ARPL+CS [7, 48]	0.89	0.91	0.91	0.89
RegMixup [38]	0.90	0.97	-	0.90
GROOD [48]	**0.97**	**0.99**	**0.99**	**0.96**
DNN [43]†	-	0.95	0.95	-
DNN w/ CL[43]	-	**0.99**	**0.99**	-

These results suggest that branched neural networks, even when trained using conventional supervised methods, are capable of distinguishing their target dataset from other OOD datasets. However, when models are deployed into real-world situations, it is not only OOD inputs they might encounter. In some scenarios they will encounter adversarial attacks, designed to deliberately confuse the model. We consider this case in the next section.

4.2 Detecting adversarial attacks

Adversarial perturbations in the inputs present a different challenge, since their effect on the output classification is non-binary. That is, there is a continuous range between the unperturbed inputs and those that are completely perturbed. Hence, some inputs, whilst being perturbed, are still able to be classified.

Therefore, rather than classifying the perturbed from the clean inputs, it is more pertinent to identify which inputs will be classified by the model and which inputs are unclassifiable.

To analyse their distribution in the embedding space, we follow a similar analysis to the previous section. We analyse the distributional shift of the perturbation of the CIFAR10 dataset in figure 6.

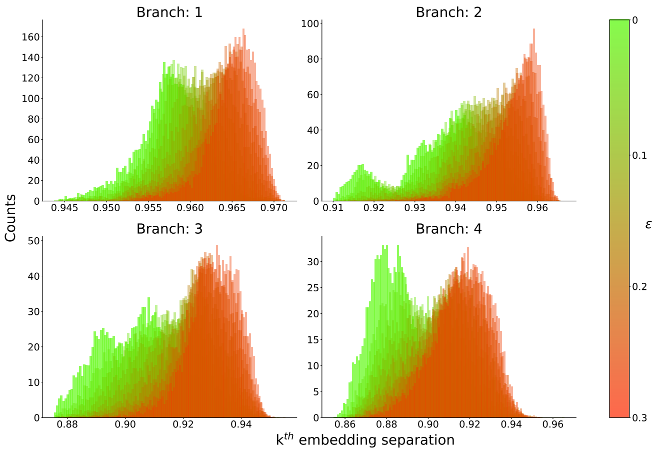

Figure 6. KNN embedding separation for various perturbation values of the target dataset, ranging from $e = 0 \rightarrow 0.3$. We represent increasing perturbation visually by varying the distribution colours from green to red.

We find that as the data becomes increasingly perturbed, its distribution in the embedding space separates from the clean data. Much like the OOD data in section 4.1, the distributional shift is most evident in the later layers. However, there is a clear shift as perturbation increases in all branches.

To further examine this, we use the same technique as in the previous section to determine the ROC curve of the model at each branch, for each perturbation amount. We show these results in figure 7, and we present AUROC in the appendix.

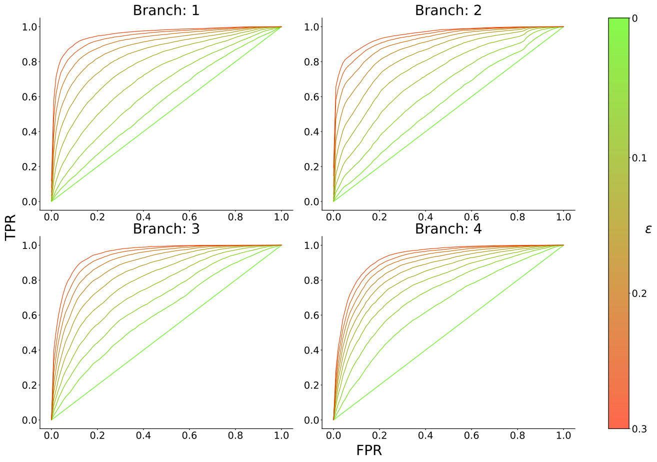

Figure 7. ROC curves for each branch on adversarial samples for various perturbation values of the target dataset. In all cases, the x and y axes denote the false positive rate (FPR), and true positive rate (TPR), respectively.

We find that the later branches are more successful in identifying the lower perturbation values. However, as the perturbation increases, the AUROC performance increases in the earlier branches, and at the greatest value of ϵ, we find that the first branch distinguishes the adversarial samples most successfully.

As expected, we find that the KNN classifier does not work at low values of ϵ. However, at these values the performance of the model is not prohibitively poor and the performance at greater values of ϵ is competitive. Hence, by treating significantly perturbed values like OOD inputs, there is a potential to detect and reject the classification of such inputs. We consider this scenario and the cost benefits it can achieve in the next section.

5 Rejecting Classification to Save Power

We have so far established that branched neural networks can detect OOD inputs, as well as adversarial inputs. We now consider the beneficial situation branched neural networks allow for: using epistemic uncertainty to minimise resource usage. To accomplish this, we propose a multi-step modification to the conventional exit policy.

First, the aleatoric uncertainty is handled, ID domain inputs will be processed normally, this will catch any opportunities to perform an **early exit classification**. For this, we employ an entropic decision policy on the softmax output of each classification branch. We impose a classification threshold on the entropy of this output, defined as λ. If this condition is not met, we move to the epistemic uncertainty quantification phase, designed to classify inputs outside of the training distribution using the KNN distance. If this distance exceeds a certain threshold, δ, then an **early exit reject** can take place. If neither condition is met, inference moves to the next branch.

Much like work in [43], we define our KNN threshold using the validation data distribution on the training set and use the 50th nearest neighbour, as this empirically produces the best results. We model it as a Gaussian distribution, and take thresholds on the KNN values based on the percentiles of this distribution. That is, we define a classification boundary that contains δ of the validation distribution. We vary this value between 1.0 and 0.9 at selected intervals.

To test OOD detection we introduce a mixed dataset, $\mathcal{D}$, which contains in equal parts ID inputs and OOD inputs. We produce these for four OOD datasets, Describable Textures Dataset[1] (DTD), CIFAR100, Street View House Numbers (SVHN), and Tiny-ImageNet. We then pass the dataset to our branched neural network, f. Our *distribution-aware* early exiting process is detailed in algorithm 1.

Algorithm 1. Distibution-aware early exiting algorithm

```
for $x$ in $\mathcal{D}$:
   early_exit $\rightarrow$ False
   for $b$ in $B$:
      embedding $\rightarrow$ $f_{b-1}(x)$
      out $\rightarrow$ $f_b(x)$

      entropy $\rightarrow$ ent(out)
      if entropy < $\lambda$:
         early_exit $\rightarrow$ True
         pred[$x$] $\rightarrow$ argmax(out)
         break

      knn $\rightarrow$ get_knn(embedding,train_emb)
      if knn[k] > $\delta$:
         early_exit $\rightarrow$ True
         pred[$x$] $\rightarrow$ -1
         break

   if early_exit = False:
      final_out $\rightarrow$ $f_B(x)$
      pred[$x$] $\rightarrow$ argmax(final_out)
```

Here `train_emb` refers to the collected train embeddings, `get_knn`() a function returning the k nearest neighbours, and `k` the value of k. Following work in [43], we find k=50 empirically returns the best results. Hence, we introduce two conditions for an early exit, increasing the opportunity for early exits to take place. To understand the performance of models using such inference methods, we analyse their operating ranges, much like the work in [13].

We vary the entropy threshold for classification, λ, from 0 to the maximum value, $\log(K)$. This is to understand the operating points at which the model can work in the accuracy-power space, where accuracy refers to that on the ID data and power the average MAC operations at that particular threshold.

We also scale the KNN distance threshold, δ, depending on the entropy threshold. We do this between the KNN distance corresponding to the selected detection percentile and that corresponding to the 100th percentile. This is so that as the classification threshold becomes more stringent and conservative, the KNN rejection threshold does the same. In practice we normalise both λ and δ.

The operating range for OOD detection and adversarial detection are shown in figures 8 and 9 respectively. We show the ID accuracy against the average power usage, for a given entropy exit threshold. A number of minimum values for δ are shown. We show a conventional exit policy in red, which without the early exit reject recourse is forced to process all inputs.

By assuming the accuracy of the model represents a point of a Gaussian, we can define the confidence interval as:

$$\Delta = \sigma\sqrt{\frac{A(1-A)}{n}}.$$

Where σ represents the number of standard deviations from the mean, n the number of samples, and A the mean accuracy. For the following results, a σ of 3 is used corresponding to a certainty of 99.7%. This value Δ is represented by the shaded areas surrounding the data in figures 8 and 9.

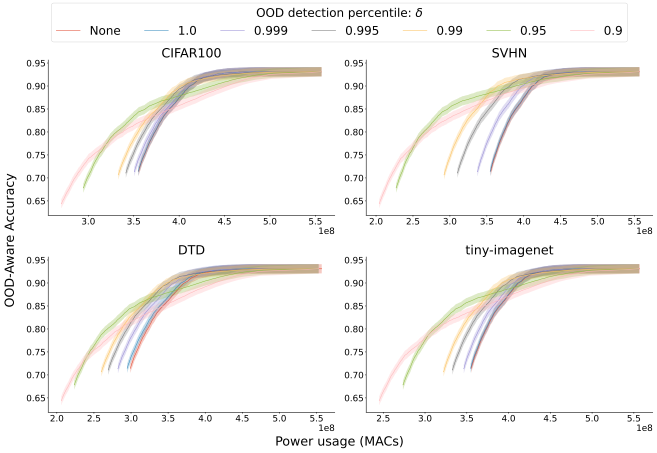

Figure 8. Operating range of the distribution-aware early exiting algorithm on various OOD datasets. In all cases the x axis denotes ID accuracy, y axis denotes average multiply accumulate operations (MACs).

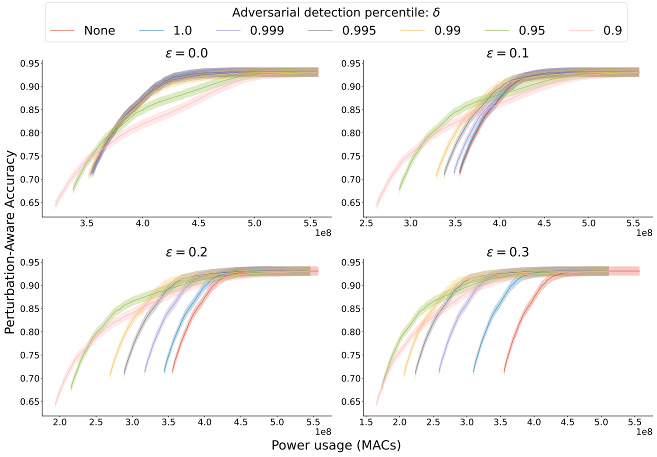

Figure 9. Operating range of distribution-aware early exiting algorithm on adversarial inputs with a selection of ϵ values.

[1] Since DTD only has $\sim$ 5000 input patterns, we use all of the inputs of this dataset for OOD detection.

We find that modest gains can be achieved with OOD CIFAR100 inputs, where the exit policy can improve accuracy across the entire power range of the model, and operate at lower power ranges when the inference policy is at its lowest threshold. The algorithm performs better on the other datasets. There is a clear distinction between each inference policy threshold, across most of the MACs range. We find that the most gains can be achieved on SVHN, with a δ of 0.99, there is no drop in ID accuracy for large power savings.

On adversarial data, we find that the model has difficulty recognising the adversarial inputs, at small values of ϵ, and hence early exiting gains are minimal. However, as ϵ increases, the early exiting gains become more prevalent. At ϵ=0.3 the δ=0.99 exit policy makes significant power reduction gains across the entire accuracy range. In both tests the δ of 0.95 and 0.90 policies have greater power savings, but with significant drops in accuracy.

These findings will allow the model to operate much more stringent ID classification policies, when compared to the conventional early exiting method which wastes resources on the OOD and adversarial data. When the model is rejecting at most 1% of the ID samples ($\delta \geq 0.99$), we do not see this reflected in the ID performance readings. Hence, any power gain is effectively *free* compared to the conventional exit policy. We analyse the performance gains of our distribution-aware exiting process in more detail in the next section.

6 Results

Since the power savings inherently depend on the ratio of ID inputs to unclassifiable inputs in $\mathcal{D}$, it is more informative to first consider the detection accuracy. This is shown for OOD inputs in table 2.

Table 2. OOD detection results for various OOD datasets. Relative ID accuracy is shown, and we enter results for a number of different detection percentile thresholds.

		OOD Detection (%)			
Relative ID	Threshold	Dataset			
Performance (%)	percentile	CIFAR100	SVHN	DTD	Tiny-Imagenet
Zero Loss	1.0	0.36±0.11	0.44±0.29	2.64±1.24	0.77±0.21
99	0.999	3.53±0.72	6.12±3.12	13.29±3.19	6.59±1.06
	0.995	10.09±0.53	19.55±4.81	27.06±3.19	17.12±0.92
	0.99	**13.38±1.17**	**26.66±5.58**	**32.79±3.19**	**22.05±1.18**
	0.95	12.70±0.62	25.59±5.43	29.99±2.55	20.68±0.71
	0.90	10.53±0.19	21.02±3.92	21.94±0.91	16.05±0.67
95	0.999	4.40±0.77	7.89±3.12	15.59±3.15	8.22±1.14
	0.995	13.80±1.17	27.53±5.99	33.30±3.54	22.65±1.26
	0.99	21.79±0.75	43.97±7.51	45.43±3.45	33.36±1.15
	0.95	**42.34±0.93**	**74.11±5.87**	**66.80±2.70**	**57.21±2.36**
	0.90	40.90±1.22	72.58±5.89	65.49±3.24	55.75±2.49
90	0.999	4.77±0.73	8.64±3.25	16.46±3.12	8.82±1.09
	0.995	15.08±1.12	30.17±6.02	35.20±3.59	24.45±1.37
	0.99	23.61±0.90	47.43±7.92	46.87±3.54	35.70±1.06
	0.95	54.43±0.93	85.37±3.36	76.86±3.18	68.97±1.62
	0.90	**61.95±0.69**	**90.42±2.18**	**82.23±2.53**	**75.62±1.32**

We find, as the figures in section 5 suggest, that the optimal threshold depends on the allowable ID accuracy drop. However $\delta = 0.99$ operates well with little drop in performance. We define a *zero loss* row, which denotes either a threshold of $\delta = 1.0$ or 100% relative ID accuracy, detection performance is limited without a small compromise in ID performance.

We find the most liberal exiting policies allow for up to 90% of the OOD samples to be detected. Whilst a relative performance drop of 1% allows for 10-30% of the OOD inputs to be detected, depending on the dataset. Table 3 shows similar results for adversarial data.

Since adversarial perturbation is not guaranteed to result in an incorrect classification, like OOD data, rather than denoting the raw detection accuracy, we detail the inference accuracy. That is, we allow the KNN classifier to predict the correctness of the output, as opposed to the origin of the input.

Table 3. Adversarial detection results for selected thresholds. We show results for a variety of relative ID accuracies.

		Adversarial Detection Performance (%)			
Relative ID	Threshold	ϵ			
Performance (%)	percentile	0.0	0.1	0.2	0.3
Zero Loss	1.0	93.26±0.29	30.41±1.90	19.81±3.32	23.40±8.90
99	0.999	93.36±0.30	33.37±2.19	32.57±8.19	43.48±16.57
	0.995	93.59±0.30	38.18±2.23	45.64±8.66	60.63±14.24
	0.99	**93.74±0.29**	**40.54±2.04**	**50.85±8.66**	**66.79±12.96**
	0.95	93.72±0.29	39.77±1.96	47.51±7.65	61.52±10.93
	0.90	93.68±0.29	37.69±1.59	38.34±4.97	47.47±7.83
95	0.999	93.39±0.30	34.14±2.39	35.06±8.88	46.68±16.78
	0.995	93.75±0.29	40.79±2.20	51.37±8.67	67.23±13.16
	0.99	94.12±0.29	45.83±1.88	60.47±7.38	76.79±9.34
	0.95	**95.30±0.27**	**58.47±2.24**	**76.93±5.32**	**90.71±3.39**
	0.90	95.21±0.25	57.70±2.10	75.96±5.52	90.11±3.72
90	0.999	93.40±0.29	34.36±2.38	35.68±8.95	47.34±16.79
	0.995	93.81±0.29	41.52±2.21	52.75±8.69	68.79±12.71
	0.99	94.22±0.27	46.97±1.97	62.22±7.09	78.51±8.53
	0.95	96.20±0.17	66.26±1.81	84.37±3.66	94.72±2.18
	0.90	**96.88±0.23**	**71.15±1.93**	**87.96±2.88**	**96.41±1.51**

We again find a δ=0.99 is the most effective. There is very little drop in relative performance at an ϵ of 0.0, which is likely due to the base classifier achieving $\sim$ 95% accuracy. However, this performance drops as ϵ increases, before rising again. At higher perturbation levels, we find over 30% of these inputs can be classified, with no performance drop in the classifier. Allowing a 1% drop increases this detection accuracy to ~80%. Detection accuracy peaks at ~86% when relative performance is at 90%.

To better understand the performance gains this gives rise to, we analyse the power savings and accuracy benefits from the operating range curves shown in figures 8 and 9. We can record accuracy increases, and power increases, by taking the difference in these values from the figures. This is demonstrated in figure 10.

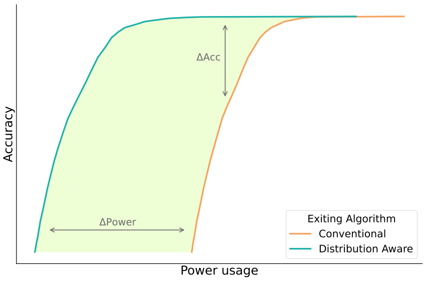

Figure 10. Accuracy change is denoted by the distance between the curves on the y axis, and power improvement by the difference on the x axis. The shaded area represents the improvement made over the conventional algorithm.

We record the maximum increases in each of these metrics for our exit policies. These values are shown for OOD data and adversarial data in tables 4 and 5, respectively.

For OOD data we find the highest peak accuracy gains are made in δ=0.99 and δ=0.95 thresholds, depending on the dataset. It should be noted, however, that from figure 8 it is evident the δ=0.99 policy can make similar accuracy gains without moving far from the maximum accuracy of the base model. Peak performance gains are achieved by the least stringent detection threshold: δ=0.90. But, accuracy values show this is at the expense of decreased ID performance.

In the adversarial test, we find that the top performing detection threshold is dependent in some part on the test data. In this experiment, the δ=0.995 threshold is marginally better than the δ=0.99 threshold. Once again, in figure 9, it is evident the 0.99 policy makes similar accuracy gains without compromising maximum ID accuracy. Peak gains are again achieved by the least stringent detection threshold: δ=0.90, at the expense of decreased performance.

All recorded results throughout the paper were averaged over 5 runs of the experiment, each trained from random initialisations. Results using CIFAR100 as the ID dataset are shown in the appendix.

Table 4. Peak improvements made by the early exiting algorithm on $\mathcal{D}$, for a number of different OOD datasets, when compared to conventional early exiting algorithms. We compare a number of different detection thresholds, peak accuracy and power improvements are shown. We highlight the best performing values for each dataset.

		Peak Improvement (%)			
Detection Threshold	Improvement	Dataset			
		CIFAR100	SVHN	DTD	Tiny-Imagenet
1.0	Acc	0.18±0.01	0.23±0.19	1.00±0.52	0.45±0.18
	Power	0.08±0.03	0.11±0.07	0.80±0.39	0.20±0.06
0.999	Acc	1.97±0.51	4.03±2.50	4.83±1.38	3.92±0.66
	Power	0.75±0.12	1.62±1.04	2.41±0.57	1.49±0.25
0.995	Acc	5.69±0.55	10.96±3.04	8.97±0.76	9.35±0.96
	Power	2.79±0.25	6.13±2.05	5.78±0.57	4.76±0.57
0.99	Acc	8.56±0.60	**13.91±2.27**	**10.74±0.50**	**11.87±0.94**
	Power	4.84±0.19	10.24±2.31	8.44±0.37	7.64±0.67
0.95	Acc	**9.04±1.60**	7.20±1.68	9.06±1.59	8.24±2.04
	Power	14.77±0.40	24.43±1.47	18.52±0.59	19.62±1.20
0.90	Acc	-0.75±2.24	-4.80±2.36	0.89±2.03	-1.81±1.66
	Power	**22.67±0.66**	**31.44±1.91**	**25.30±0.63**	**27.72±1.26**

7 Conclusions

This paper considers a key idea in the robustness space for neural networks: distinguishing uncertainty of epistemic origin, from that of aleatoric origin. We introduce KNN classifiers to branched neural networks in order to detect OOD and adversarial inputs, that is, inputs they are not equipped to classify. This allows for an additional avenue of resource savings in these systems: early exit reject.

We present extensive experimentation on pre-trained branched neural networks and motivate the detection of such samples. Our proposed approach functions on a number of benchmark OOD tests and on FGSM adversarial attacks. AUROC results are competitive, given the out-of-the-box nature of our method.

A novel early exiting algorithm is detailed, which allows for early exit rejects. To show the entire operating range of the model, we vary the classification and rejection thresholds in unison. This allows us to understand the performance gains our exiting algorithm can make over conventional methods, recording the peak gains.

We show up to $\sim$90% of OOD data can be detected and rejected using our methods. When compared to conventional early exiting methods under the same resource constraints, we show this can lead to a $\sim$15% accuracy improvement, or a $\sim$30% power improvement.

We also find that substantial performance gains can be achieved through detecting adversarial inputs. We instead consider the final model accuracy as our detection target and find we can detect up to $\sim$95% of these inputs. Comparing to a conventional exiting method under the same constraints, we find our exiting algorithm records accuracy improvements of $\sim$20% or power saving of $\sim$40%, depending on the strength of the adversarial attack. We find our algorithm is better at detecting the stronger adversarial attacks.

Future work will investigate applications to more advanced adversarial attacks, more advanced detection techniques, and robustness-aware optimisation of the network backbone.[2]

Acknowledgements

This work was supported and funded by: The UK Research and Innovation (UKRI) Centre for Doctoral Training in Machine Intelligence for Nano-electronic Devices and Systems [EP/S024298/1]; the UKRI Turing AI Acceleration Fellowship on Citizen-Centric AI Systems [EP/V022067/1]; and the Defence Science and Technology Laboratory (Dstl).

Table 5. Peak improvements made by the early exiting algorithm on $\mathcal{D}$, for a number of different ϵ values, when compared to conventional early exiting algorithms. We compare a number of different detection thresholds, peak accuracy and power improvements are shown.

		Peak Improvement (%)			
Detection Threshold	Improvement	ϵ			
		0.0	0.1	0.2	0.3
1.0	Acc	0.00±0.00	0.66±0.34	5.31±3.04	9.24±5.39
	Power	0.00±0.00	0.31±0.15	3.07±1.85	6.99±4.48
0.999	Acc	0.16±0.01	4.14±1.77	12.64±3.45	15.44±4.10
	Power	0.14±0.00	1.61±0.75	7.63±3.73	13.06±5.82
0.995	Acc	0.62±0.08	8.67±1.28	**15.90±1.83**	**16.52±1.95**
	Power	0.70±0.01	4.44±1.10	14.23±4.05	21.78±5.13
0.99	Acc	1.52±0.14	**10.88±1.12**	15.46±1.83	15.11±2.01
	Power	1.37±0.01	6.87±1.18	18.22±3.68	26.11±3.95
0.95	Acc	**3.34±0.36**	8.61±1.98	5.90±2.64	4.16±1.74
	Power	6.42±0.04	17.51±1.35	30.01±2.58	35.74±1.69
0.90	Acc	1.05±1.00	-1.51±2.23	-6.75±1.49	-6.75±1.49
	Power	**12.20±0.11**	**25.46±1.51**	**36.39±1.75**	**40.33±0.84**

References

[1] Ahmed Abusnaina, Yuhang Wu, Sunpreet Arora, Yizhen Wang, Fei Wang, Hao Yang, and David Mohaisen, 'Adversarial example detection using latent neighborhood graph', in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7687–7696, (October 2021).

[2] Raphaël Achddou, J Matias Di Martino, and Guillermo Sapiro, 'Nested learning for multi-level classification', in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2815–2819. IEEE, (2021).

[3] Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges, 'Adversarial example detection for dnn models: A review and experimental comparison', *Artificial Intelligence Review*, **55**(6), 4403–4462, (2022).

[4] Marília Barandas, Duarte Folgado, Ricardo Santos, Raquel Simão, and Hugo Gamboa, 'Uncertainty-based rejection in machine learning: Implications for model development and interpretability', *Electronics*, **11**(3), 396, (2022).

[5] Liron Bergman, Niv Cohen, and Yedid Hoshen, 'Deep nearest neighbor anomaly detection', *arXiv preprint arXiv:2002.10445*, (2020).

[6] Bertrand Charpentier, Daniel Zügner, and Stephan Günnemann, 'Posterior network: Uncertainty estimation without ood samples via density-based pseudo-counts', *Advances in Neural Information Processing Systems*, **33**, 1356–1367, (2020).

[7] Guangyao Chen, Peixi Peng, Xiangqian Wang, and Yonghong Tian, 'Adversarial reciprocal points learning for open set recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **44**(11), 8065–8081, (2021).

[8] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi, 'Describing textures in the wild', in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, (2014).

[9] Xin Dai, Xiangnan Kong, and Tian Guo, 'Epnet: Learning to exit with flexible multi-branch network', in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, p. 235–244, New York, NY, USA, (2020). Association for Computing Machinery.

[10] Nathan Drenkow, Numair Sani, Ilya Shpitser, and Mathias Unberath, 'A systematic review of robustness in deep learning for computer vision: Mind the gap?', *arXiv preprint arXiv:2112.00639*, (2021).

[11] Xuefeng Du, Xin Wang, Gabriel Gozum, and Yixuan Li, 'Unknown-aware object detection: Learning what you don't know from videos in the wild', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13678–13688, (2022).

[12] Jack Dymond, 'Graceful degradation and related fields', *arXiv preprint arXiv:2106.11119*, (2021).

[13] Jack Dymond, Sebastian Stein, and Steve R Gunn, 'Adapting branched networks to realise progressive intelligence', in *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*. BMVA Press, (2022).

[14] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner, 'Detecting adversarial samples from artifacts', *arXiv preprint arXiv:1703.00410*, (2017).

[15] Yonatan Geifman and Ran El-Yaniv, 'Selectivenet: A deep neural net-

[2] Source code and supplementary material can be found at: `github.com/J-Dymond/distribution-aware-exiting`

work with an integrated reject option', in *International conference on machine learning*, pp. 2151–2159. PMLR, (2019).

[16] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann, 'Shortcut learning in deep neural networks', *Nature Machine Intelligence*, **2**(11), 665–673, (2020).

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy, 'Explaining and harnessing adversarial examples', *arXiv preprint arXiv:1412.6572*, (2014).

[18] Nathan A. Greenblatt, 'Self-driving cars and the law', *IEEE Spectrum*, **53**(2), 46–51, (2016).

[19] Xiaoyi Gu, Leman Akoglu, and Alessandro Rinaldo, 'Statistical analysis of nearest neighbor methods for anomaly detection', *Advances in Neural Information Processing Systems*, **32**, (2019).

[20] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger, 'On calibration of modern neural networks', in *International conference on machine learning*, pp. 1321–1330. PMLR, (2017).

[21] Danijar Hafner, Dustin Tran, Timothy Lillicrap, Alex Irpan, and James Davidson, 'Noise contrastive priors for functional uncertainty', in *Uncertainty in Artificial Intelligence*, pp. 905–914. PMLR, (2020).

[22] 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*, pp. 770–778, (2016).

[23] Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis, 'Machine learning with a reject option: A survey', *arXiv preprint arXiv:2107.11277*, (2021).

[24] Hanzhang Hu, Debadeepta Dey, Martial Hebert, and J Andrew Bagnell, 'Learning anytime predictions in neural networks via adaptive loss balancing', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3812–3821, (2019).

[25] Ting-Kuei Hu, Tianlong Chen, Haotao Wang, and Zhangyang Wang, 'Triple wins: Boosting accuracy, robustness and efficiency together by enabling input-adaptive inference', *arXiv preprint arXiv:2002.10025*, (2020).

[26] Bo Huang, Yi Wang, and Wei Wang, 'Model-agnostic adversarial detection by random perturbations.', in *IJCAI*, pp. 4689–4696, (2019).

[27] Ryo Kamoi and Kei Kobayashi, 'Why is the mahalanobis distance effective for anomaly detection?', *arXiv preprint arXiv:2003.00402*, (2020).

[28] Ziv Katzir and Yuval Elovici, 'Detecting adversarial perturbations through spatial behavior in activation spaces', in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, (2019).

[29] Alex Krizhevsky et al., 'Learning multiple layers of features from tiny images', (2009).

[30] Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach, 'Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration', *Advances in neural information processing systems*, **32**, 12316–12326, (2019).

[31] Ya Le and Xuan Yang, 'Tiny imagenet visual recognition challenge', *CS 231N*, **7**(7), 3, (2015).

[32] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin, 'Training confidence-calibrated classifiers for detecting out-of-distribution samples', *arXiv preprint arXiv:1711.09325*, (2017).

[33] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin, 'A simple unified framework for detecting out-of-distribution samples and adversarial attacks', *Advances in neural information processing systems*, **31**, (2018).

[34] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey, 'Characterizing adversarial subspaces using local intrinsic dimensionality', in *International Conference on Learning Representations*.

[35] Lassi Meronen, Martin Trapp, Andrea Pilzer, Le Yang, and Arno Solin, 'Fixing overconfidence in dynamic neural networks', *arXiv preprint arXiv:2302.06359*, (2023).

[36] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng, 'Reading digits in natural images with unsupervised feature learning', in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, (2011).

[37] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel, 'Deep learning for anomaly detection: A review', *ACM computing surveys (CSUR)*, **54**(2), 1–38, (2021).

[38] Francesco Pinto, Harry Yang, Ser-Nam Lim, Philip Torr, and Puneet K. Dokania, 'Using mixup as a regularizer can surprisingly improve accuracy & out-of-distribution robustness', in *Advances in Neural Information Processing Systems*, eds., Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, (2022).

[39] Catarina Pires, Marília Barandas, Letícia Fernandes, Duarte Folgado, and Hugo Gamboa, 'Towards knowledge uncertainty estimation for open set recognition', *Machine Learning and Knowledge Extraction*, **2**(4), 505–532, (2020).

[40] Murat Sensoy, Lance Kaplan, Federico Cerutti, and Maryam Saleki. Uncertainty-aware deep classifiers using generative models, 2020.

[41] Murat Sensoy, Lance Kaplan, and Melih Kandemir, 'Evidential deep learning to quantify classification uncertainty', in *Advances in Neural Information Processing Systems 31*, eds., S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, 3179–3189, Curran Associates, Inc., (2018).

[42] Tianxiang Sun, Yunhua Zhou, Xiangyang Liu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu, 'Early exiting with ensemble internal classifiers', *arXiv preprint arXiv:2105.13792*, (2021).

[43] Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li, 'Out-of-distribution detection with deep nearest neighbors', in *International Conference on Machine Learning*, pp. 20827–20840. PMLR, (2022).

[44] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, 'Going deeper with convolutions', in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, (2015).

[45] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin, 'Csi: Novelty detection via contrastive learning on distributionally shifted instances', in *Advances in Neural Information Processing Systems*, eds., H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, volume 33, pp. 11839–11852. Curran Associates, Inc., (2020).

[46] Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. Branchynet: Fast inference via early exiting from deep neural networks, 2017.

[47] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung, 'Branchynet: Fast inference via early exiting from deep neural networks', in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 2464–2469. IEEE, (2016).

[48] Tomas Vojir, Jan Sochman, Rahaf Aljundi, and Jiri Matas. Calibrated out-of-distribution detection with a generic representation, 2023.

[49] Shuo Wang, Surya Nepal, Alsharif Abuadbba, Carsten Rudolph, and Marthie Grobler, 'Adversarial detection by latent style transformations', *IEEE Transactions on Information Forensics and Security*, **17**, 1099–1114, (2022).

[50] Y. Wang, J. Shen, T. K. Hu, P. Xu, T. Nguyen, R. Baraniuk, Z. Wang, and Y. Lin, 'Dual dynamic inference: Enabling more efficient, adaptive, and controllable deep inference', *IEEE Journal of Selected Topics in Signal Processing*, **14**(4), 623–633, (2020).

[51] Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li, 'Mitigating neural network overconfidence with logit normalization', in *International Conference on Machine Learning*, pp. 23631–23644. PMLR, (2022).

[52] Maciej Wolczyk, Bartosz Wójcik, Klaudia Bałazy, Igor T. Podolak, Jacek Tabor, Marek Śmieja, and Tomasz Trzcinski, 'Zero time waste: Recycling predictions in early exit neural networks', in *Advances in Neural Information Processing Systems*, eds., A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, (2021).

[53] Jingkang Yang, Haoqi Wang, Litong Feng, Xiaopeng Yan, Huabin Zheng, Wayne Zhang, and Ziwei Liu, 'Semantically coherent out-of-distribution detection', in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8301–8309, (2021).

[54] Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, WENXUAN PENG, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, et al., 'Openood: Benchmarking generalized out-of-distribution detection', in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, (2022).

[55] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael Jordan, 'Ml-loo: Detecting adversarial examples with feature attribution', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 6639–6647, (2020).

[56] Shigeng Zhang, Shuxin Chen, Xuan Liu, Chengyao Hua, Weiping Wang, Kai Chen, Jian Zhang, and Jianxin Wang, 'Detecting adversarial samples for deep learning models: A comparative study', *IEEE Transactions on Network Science and Engineering*, **9**(1), 231–244, (2022).

A Additional Results

A.1 *Sensitivity with k*

We analyse the sensitivity of our algorithm as k is changed in the KNN algorithm. We change it from 1 to 5, and then to 150 at increments of 5. We find that AUROC does not increase significantly as we do this, we present results in figure 11.

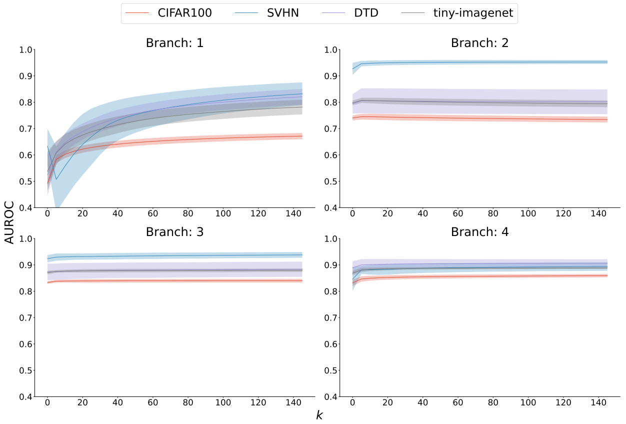

Figure 11. AUROC results shown on the y axis as k is increased, shown on the x axis. In all plots the shaded area denotes the standard deviation across 3 runs of the experiment.

In the first branch it is evident that the performance of our algorithm would benefit from sampling more nearest neighbour distances. However, as we get further along the branches of the network, we find there are diminiishing returns as k is increased.

A.2 *CIFAR10 as ID*

We show AUROC results for the branched network on adversarial data, comparing against various benchmarks. This is shown in table 11. We find our method is most competitive at higher perturbation values.

A.3 *CIFAR100 as ID*

We also repeated our experiments with CIFAR100 as the ID dataset, instead treating CIFAR10 as OOD. We present our results in this section.

AUROC readings are shown for OOD data below in table 6, compared with the benchmarks we could find in the literature. We again find that we get competitive results when this dataset is used as ID.

Table 6. Table showing the AUROC for each branch given different OOD datasets. The best performing branch for each dataset is highlighted. Recent benchmarks are shown. The mean and standard deviation are taken over 5 runs, and CIFAR100 was used as the ID dataset.

	AUROC			
	Dataset			
Branch/Benchmark	CIFAR10	SVHN	DTD	Tiny-Imagenet
1	0.59±0.00	0.74±0.05	0.48±0.02	0.55±0.02
2	0.59±0.01	0.81±0.03	0.80±0.03	0.76±0.03
3	0.62±0.02	**0.91±0.02**	**0.90±0.01**	0.81±0.01
4	**0.75±0.01**	0.81±0.02	0.85±0.01	**0.83±0.01**
UDG [53]	0.76	0.88	0.80	0.77
RegMixup [38]	**0.81**	**0.89**	-	**0.83**

We take the same readings for the adversarial data in table 12. We find our algorithm gets competitive results at higher perturbation values.

We also present the OOD detection accuracy in table 7. We find more modest detection accuracies with this ID dataset, however, there can still be high detection accuracy with a small drop in relative ID accuracy.

Table 7. OOD detection results for various OOD datasets, with CIFAR100 as ID. Relative ID accuracy is shown, and we enter results for a number of different detection percentile thresholds. We take the mean and standard deviation over 5 runs. That is, we record the results from 5 models trained from different random initialisations.

		OOD Detection (%)			
Relative ID	Threshold	Dataset			
Performance (%)	percentile	CIFAR10	SVHN	DTD	Tiny-Imagenet
ZeroLoss	1.0	0.09±0.08	0.08±0.04	3.19±1.60	0.39±0.29
99	0.999	0.39±0.12	0.97±0.46	9.95±1.58	1.75±0.40
	0.995	1.53±0.12	5.43±2.90	20.36±1.81	5.24±0.67
	0.99	2.19±0.31	**7.35±3.43**	**24.14±2.37**	**7.09±1.11**
	0.95	2.41±0.30	8.30±3.86	21.55±1.46	7.31±0.94
	0.90	**2.42±0.21**	7.47±2.70	16.46±1.35	6.59±0.63
95	0.999	0.58±0.15	1.91±1.18	12.53±1.69	2.44±0.56
	0.995	2.59±0.24	9.09±4.25	26.66±1.12	8.26±0.71
	0.99	4.63±0.41	15.54±6.61	34.68±1.11	13.37±1.10
	0.95	**11.87±0.74**	**34.38±7.97**	51.79±1.72	**27.54±1.54**
	0.90	11.67±0.66	34.06±8.39	**55.39±1.13**	27.28±1.54
90	0.999	0.65±0.14	2.43±1.59	13.60±1.56	2.76±0.56
	0.995	3.06±0.33	10.63±4.84	28.59±1.39	9.51±0.94
	0.99	5.50±0.50	18.16±7.30	37.47±1.19	15.43±1.30
	0.95	19.13±1.01	49.24±8.68	62.67±1.41	39.30±1.68
	0.90	**22.90±1.50**	**55.76±8.28**	**66.93±1.48**	**44.47±2.29**

We repeat the experiment for adversarial data, instead allowing the algorithm to classify the correctness of the classification. These results are shown in table 8. We find the base model could have been better optimised on the target dataset, however we find similar findings to that in the main paper. Detection performance is maximised when the adversarial perturbation is maximised.

Table 8. Adversarial detection results for selected thresholds. We show results for a variety of relative ID accuracies. We take the mean and standard deviation over 3 runs, the ID data the model is trained on is CIFAR100.

		Adversarial Detection Performance (%)			
Relative ID	Threshold	ϵ			
Performance (%)	percentile	0.0	0.1	0.2	0.3
Zero Loss	1.0	72.87±0.24	14.19±0.62	6.63±0.60	7.15±2.06
99	0.999	72.98±0.23	14.87±0.60	12.42±2.43	20.66±6.96
	0.995	73.26±0.22	17.18±0.71	26.17±6.13	43.53±9.88
	0.99	73.45±0.23	18.43±0.96	**31.35±7.32**	**51.04±11.35**
	0.95	**73.49±0.27**	**18.77±0.76**	29.59±4.96	44.34±7.79
	0.90	**73.49±0.25**	18.54±0.72	23.56±3.14	31.65±4.63
95	0.999	73.03±0.24	15.27±0.67	15.38±2.89	26.69±8.09
	0.995	73.56±0.25	19.25±0.90	34.59±6.41	55.52±10.26
	0.99	74.08±0.26	23.06±1.12	46.66±7.15	68.79±9.62
	0.95	**75.90±0.40**	**34.21±1.94**	**66.83±7.91**	**85.62±6.96**
	0.90	75.85±0.41	34.02±1.86	66.57±7.60	85.36±6.71
90	0.999	73.05±0.23	15.45±0.73	16.67±2.89	28.61±8.22
	0.995	73.67±0.27	20.18±1.06	37.70±6.76	59.35±10.52
	0.99	74.34±0.23	24.70±1.27	50.44±7.52	72.58±9.07
	0.95	77.58±0.39	43.63±2.34	77.03±6.65	91.72±4.52
	0.90	**78.52±0.41**	**47.71±2.72**	**80.25±6.40**	**93.30±3.96**

We record additional peak improvements, with CIFAR100 as the ID dataset, we show this in table 9. The improvements are lower when CIFAR100 is the ID dataset, however, we still find ∼20% power can be saved, or ∼15% accuracy improvements can be made. We generally found the more liberal exit policies performed slightly better in this experiment.

We take the same readings for the adversarial tests using the CIFAR100 dataset, these are shown in table 10. We again find that the more liberal exit policies perform better on this dataset. We find that a ∼24% accuracy improvement can be made over conventional exiting algorithms, or a ∼29% power improvement.

B Reproducibility

B.1 Source Code

For reproducibility, our source code along with trained weights for CIFAR10 is made publicly available. It is available at: `github.com/J-Dymond/distribution-aware-exiting`

B.2 Datasets

The experiments have used a number of Datasets: CIFAR10/100, DTD, SVHN, and tiny-imagenet. We use these datasets because they are benchmark datasets in the field, including the field of OOD detection [54], and are often a good representation of what can be achieved with larger datasets when given more resources. Furthermore, they are all open-source, easy to work with using `torchvision`, and referenced properly in the main body of the paper.

When generating our training sets we use the standard normalisation factors for CIFAR10:

`transforms.Normalize(mean = [0.4914, 0.4822, 0.4465], std=[0.2023, 0.1994, 0.2010]).`

For tiny-imagenet we use the ImageNet normalisation terms:

`transforms.Normalize(mean = [0.485, 0.456, 0.406] , std = [0.229, 0.224, 0.225])`

For the other datasets we could not find the standard normalisation factors and hence used our CIFAR10 values.

We also apply random aumentations to encourage better generalisaion:

`transforms.RandomCrop(32, padding=4), transforms.RandomHorizontalFlip().`

Default values are used for these augmentations.

B.3 Random seeds

When generating our test, train, and validation sets we do use a random seed, due to the random nature of the train-validation split. We generate our random splits using: `torch.utils.data.random_split` and define our random seed using `torch.manual_seed(43)`, where `43` is the seed we use for this process. This is kept consistent for every dataset used in the paper.

For our experiments some factors use random values, for example: initial weights. We have tracked the seeds, and will release them with our codebase on successful publication.

B.4 Implementation Details

B.4.1 Hardware

Models were trained on a single GPU: Nvidia Quadro RTX 8000 or Nvidia V100, depending on computing cluster resource availability. However, the same experiments can be ran on more limited computational hardware.

The same hardware was used to generate the reference embedding file on the train set, used for nearest neighbour calculations. Memory issues may arise when generating these files as embeddings are highly dimensional. Thus, we calculated and saved the reference embedding file in batches, saving each batch individually. We then stacked them into a single file at the end of the script. Without large memory GPUs, this process may be more time consuming as a smaller batch size can be afforded (we use a batch size of `512` for this process). However, we believe this will not be prohibitively time consuming.

B.4.2 Hyperparameters

For CIFAR10/100 experiments we train our four branch model for 300 epochs, taking $\sim$3 hours the hardware used for this work. We perform a search for the best learning rate over the range $\mathtt{lr} \in \{\mathtt{0.5, 0.1, 0.05, 0.01, 0.005, 0.001}\}$, finding `0.01` to be the most successful.

Models are optimised using stochastic gradient descent, using the aforementioned learning rate, a momentum of `0.9` and a weight decay of `5e-4`. We also implement a learning rate scheduler, designed to drop the learning rate on plateau. That is, it lowers the learning rate when convergence stops. We use a `patience` value of `50`, that is it allows 50 epochs of no improvement before dropping the learning rate by a multi[plicative factor we chose: `0.1`.

We follow the findings in [13], and in equation 1 we use weightings of `[0.2,0.2,0.2,0.4]`.

Table 9. Peak improvements made by the early exiting algorithm on $\mathcal{D}$, for a number of different OOD datasets, when compared to conventional early exiting algorithm. We compare a number of different detection thresholds, peak accuracy and power improvements are shown. We highlight the best performing values for each dataset. We take the mean and standard deviation over 5 runs of the experiment, CIFAR100 is used as the ID dataset.

		Peak Improvement (%)			
Detection Threshold	Improvement	Dataset			
		CIFAR10	SVHN	DTD	Tiny-Imagenet
1.0	Acc	0.01±0.01	0.00±0.00	0.31±0.15	0.02±0.03
	Power	0.02±0.01	0.01±0.01	0.40±0.18	0.02±0.02
0.999	Acc	0.23±0.08	0.57±0.53	2.51±0.22	0.77±0.22
	Power	0.17±0.01	0.41±0.26	1.66±0.17	0.41±0.05
0.995	Acc	0.97±0.03	3.49±1.60	6.26±0.22	3.35±0.70
	Power	0.91±0.05	1.84±0.71	4.10±0.37	1.89±0.29
0.99	Acc	1.99±0.32	5.34±2.45	8.31±0.55	5.92±0.99
	Power	1.74±0.06	3.31±1.01	5.99±0.32	3.42±0.37
0.95	Acc	**14.74±1.94**	9.06±1.59	**12.54±0.88**	13.84±0.72
	Power	7.86±0.23	12.63±1.23	14.14±0.63	12.04±0.79
0.90	Acc	9.92±0.33	**16.76±2.16**	11.81±0.46	**14.81±0.49**
	Power	**22.55±1.88**	**25.30±0.63**	**20.64±0.69**	**19.73±0.91**

.

Table 10. Peak improvements made by the early exiting algorithm on $\mathcal{D}$, for a number of different ϵ values, when compared to conventional early exiting algorithms. We compare a number of different detection thresholds, peak accuracy and power improvements are shown. We take the mean and standard deviation over 5 runs, trained on CIFAR100 dataset.

		Peak Improvement (%)			
Detection Threshold	Improvement	ϵ			
		0.0	0.1	0.2	0.3
1.0	Acc	0.00±0.00	0.02±0.03	0.17±0.21	0.61±0.43
	Power	0.00±0.00	0.02±0.02	0.14±0.12	0.48±0.36
0.999	Acc	0.08±0.02	0.46±0.16	4.29±1.22	8.35±2.43
	Power	0.14±0.00	0.32±0.07	1.97±0.48	3.98±1.09
0.995	Acc	0.47±0.03	2.66±0.60	12.73±3.62	19.02±3.70
	Power	0.68±0.01	1.60±0.28	6.81±1.76	10.86±2.40
0.99	Acc	1.14±0.29	4.89±1.22	17.49±3.08	22.84±3.01
	Power	1.33±0.01	3.04±0.37	10.14±2.04	14.66±2.62
0.95	Acc	4.66±0.62	13.00±1.22	**21.73±1.73**	**23.67±0.81**
	Power	6.23±0.05	11.57±1.06	20.21±2.71	23.40±2.87
0.90	Acc	**6.47±0.04**	**14.47±0.49**	17.47±0.88	17.53±1.52
	Power	**11.88±0.09**	**19.31±1.18**	**25.91±2.31**	**27.95±2.29**

Table 11. Table showing the AUROC for each branch given different perturbation values for adversarial data. The best performing branch for each dataset is highlighted for each value of ϵ. We omit the results for ϵ=0, since the data is unperturbed in this range. We use the benchmark results reported in [56].

Branch	AUROC								
					ϵ				
	0.03	0.07	0.1	0.13	0.17	0.2	0.23	0.27	0.3
1	0.57±0.00	0.66±0.01	0.74±0.02	**0.82±0.03**	**0.87±0.03**	**0.90±0.03**	**0.93±0.02**	**0.95±0.02**	**0.96±0.01**
2	0.58±0.00	0.64±0.01	0.71±0.01	0.77±0.02	0.83±0.02	0.87±0.02	0.90±0.02	0.92±0.02	0.93±0.03
3	**0.63±0.01**	0.70±0.01	**0.76±0.01**	0.80±0.01	0.85±0.01	0.88±0.01	0.90±0.02	0.91±0.03	0.91±0.05
4	**0.63±0.01**	**0.71±0.01**	**0.76±0.01**	0.80±0.01	0.83±0.01	0.85±0.01	0.87±0.02	0.88±0.03	0.88±0.04
SPBAS [28]	-	-	-	-	-	-	-	-	0.79
ML + LOO [55]	-	-	-	-	-	-	-	-	**1.00**
KD + BU [14]	-	-	-	-	-	-	-	-	0.90
LID [34]	-	-	-	-	-	-	-	-	0.95
MAHA [33]	-	-	-	-	-	-	-	-	**1.00**

Table 12. Table showing the AUROC for each branch given different perturbation values for adversarial data, with a model trained on CIFAR100. The best performing branch for each dataset is highlighted for each value of ϵ. We omit the results for ϵ=0, since the data is unperturbed in this range. We use the benchmark results reported in [56].

Branch	AUROC								
					ϵ				
	0.03	0.07	0.1	0.13	0.17	0.2	0.23	0.27	0.3
1	0.54±0.01	0.57±0.01	0.58±0.01	0.56±0.03	0.54±0.01	0.53±0.02	0.53±0.02	0.52±0.03	0.53±0.04
2	0.58±0.00	**0.69**±0.01	**0.77±0.02**	**0.84±0.03**	**0.88±0.03**	0.91±0.04	0.93±0.04	0.94±0.04	0.95±0.04
3	0.58±0.01	0.67±0.01	0.75±0.01	0.82±0.01	**0.88±0.01**	**0.92±0.01**	**0.95±0.00**	**0.96±0.00**	**0.97±0.00**
4	**0.59±0.01**	0.68±0.01	0.75±0.01	0.81±0.01	0.85±0.01	0.88±0.01	0.91±0.02	0.92±0.02	0.93±0.02
SPBAS [28]	-	-	-	-	-	-	-	-	0.51
ML + LOO [55]	-	-	-	-	-	-	-	-	**1.00**
KD + BU [14]	-	-	-	-	-	-	-	-	0.98
LID [34]	-	-	-	-	-	-	-	-	1.00
MAHA [33]	-	-	-	-	-	-	-	-	**1.00**