EBJR: Energy-Based Joint Reasoning for Adaptive Inference

Mohammad Akbari
mohammad.akbari@huawei.com
Amin Banitalebi-Dehkordi
amin.banitalebi@huawei.com
Yong Zhang
yong.zhang3@huawei.com

Huawei Technologies Canada Co., Ltd.

Abstract

State-of-the-art deep learning models have achieved significant performance levels on various benchmarks. However, the excellent performance comes at a cost of inefficient computational cost. Light-weight architectures, on the other hand, achieve moderate accuracies, but at a much more desirable latency. This paper presents a new method of jointly using the large accurate models together with the small fast ones. To this end, we propose an Energy-Based Joint Reasoning (EBJR) framework that adaptively distributes the samples between shallow and deep models to achieve an accuracy close to the deep model, but latency close to the shallow one. Our method is applicable to out-of-the-box pre-trained models as it does not require an architecture change nor re-training. Moreover, it is easy to use and deploy, especially for cloud services. Through a comprehensive set of experiments on different down-stream tasks, we show that our method outperforms strong state-of-the-art approaches with a considerable margin. In addition, we propose specialized EBJR, an extension of our method where we create a smaller specialized side model that performs the target task only partially, but yields an even higher accuracy and faster inference. We verify the strengths of our methods with both theoretical and experimental evaluations. Code and demo are available here.

1 Introduction

Recent years have witnessed exciting achievements in the development of highly capable deep neural networks (DNNs), to the extent that new state-of-the-art (SOTA) results are being published frequently. However, achieving this level of performance requires either using extremely large architectures such as GPT-3 [3] or SEER [11] with billions of parameters (350GB memory and 175B parameters in GPT-3), or ensembling many models. Consequently, this results in an inefficient inference compared to light-weight models.

To alleviate the problem of slow inference on large architectures, a natural solution is to apply some form of model compression. Model compression literature is rich and mature, and covers various techniques such as network quantization [4, 8, 20, 21, 38], knowledge distillation [16, 48], pruning [5, 10, 15, 23], or a combination of multiple techniques [6, 14, 31]. After compression, the output DNN may have a reduced number of parameters or may operate in lower bit precision. However, it can be observed that there is a trade-off between the compression ratio and accuracy of a model. Aggressive compression leads to significant performance drop that defeats the purpose. Moreover, compression is one solution for all data and is inference-time deterministic (static) with no flexibility over different

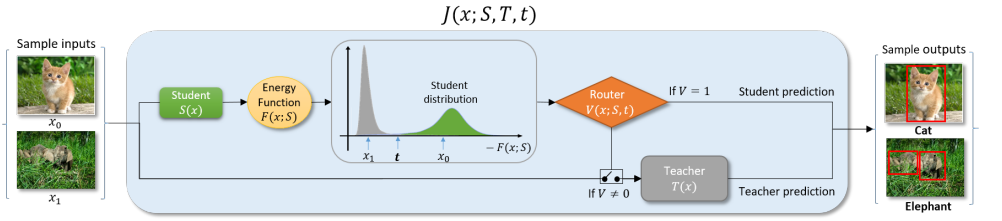

Figure 1: The overall flow-diagram of the proposed energy-based joint reasoning (EBJR) method.

data samples. That being said, compression techniques are commonly orthogonal to other approaches in that a degree of compression can be added to other methods.

On the other hand, adaptive inference approaches propose to route to different branches of a DNN either stochastically, or based on some decision criteria on input data [18, 36, 40, 41, 42, 43, 44, 45]. These methods mostly are based on architecture re-design, i.e., the model needs to be built in a specific way to support dynamic inference. This makes their training more complex and imposes additional non-trivial hyper-parameter tuning. Adaptive inference methods can broadly be categorized as redundancy-based and multi-exit structures. The redundancy-based approaches exploit the parameter redundancy in neural networks. To this end, [7] designed a convolution-based controller layer, which reduces the computations in practice, even though increases the overall network size. Or [9, 27, 32, 37, 39, 41] dynamically skip some layers or blocks on the fly via selective layer execution.

Multi-exit or multi-stage approaches, however, are based on architectures in which a network can exit from different paths based on some confidence criteria. Earlier techniques such as BranchyNet [36] incorporated an entropy-based threshold for routing. A similar approach was taken by [2, 29] by training side classifiers to navigate to different paths. [18] proposed a multi-scale dense network to reuse feature maps of different scales, which was further improved in [42] by designing a resolution adaptive network (RANet) to identify low resolution inputs as easy cases, and process them with cheaper computations. There are also works based on architecture search for dynamic inference models [46]. It is also worth noting that the majority of the existing methods focus on the task of image classification and fail to study the other applications. [47] is an example were adaptive inference was investigated for the task of object detection, by leveraging a Support Vector Machine (SVM) classifier to route the work-load. A down-side for [47], however, is that dynamically changing the routing traffic between the fast and slow branches requires retraining.

Although the redundancy-based and multi-exit methods have made a significant progress and work well in practice, we will show that they do not reach the levels of performance provided by our energy-based strategy. In addition, most of these methods require training models in a specific way necessitated by their architecture design. In contrast, our method works with out-of-the-box already trained models without a need for re-training.

In this paper, we propose an adaptive inference strategy that combines a large, deep accurate model (called Teacher) with a small, shallow fast one (called Student). Our method is based on an effective energy-based decision making module for routing different samples to deep or shallow models. In this way, certain examples will be sent to the Student model that yields high speed inference, and other examples go to the Teacher model, which is slower, but highly accurate. Our method provides an inference-time trade-off between the inference latency and task accuracy. This can be thought of as a knob for the users to play with, and to dynamically choose a desired point in the trade-off based on their required accuracy or latency. Figure 1 shows a high-level schematic of the proposed framework.

In addition to our main adaptive inference strategy, we provide an extension called specialized EBJR, which provides more accurate and efficient inference by training the Student in a way that it only learns to perform the down-stream task partially (details in Section 2.4).

The main contributions of this paper are summarized as follows:

- Combining small/shallow models (low accuracy and latency) with large/deep models (high accuracy and latency) to achieve **high accuracy and low latency**. Our method is easy to build and deploy, is architecture agnostic, applicable to different downstream tasks (e.g., classification and object detection), and can be applied to existing pre-trained models (with no need for re-training).
- An **energy-based routing mechanism** for directing examples to the small (Student) or large (Teacher) models. This allows a dynamic trade-off between accuracy and computational cost that outperforms the previous works in adaptive inference (with zero overhead for real-time adjustment of speed/accuracy).
- Creating a small, Student model **specialized** for a subset of tasks (e.g., top-C classes only) with high accuracy; along with a plus-one (+1) mechanism, to distinguish the top-C-class data from the others.

2 Energy-Based Joint Reasoning (EBJR)

We introduce EBJR, a novel energy-based joint reasoning approach for adaptive inference. Our method is inspired by the fact that smaller (shallower/narrower) models typically have lower accuracy, but very fast inference; and larger (deeper/wider) models, on the other hand, are highly accurate, but very slow. We combine the small model (denoted by Student) and the large model (denoted by Teacher) in an efficient and effective way to provide a fast inference, while maintaining the high accuracy. A schematic of our framework is shown in Figure 1.

The main challenge here is to design an effective routing mechanism (denoted by Router) to decide which model to use for each input. As for the adaptive inference, the Router should also provide the option of dynamic trade-offs between accuracy and latency at the inference time. The Router module essentially operates similar to a binary classifier that directs easy samples to the Student and the hard ones to the Teacher. In some ways, this problem is also similar to the out-of-distribution detection (OOD) problem [28] in which in- and out-of-distribution data are differentiated. OOD is generally used when a model sees some input test data that differs from its training data (in-distribution data). Consequently, the predictions of the model on OOD samples would be unreliable. For our case, the Router should be able to identify whether or not the input data fits in the distribution with which the Student has been trained (i.e., there is a high probability that the Student can make accurate predictions for that input data). If not, the data is labelled as hard for the Student and should be forwarded to the Teacher that has higher capability. In our work, we investigate the energy characteristics of data samples to route them effectively.

Energy definitions. Given an input data point $\mathbf{x}$, the energy function is defined as $E(\mathbf{x}) : \mathbb{R}^{-D} \rightarrow \mathbb{R}$ to map the input $\mathbf{x}$ to a scalar, non-probabilistic energy value y. The probability distribution over a collection of energy values can be defined according to the Gibbs distribution [17, 24]: $p(y|\mathbf{x}) = \frac{1}{Z}\left(e^{-E(\mathbf{x},y)}\right)$, where $Z(\mathbf{x}) = \int_{y'} e^{-E(\mathbf{x},y')}$ is the partition function. The free energy [24] of $\mathbf{x}$ can then be expressed as the negative log of the partition function:

$$F(\mathbf{x}) = -\log\left(Z(\mathbf{x})\right). \tag{1}$$

In the following subsections, we will describe our energy-based joint reasoning method, and give formulations for classification, regression, and object detection problems.

2.1 Classification

The Student classifier is defined as a function $S^c(\cdot)$ for mapping the input $\mathbf{x}$ to C real-valued logits (i.e., for C number of class labels): $S^c(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^C$. In probability theory, we can use the output of the softmax function to represent a categorical distribution that is a probability

distribution over C different possible outcomes [28]. A categorical distribution using the softmax function is expressed by:

$$p(y|\mathbf{x}) = \frac{e^{S^c_y(\mathbf{x})}}{\sum_i^C e^{S^c_i(\mathbf{x})}}, \tag{2}$$

where $S^c_y(\mathbf{x})$ denotes the logit (probability) of the yth class label. The energy for a given input $(\mathbf{x}, y)$ in this case is defined as $E(\mathbf{x}, y) = -S^c_y(\mathbf{x})$ [28]. The free energy function $F^c(\mathbf{x}; S^c)$ is then expressed similar to (1) as:

$$F^c(\mathbf{x}; S^c) = -\log \sum_i^C e^{S^c_i(\mathbf{x})}. \tag{3}$$

Problem. We seek to identify samples suitable for the Student and would like to direct the others to the Teacher. A natural solution to this problem is to use the data density function and consider the inputs with low likelihood as hard (or unfit) samples. To this end, an energy-based density function for the Student can be defined as:

$$p(\mathbf{x}) = \frac{1}{Z^c}\left(e^{-F^c(\mathbf{x}; S^c)}\right), \tag{4}$$

where the denominator $Z^c = \int_{\mathbf{x}} e^{-F^c(\mathbf{x}; S^c)}$ is the normalized densities, which can be intractable to compute or estimate. By taking the logarithm of both sides, we obtain:

$$\log\left(p(\mathbf{x})\right) = -F^c(\mathbf{x}; S^c) - \log(Z^c). \tag{5}$$

Solution. The $\log(Z^c)$ term is constant for all $\mathbf{x}$, and does not affect the overall energy values distribution. Thus, the negative free energy is linearly aligned with the log likelihood function. This makes it a suitable solution to our problem for detecting easy and hard samples. In this case, higher energy means lower likelihood, which represents harder (or more unfit) samples for the Student's training distribution.

More precisely, for a threshold δ on the density function such that $p(\mathbf{x}) < \delta$, then a threshold t on the negative free energy can be calculated based on (5) as $-F^c(\mathbf{x}; S^c) < t = \log(\delta Z^c)$. In practice, for a given input, an energy function is applied to the Student outputs at inference to compute the energy score. Then, if the negative energy value is smaller than a threshold, the input is identified as a bad sample for the Student, and is sent to the Teacher.

Therefore, given the input data $\mathbf{x}$, the Student $S^c(\mathbf{x})$, and the threshold t, our energy-based Router $V^c(\mathbf{x}; S^c, t) \in \{0, 1\}$ can simply be defined as:

$$V^c(\mathbf{x}; S^c, t) = \begin{cases} 1 & \text{if } -F^c(\mathbf{x}; S^c) \geq t \\ 0 & \text{if } -F^c(\mathbf{x}; S^c) < t. \end{cases} \tag{6}$$

Let the Teacher classifier be $T^c(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^{C'}$, with $C = C'$ (the same number of class labels as in the Student). Our joint reasoning classification function $J^c(\mathbf{x}; S^c, T^c, t) \in [1, C]$ can then be written by:

$$J^c(\mathbf{x}; S^c, T^c, t) = \begin{cases} S^c(\mathbf{x}) & \text{if } V^c(\mathbf{x}; S^c, t) = 1 \\ T^c(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{7}$$

2.2 Regression

A regressor maps an input $\mathbf{x}$ to a target scalar y defined as $S^r(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$. For a given input $(\mathbf{x}, y)$, the energy function for a regressor is simply defined as $E(\mathbf{x}, y) = -S^r(\mathbf{x}, y)$. The regression problem can then be expressed by creating an energy-based model of the conditional density $p(y|\mathbf{x})$ as:

$$p(y|\mathbf{x}; S^r) = \frac{e^{S^r(\mathbf{x}, y)}}{\int_{y'} e^{S^r(\mathbf{x}, y')}}, \tag{8}$$

where the denominator is the normalizing partition function that involves a computationally intractable integral. One solution is to obtain its approximations using Monte Carlo importance sampling method as described in [13]. The free energy in this case is defined by:

$$F^r(\mathbf{x};S^r) = -\log\left(\int_{y'} e^{S^r(\mathbf{x},y')}\right). \tag{9}$$

Similar to (4), the density function for a regressor using the energy-based model can be obtained as follows:

$$p(\mathbf{x}) = \frac{1}{Z^r}\left(e^{-F^r(\mathbf{x};S^r)}\right), \tag{10}$$

where the denominator is the normalized densities defined as $Z^r = \int_{\mathbf{x}} e^{-F^r(\mathbf{x};S^r)}$. By taking the log of both sides:

$$\log\left(p(\mathbf{x})\right) = -F^r(\mathbf{x};S^r) - \log(Z^r), \tag{11}$$

which, as in the classification problem, shows that $-F^r(\mathbf{x};S^r)$ has a linear alignment with the log likelihood function by considering the fact that $\log(Z^r)$ is constant for all $\mathbf{x}$, which makes it desirable for our problem.

Given the input data $\mathbf{x}$, the Student regression model $S^r(\mathbf{x})$, and a threshold t, our energy-based Router for a regression problem can be simply defined with $V^r(\mathbf{x};S^r,t) \in \{0,1\}$ based on (6). The joint reasoning function $J^r(\mathbf{x};S^r,T^r,t) \in \mathbb{R}$ can also be expressed based on (7), where $T^r(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$ is the Teacher regression model.

2.3 Object detection

For the object detection task with a combination of classification and regression, we can define the total free energy score as: $F^o(\mathbf{x};S^c,S^r) = F^c(\mathbf{x};S^c) + F^r(\mathbf{x};S^r)$, where the regressor for predicting 4 points of a bounding box is defined as $S^r(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^4$. With B number of detected boxes and C labels, the classifier's free energy score $F^c(\mathbf{x};S^c)$ is formulated as:

$$F^c(\mathbf{x};S^c) = \frac{1}{B}\left(-\sum_{b}^{B}\log\sum_{i}^{C} e^{S^c_{b,i}(\mathbf{x})}\right), \tag{12}$$

where $S^c_{b,i}$ is the classifier's output for the ith class label of the bth bounding box with $b \in [1,B]$ and $i \in [1,C]$. The regressor's free energy $F^r(\mathbf{x};S^r)$ is also given by:

$$F^r(\mathbf{x};S^r) = \frac{1}{4B}\left(-\sum_{b}^{B}\sum_{j}^{4}\log\int_{y'} e^{S^r_{b,j}(\mathbf{x},y')}\right), \tag{13}$$

where $S^r_{b,j}$ is the regression output for the jth point of the bth bounding box with $j \in [1,4]$.

The energy-based joint reasoning function for object detection task is finally defined as:

$$J^o(\mathbf{x};S^o,T^o,t) = \begin{cases} T^o(\mathbf{x}) & \text{if } -F^o(\mathbf{x};S^o) < t \\ S^o(\mathbf{x}) & \text{if } -F^o(\mathbf{x};S^o) \geq t, \end{cases} \tag{14}$$

where $S^o = \{S^c,S^r\}$ and $T^o = \{T^c,T^r\}$ denote the Student and Teacher object detection models.

2.4 Specialized EBJR

In Section 2.1, it was assumed that the Student and Teacher models have equal number of classes that is $C = C'$. As proved in [1], in order to achieve a good performance for a classifier with large number of classes, significantly large number of features are required. Since the Teacher model is assumed to be a very large model with significant number of features, it is capable of handling more difficult tasks with a large C'. On the other hand, the small Student model may lack enough features to be able to effectively deal with a large C.

In addition, in inference services such as public clouds, the majority of input data usually belongs to a small, popular subset of classes that are used frequently, for example,"people",

"cat", "dog", "car", etc (supplementary materials contain example-per-class histogram plots for public datasets, and confirms this intuition). Considering this fact, the Student can be trained and specialized to be highly accurate on this specific/popular subset (with a small C). Consequently, in our joint reasoning scheme, most of the input data can be handled by the Student in a very accurate and computationally efficient way.

Let the specialized Student be $\bar{S}^c(x) : \mathbb{R}^D \rightarrow \mathbb{R}^{\bar{C}+1}$, where $\bar{C} \ll C$. To make sure the model can still exploit and learn from all data at the training time, we label the data that do not belong to $\bar{C}$ as an additional class (i.e., the 'other' class). The extra class is also utilized as a supplementary mechanism in our Router to evaluate the performance of $\bar{S}^c$ on a given input data at the inference time. Similar to a binary classifier, it is used for distinguishing the data with $\bar{C}$ labels from the others.

The specialized Student $\bar{S}^c$ has another benefit for our energy-based Router. Since only a subset of class labels is used for training the Student, the energy differences between in- and out-of-distribution data respectively denoted by $(\mathbf{x}, i)$ and $(\bar{\mathbf{x}}, j)$ tends to be larger:

$$|\bar{F}^c(\mathbf{x};\bar{S}^c_i) - \bar{F}^c(\bar{\mathbf{x}};\bar{S}^c_j)| > |F^c(\mathbf{x};S^c_i) - F^c(\bar{\mathbf{x}};S^c_j)|, \tag{15}$$

where $i \in [1,\bar{C}]$ and $j \notin [1,\bar{C}]$. The larger the energy difference, the better the Router can distinguish the fit and unfit data for the Student, which results in more accurate and efficient adaptive inference. Given the input data $\mathbf{x}$, the specialized Student $\bar{S}^c(\mathbf{x})$, and a threshold t, our specialized energy-based Router $\bar{V}(\mathbf{x};\bar{S}^c,t) \in \{0,1\}$ is expressed as:

$$\bar{V}(\mathbf{x};\bar{S}^c,t) = \begin{cases} 1 & \text{if } -\bar{F}^c(\mathbf{x};\bar{S}^c) \geq t \quad \text{and} \quad \bar{S}^c(\mathbf{x}) \in [1,\bar{C}] \\ 0 & \text{if } -\bar{F}^c(\mathbf{x};\bar{S}^c) < t \quad \text{or} \quad \bar{S}^c(\mathbf{x}) \in \{\bar{C}+1\}, \end{cases} \tag{16}$$

where $\bar{C}+1$ denotes the extra class defined in $\bar{S}^c$. The free energy $\bar{F}^c(\mathbf{x};\bar{S}^c)$ for the specialized Student is calculated only over the top-$\bar{C}$ classes, not the extra class, as follows:

$$\bar{F}^c(\mathbf{x};\bar{S}^c) = -\log \sum_{i}^{\bar{C}} e^{\bar{S}^c_i(\mathbf{x})} \quad \text{with} \quad i \notin \{\bar{C}+1\}. \tag{17}$$

Let the Teacher be $T^c(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^{C'}$ with $\bar{C} \ll C'$. Then, the specialized joint reasoning function $\bar{J}(\mathbf{x};\bar{S}^c,T^c,t) \in [1,C']$ for making the predictions related to $\mathbf{x}$ can be given by:

$$\bar{J}(\mathbf{x};\bar{S}^c,T^c,t) = \begin{cases} \bar{S}^c(\mathbf{x}) & \text{if } \bar{V}(\mathbf{x};\bar{S},t) = 1 \\ T^c(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{18}$$

3 Experiments

In this section, we evaluate and discuss the performance of our EBJR approach along with the other related methods on image classification and object detection tasks on different benchmarks. We provide more results and ablation studies in the supplementary materials.

3.1 Adaptive inference results

Figures 2 and 3 show the classification results for EBJR and the SOTA in adaptive inference on CIFAR-10, CIFAR-100, ImageNet, and Caltech-256 [17] datasets. We use multiple datasets not only to evaluate the generality of our method, but also because not all other methods published results on a single standard dataset. For all the datasets, we use DenseNet models [19] for our Student and Teacher, except Caltech-256 for which ResNet models are used. Table 1 shows and compares the details about the Student, Teacher, and EBJR models and their accuracy, floating point operations (FLOPs), and average inference time (latency).

Note that many previous approaches are based on the DenseNet architecture, and then adaptively dropping connections for inference speed-up. Thus, we also choose DenseNet as the main architecture to establish a fair comparison although our method does not rely on any

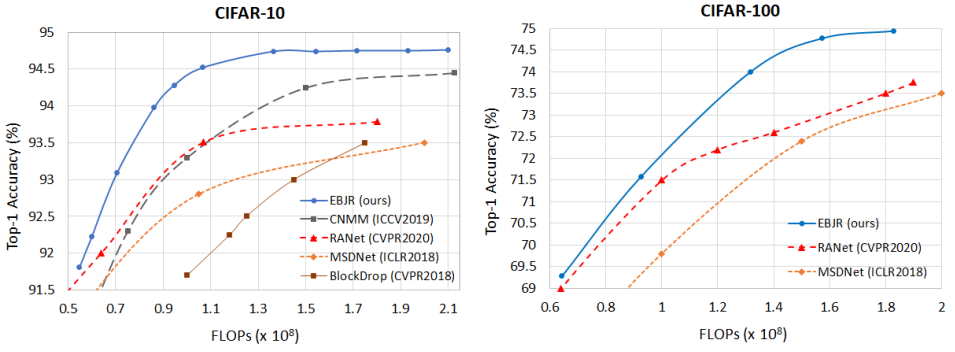

Figure 2: Evaluation of EBJR and the SOTA in adaptive inference on CIFAR datasets.

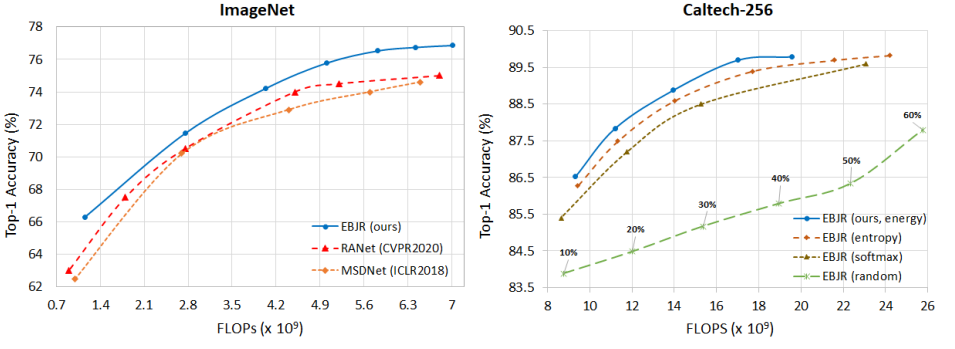

Figure 3: Evaluation of EBJR on the ImageNet (left) and Caltech-256 (right) datasets. The numbers on the EBJR (random) curve show the percentage of samples processed by the Teacher.

specific network design and can work with any black-box architectures. Moreover, we follow the standard practice in the previous works and analyze the results with FLOPs [18, 32, 41, 42]. For our method, the total FLOPs is measured as a weighted average of the Teacher and Student FLOPs based on their usage frequency as: $FLOPs = \frac{1}{N_S+N_T}\big(N_S.F_S + N_T.(F_S+F_T)\big)$, where N_S and N_T are respectively the number of samples processed by Student (with F_S FLOPs) and Teacher (with F_T FLOPs). Note that the metric used in [18, 32, 42] is multiply-accumulates (MACs), i.e., half the FLOPs used in this work.

In Figures 2 and 3, the trade-off between accuracy and computational cost is adaptively achieved in our method by choosing different values for the threshold parameter t as defined in (6) and (7). The larger the threshold, the more input data are routed to the Teacher model, which results in more accurate, but slower inference. As the Student is able to make accurate predictions for the majority of input data, the adaptive inference with an appropriate small enough t can almost reach the Teacher's accuracy but with a much lower computational cost. For CIFAR-10, this strategy achieves the Teacher's accuracy with $\approx$2.2$\times$ less FLOPs. It can also lead to approximately 3$\times$ less FLOPs with an accuracy of $\approx$94.5% (i.e., only $\approx$0.2% lower than the Teacher). The amount of speed-up for Caltech-256 is about 2$\times$, while maintaining the Teacher's accuracy of 89.87%. For CIFAR-100 and ImageNet, which are more complicated benchmarks, Teacher's top-1 accuracy is almost achieved with approximately 1.5$\times$ savings on the computations. Moreover, as illustrated in Figures 2 and 3, our method outperforms the previous works such as RANet [42] and MSDNet [18] on all the three benchmarks across a variety of accuracy and cost combinations.

To investigate the performance of energy-based routing mechanism compared to other alternatives, we perform an ablation study on Caltech-256, where the energy score is replaced

	CIFAR-10			CIFAR-100			ImageNet			Caltech-256		
	S	**T**	**EBJR**	**S**	**T**	**EBJR**	**S**	**T**	**EBJR**	**S**	**T**	**EBJR**
Depth	52	64	-	58	88	-	121	201	-	18	152	-
Growth Rate	6	12	-	6	8	-	12	32	-	-	-	-
Accuracy (%)	91.81	94.76	**94.74**	69.28	74.94	**74.87**	66.28	76.92	**76.62**	83.16	89.87	**89.87**
FLOPs ($\times 10^8$)	0.54	2.92	**1.36**	0.64	2.14	**1.57**	11.51	86.37	**58.1**	54.0	340.0	**170.1**
Latency (ms)	14.0	35.0	**23.78**	26.0	51.0	**42.1**	84.0	225.0	**196.8**	25.0	200.0	**113.6**

Table 1: Comparison of EBJR with the Student (**S**) and Teacher (**T**) DensetNet models for CIFAR-10, CIFAR-100, and ImageNet; and ResNet models for Caltech-256 experiments.

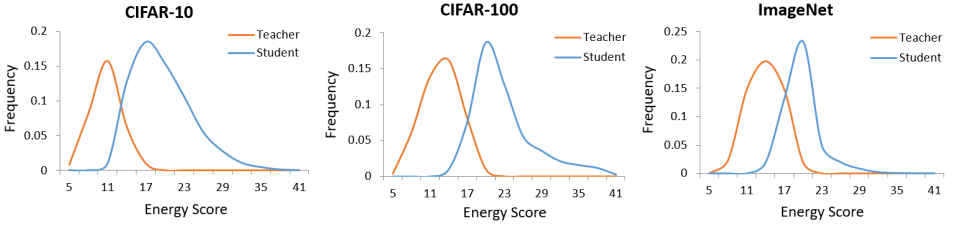

Figure 4: Energy score distribution for CIFAR-10, CIFAR-100, and ImageNet.

by the softmax confidence or entropy [36] scores. We also include the random baseline in this experiment, where the input samples are randomly distributed between the Student and Teacher models (the experiment was run multiple times and the best of them was reported). The corresponding adaptive inference results are presented in Figure 3-right. It is observed that softmax- and entropy-based mechanisms can reach the Teacher's accuracy with $\approx$1.4$\times$ and $\approx$1.7$\times$ less FLOPs, which is lower than the energy-based strategy with 2$\times$ speed-up. The theoretical analysis for the entropy score will be given in the supplementary materials.

Figure 4 illustrates the energy score distribution for the samples processed by the Student (i.e., in-distribution data) and Teacher (i.e., out-of-distribution data). As observed, the in-distribution samples (suitable for the Student) tend to have higher energy scores. Based on our experiments, the optimal setup for EBJR is achieved by choosing the threshold t at the crossing point of the two distributions. As a consequence, by choosing t=12.0 for CIFAR-10, $\approx$70% of the samples are handled by the Student with an accuracy of 99.0%, and only $\approx$30% are routed to the Teacher, which results in $\approx$3X less total FLOPs. For CIFAR-100 (with t=15.0) and ImageNet (with t=15.5), $\approx$50% are processed by the Student (with an accuracy of $\approx$91.0%), which achieve about 1.5X less FLOPs.

Power-Accuracy Tradeoff. In the literature, there are also some adaptive inference methods that are proposed for efficient power-accuracy trade-off, for example, [35] and BL-Net [30]. In order to compare EBJR with these approaches, we use the strategy in [25] to calculate the power (or energy) consumption per image. As summarized in Table 2, the method in [35] reduces the power consumption by 58.74% with 0.96% accuracy loss on CIFAR-10, while EBJR (Figure 2) achieves 64.03% power savings without any accuracy loss. Moreover, BL-Net [30] achieves 53.7% reduction in power consumption with an accuracy loss of 0.9% on ImageNet. EBJR (Figure 5), on the other hand, provides a reduction of 56.63% in power consumption with the same accuracy drop. Unlike BL-Net that does not reach the big model's accuracy, our method achieves the Teacher's accuracy with 32.93% less power consumption.

	CIFAR-10		ImageNet	
	EBJR	[35]	**EBJR**	**BL-Net** [30]
Accuracy loss (%)	**0.0**	0.96	**0.9** (0.0)	0.9
Power savings (%)	**64.03**	58.74	**56.63** (32.93)	53.7

Table 2: Power consumption vs. accuracy comparison.

MobileNetV2-Based EBJR. In addition to DenseNet, there exist some SOTA that are based on other architectures such as MobileNetV2 [33], for example, S-Net [45], US-Net [44], Mutual-Net [43], and RS-Net [40]. In order to compare EBJR with these approaches, we run another set of experiments on ImageNet, where MobileNetV2 models with 128$\times$128 and 224$\times$224 input resolutions are respectively used as our Student and Teacher. As shown

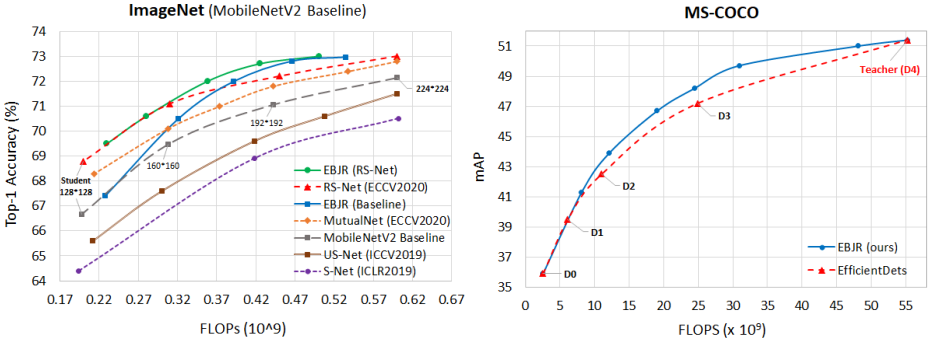

Figure 5: **(Left)** Evaluation of MobileNetV2-based EBJR with previous works on ImageNet. **(Right)** The performance of EBJR for object detection on MS-COCO (compared with EfficientDet [34]).

in Figure 5-Left, EBJR achieves better performance than S-Net, US-Net, and MutualNet across all FLOPs, and also better than RS-Net in high FLOPs. RS-Net provides better results than EBJR in low FLOPs, which is due to the less accurate Student used in EBJR. However, when EBJR and RS-Net are integrated and the RS-Net's 128×128 path is employed as the Student, the results are improved and EBJR outperforms RS-Net at all trade-off points.

Significance Test. In order to evaluate the statistical significance of the results, we perform the McNemar's test [6] over EBJR and SOTA including RANet and RS-Net. The McNemar's test is interpreted based on a given significance level α (commonly set to 0.05 showing 95% confidence) as well as the p-value and odds ratio calculated by the test. The default assumption (null hypothesis), i.e., if $p > \alpha$, states that the two classifiers should have the same error rate or there should be no difference in the disagreements between them. However, if null hypothesis is rejected, i.e., if $p \leq \alpha$, it suggests that the two classifiers disagree in different ways. After running the test over the EBJR vs. RANet predictions on CIFAR-10 and the EBJR vs. RS-Net predictions on ImageNet , p-values of 3.2×10^{-5} and 2.4×10^{-4} are respectively obtained. The very low p-values ($\ll 0.05$), which reject the null hypothesis, strongly confirms that there is a significant difference in the disagreements between EBJR and other two models. Also, an odds ratio of 1.42 and 1.14 is respectively obtained, which gives an estimation of how much better EBJR is compared to RANet and RS-Net.

Unlike image classification, the adaptive inference for the object detection task has rarely been explored. We analyze the performance of EBJR on the task of object detection (formulated and described in Section 2.3) on the MS-COCO dataset [26]. We employ the EfficientDet-D0 and EfficientDet-D4 [34] as the Student and Teacher, respectively. Figure 5-Right shows the adaptive inference results compared to the EfficientDet models (D0, D1, D2, D3, and D4). As shown in the figure, EBJR outperforms the standard EfficientDet models, where it reaches 97% of the Teacher's mAP on MS-COCO with $1.8\times$ speed-up. For the same mAP level, the adaptive feeding method of [47] reports only $1.3\times$ speed-up.

3.2 Specialized EBJR

In Section 2.4, we argued that creating a specialized Student targeted to handle only the popular categories can make the joint inference more efficient. To study this case we run a set of experiments on a subset of the Open Images dataset (OID) [23] that has been labeled using the 256 class labels of Caltech-256 dataset. We train the Student with 20% of the class labels (i.e. $\bar{C}$=50 out of 256 labels) along with an extra one reserved for the other classes. In this setup, we choose the top-50 class labels with the most number of samples in OID training set. For testing, we randomly select a new set of size 3K from the OID validation set, where 75% of the data have the top-50 of the labels. This is done to ensure the initial

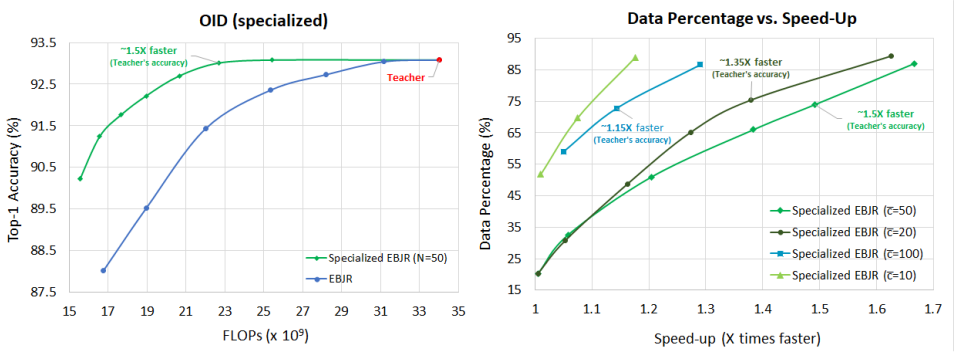

Figure 6: The performance of the specialized EBJR on OID validation set. (left) The specialized EBJR with $\bar{C} = 50$ compared with the general case. (right) The impact of different input data percentages with different chosen subsets of classes for specialized EBJR.

assumption of 'having the majority of samples from the popular classes' remains valid.

Figure 6-left shows the results of this experiment. We see that compared to the general cases of EBJR, the specialized EBJR provides the best performance under the assumption that the majority of input data belong to a small subset of classes. For example, compared to the Teacher, the specialized EBJR achieves $\approx 1.5\times$ less FLOPs with the same accuracy.

Figure 6-right shows the effect of the percentage of data that belong to the top-$\bar{C}$ classes. As expected, the more data in the top-$\bar{C}$ classes, the faster the joint model, since more load will be directed to the Student which is faster than the Teacher. We observe that when $\bar{C}$ is too low or too high, e.g., $\bar{C}$=10 or 100, the adaptive inference with the specialized Student becomes less efficient even with large percentages of data in the top-$\bar{C}$ classes. For $\bar{C} = 20$ or 50, the specialized EBJR becomes more efficient, especially when 50% or more of data belong to top-$\bar{C}$ classes. More analysis will be given in the supplementary materials.

Note that EBJR is orthogonal to SOTA dynamic inference approaches, including the weight-sharing ones. In Figure 5-left, we applied EBJR on RS-Net, and showed an improved performance on top of it. More results are given in the supplementary materials.

One limitation of EBJR is memory overheard due to the need of both Student and Teacher at inference time. One solution to deal with this problem is to perform the largest possible Student on the edge, but the Teacher on the cloud. If a desired accuracy on the edge is not met, the Router sends certain samples to the cloud for higher accuracy. Since Student is the largest size that can fit to the device, it is expected to handle most cases, while cloud will be used only sparingly in accuracy-sensitive applications. In this setup, the overall accuracy is not bounded by what can run on edge, but the upper-bound is what can run on cloud.

4 Conclusion

In this paper, we presented an adaptive inference method that combines large, but accurate models with small, but fast models. We proposed an effective energy-based routing module for directing different samples to deep or shallow models. Our method provided a trade-off between the inference latency and accuracy, which in practice is a useful knob for the users to adjust based on their required accuracy or latency, without a need for re-training. In addition, we provided an extension to our method for increasing the inference efficiency by training the shallow models in a way that they only learn to perform the down-stream tasks partially. We presented theoretical and experimental evaluations in support of our method. We hope our work can help facilitate building efficient multi-model inference systems.

References

[1] Felix Abramovich and Marianna Pensky. Classification with many classes: challenges and pluses. *Journal of Multivariate Analysis*, 174:104536, 2019.

[2] Konstantin Berestizshevsky and Guy Even. Dynamically sacrificing accuracy for reduced computation: Cascaded inference based on softmax confidence. In *International Conference on Artificial Neural Networks*, pages 306–320. Springer, 2019.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[4] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.

[5] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[6] Thomas G Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923, 1998.

[7] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5840–5848, 2017.

[8] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019.

[9] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2017.

[10] Dawei Gao, Xiaoxi He, Zimu Zhou, Yongxin Tong, Ke Xu, and Lothar Thiele. Rethinking pruning for accelerating deep inference at the edge. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 155–164, 2020.

[11] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[12] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[13] Fredrik K Gustafsson, Martin Danelljan, Goutam Bhat, and Thomas B Schön. Energy-based models for deep probabilistic regression. In *European Conference on Computer Vision*, pages 325–343. Springer, 2020.

[14] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, 6:3–10, 1994.

[18] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[20] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.

[21] Qing Jin, Linjie Yang, Zhenyu Liao, and Xiaoning Qian. Neural network quantization with scale-adjusted training. In *British Machine Vision Conference (BMVC)*, 2020.

[22] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, et al. The open images dataset v4. *International Journal of Computer Vision*, pages 1–26, 2020.

[23] Duong H Le, Vo Trung Nhan, and Nam Thoai. Paying more attention to snapshots of iterative pruning: Improving model compression via ensemble distillation. In *British Machine Vision Conference (BMVC)*, 2020.

[24] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[25] Youngwan Lee, Joong-won Hwang, Sangrok Lee, Yuseok Bae, and Jongyoul Park. An energy and gpu-computation efficient backbone network for real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[27] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[28] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33, 2020.

[29] 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)*, pages 475–480. IEEE, 2016.

[30] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong-Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. Big/little deep neural network for ultra low power inference. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 124–132. IEEE, 2015.

[31] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018.

[32] Adria Ruiz and Jakob Verbeek. Adaptative inference cost with convolutional neural mixture models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1872–1881, 2019.

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[34] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[35] Hokchhay Tann, Soheil Hashemi, R Iris Bahar, and Sherief Reda. Runtime configurable deep neural networks for energy-accuracy trade-off. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 1–10. IEEE, 2016.

[36] 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)*, pages 2464–2469. IEEE, 2016.

[37] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–18, 2018.

[38] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.

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

[40] Yikai Wang, Fuchun Sun, Duo Li, and Anbang Yao. Resolution switchable networks for runtime efficient image recognition. In *European Conference on Computer Vision*, pages 533–549. Springer, 2020.

[41] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[42] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2369–2378, 2020.

[43] Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, and Andrew Willis. Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In *European conference on computer vision*, pages 299–315. Springer, 2020.

[44] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1803–1811, 2019.

[45] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2018.

[46] Zhihang Yuan, Xin Liu, Bingzhe Wu, and Guangyu Sun. Enas4d: Efficient multi-stage cnn architecture search for dynamic inference. *arXiv preprint arXiv:2009.09182*, 2020.

[47] Hong-Yu Zhou, Bin-Bin Gao, and Jianxin Wu. Adaptive feeding: Achieving fast and accurate detections by adaptively combining object detectors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3505–3513, 2017.

[48] Peng Zhou, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S Davis. M2kd: Incremental learning via multi-model and multi-level knowledge distillation. In *British Machine Vision Conference (BMVC)*, 2020.