ISBNet: Instance-aware Selective Branching Network

Shaofeng Cai¹, Yao Shu¹, Wei Wang¹, Beng Chin Ooi¹
¹National University of Singapore
{shaofeng, shuyao, wangwei, ooibc}@comp.nus.edu.sg

Abstract

Recent years have witnessed growing interests in designing efficient neural networks and neural architecture search (NAS). Although remarkable efficiency and accuracy have been achieved, existing expert designed and NAS models neglect the fact that input instances are of varying complexity thus different amount of computation is required. Inference with a fixed model that processes all instances through the same transformations would waste plenty of computational resources. Therefore, customizing the model capacity in an instance-aware manner is highly demanded. To address this issue, we propose an Instance-aware Selective Branching Network - ISBNet, which supports efficient instance-level inference by selectively bypassing transformation branches of insignificant importance weight. These weights are determined dynamically by accompanying lightweight hypernetworks SelectionNet and further recalibrated by gumbel-softmax for sparse branch selection. Extensive experiments show that ISBNet achieves extremely efficient inference in terms of parameter size and FLOPs comparing to existing networks. For example, ISBNet takes only 8.03% parameters and 30.60% FLOPs of the state-of-the-art efficient network ShuffleNetV2 with comparable accuracy.

1 Introduction

Deep convolutional neural networks (CNNs) [1, 2] have revolutionized computer vision with increasingly larger and more sophisticated architectures. These models typically comprise hundreds of layers and contain tens of millions of parameters, which take substantial computational resources for both training and inference. Generally, the model architectures are designed and calibrated by domain experts with rich engineering experience. Recently, there has been a growing interest in efficient network design [3–6] and neural architecture search (NAS) [2, 7, 8], whose main targets are to devise network architectures that are efficient during inference and automate the architecture design process respectively.

Many efficient architectures have been designed in recent years. E.g. SqueezeNet [4] and MobileNet [3] reduce parameter size and computation greatly. These networks mainly aim to reduce the computational cost measured by FLOPs. More recent works MobileNetV2 [6] and ShuffleNetV2 [9] further reduce the FLOPs significantly. Although these networks reduce a huge amount of resource consumption, it is non-trivial to devise these architectures without plenty of engineering experience.

Automating the architecture design process via neural architecture search (NAS) has attracted increasing attention in recent years. Mainstream NAS algorithms [10, 2, 7] search for the network architecture iteratively. In each iteration, an architecture is proposed by a controller, and then trained and evaluated. The evaluation performance is in turn exploited to update the controller. This process is incredibly slow because both the controller and each proposed architecture need training. For instance, the reinforcement learning (RL) based controller NASNet [2] takes 1800 GPU days and the evolution algorithm based controller AmoebaNet [7] costs 3150 GPU days to obtain the best architecture. Many acceleration methods [11–14] have been proposed to further accelerate the search
process, more recent works \cite{8,15,16} remove the controller and instead optimize the architecture selection and parameters together with gradient-based optimization algorithms.

Although remarkable efficiency and prediction performance have been obtained, both expert designed and NAS searched models neglect one critical issue that would affect inference efficiency fundamentally. The architectures of these models are fixed during inference time and thus not adaptive to the varying complexity of input instances. However, in real-world applications, there are only a small fraction of input instances requiring deep representations \cite{17,18}. Therefore, plenty of computational resources would be wasted if all instances are treated equally. Designing a model with sufficient representational power to cover the hard instances, meanwhile a finer-grained control to provide just necessary computation dynamically for instance of varying difficulty is thus highly demanded.

In this paper, we aim to address the aforementioned issue with ISBNet, whose building block Cell is illustrated in Figure 1. Following the widely adopted strategy in NAS \cite{2,14,8,15}, the backbone network is a stack of \( L \) structurally identical cells, receiving inputs from their two previous cells and each cell contains \( N \) inter-connected computational Nodes. While the architecture of ISBNet deviates from conventional wisdom of NAS which painstakingly search for the connection topology and the corresponding transformation operation coupling each connection. In ISBNet, each node is simply connected to its two preceding nodes and each connection transforms via a candidate set of \( O \) operations (branches). To allow for instance-aware inference control in the branch level, we further integrate \( L \) lightweight hypernetworks SelectionNets accompanying each cell to determine the importance weights for each branch. Gumbel-softmax \cite{19,20} is introduced to further recalibrate these weights determined by the SelectionNet, to enable the efficient gradient-based optimization for the whole network during training, and more importantly, ensure sparse Categorical branch selection during inference for efficiency.

The main novelties and contributions of ISBNet over existing methods can be summarized as follows:

- **ISBNet** is a general architecture framework combining advantages from both efficient network design and NAS, whose components are readily customizable.
- **ISBNet** is a novel architecture supporting the instance-level selective branching mechanism by introducing lightweight SelectionNets, which improves inference efficiency significantly by reducing unnecessary computation.
- **ISBNet** successfully integrates gumbel-softmax to the branch selection process, which enables direct gradient descent optimization and is more tractable than RL-based method.
- **ISBNet** achieves state-of-the-art inference efficiency in terms of parameter size and FLOPs and inherently supports applications requiring fine-grained instance-level control, e.g. anytime prediction \cite{18}.

Our experiments show that ISBNet is extremely efficient during inference and successfully selects only vital branches for each input instance. In particular, with a minor 1.53% accuracy decrease, ISBNet reduces the parameter size and FLOPs by 10x and 11.31x respectively comparing to the NAS searched high-performance architecture DARTS \cite{8}. Furthermore, with a tiny model of 0.57M parameters, ISBNet achieves comparable accuracy while with only 8.03% and 30.60% inference time parameter size and FLOPs comparing to the expert-designed efficient network ShuffleNetV2 1.5x \cite{9}. We also conduct ablation studies and visualize the branch selection process to better understand the proposed architecture. The main results and findings are summarized in Sec 4.2 and Sec 4.3.

2 Related Work

**Efficient Network Design.** Designing resource-aware networks \cite{4,21,9,6,21} has attracted great attention in recent years, which mainly focuses on reducing parameter size and inference FLOPs
in many works. For instance, SqueezeNet [22] reduces parameters and computation with the fire module; MobileNetV2 [6] utilize depth-wise and point-wise convolution for more parameter-efficient convolutional neural networks; ShuffleNetV2 [9] proposes lightweight group convolution with channel shuffle to facilitate the information flowing across the channels. To make inference efficient, many of these transformations are introduced to the candidate operation set in ISBNet.

Many recent works explore conditional [17] and resource-constrained prediction [18] for efficiency. Similar to our work, SkipNet [17] introduces a gating hypernetwork to sequentially decide whether to bypass current residual layer [1] conditional on current input instance. However, our ISBNet provides more efficient and diversified branch selections for the backbone network and the hypernetworks in ISBNet are optimized in an end-to-end training manner instead of generally less tractable policy gradient [23]. MSDNet [18] supports anytime prediction within prescribed computational resource constraint during inference by inserting multiple classifiers into a 2D multi-scale version of DenseNet [24]. By early-exit into a classifier, MSDNet can provide approximate predictions with minor accuracy decrease. Functionally, ISBNet also supports anytime prediction by controlling the number of branches selected, thus per-input inference cost correspondingly.

Neural Architecture Search. Mainstream NAS [2, 7] treats architecture search as a stand-alone process whose optimization is severed from candidate architecture optimization. Search algorithms such as RL-based NAS [2] and evolutionary-based NAS [7] obtain state-of-the-art architectures at an unprecedented amount of the GPU-time searching cost. Recently, many works have been proposed to accelerate the search pipeline, e.g., via performance prediction [11, 12], hypernetworks generating initialization weights [25], weight sharing [13, 14]. These approaches largely alleviate the search inefficiency while the inherent issue of scalability remains.

Another line of works [8, 15, 16] instead integrates the architecture search process and architecture optimization into the same gradient-based optimization framework. In particular, DARTS [8] relaxes discrete search space to be continuous by introducing operation mixing weights to each connection and optimizes these weights directly with gradient back-propagated from validation loss. Similarly, the discrete search space in SNAS [15] is represented with sets of one-hot random variables coupling the connections, which are made differentiable by relaxing the discrete distribution with continuous concrete distribution [19, 20]. In terms of architecture optimization, ISBNet also relaxes the discrete branch selection to continuous coupling weights optimized by gradient descent; while instead of directly optimizing on the weights, we introduce SelectionNet to dynamically generate these weights which are more effective and bring about larger model capacity. Further, SelectionNet supports instance-level architecture customization rather than a fixed model as in the architecture search.

3 Instance-aware Selective Branching Network

3.1 The Backbone Network

The backbone network is constructed with a stack of \( L \) cells, each of which is a directed acyclic graph consisting of an ordered sequence of \( N \) intermediate nodes. As is illustrated in Figure 1, \( x_0^l \) and \( x_1^l \) are the two input nodes from the two preceding cells; each intermediate node \( x_i^l (i \geq 2) \) forms a latent representation in the \( l \)-th cell and receives \( n = 2^{i-1} \) input nodes \( x_j \) from its \( n \) preceding nodes:

\[
x_i^l = \sum_{j=i-n}^{i-1} O_{j,i}(x_j^l)
\]

Thereby, each cell contains \( C = n \cdot N \) connections in total. The connection passes information from node \( x_j^l \) to \( x_i^l \) after the aggregation of a candidate set of \( O \) branches of transformation inspired from widely-adopted transformations in NAS [14, 8, 15] and efficient network design [4, 6, 5]:

\[
O_{j,i}(x_j^l) = \sum_{o=1}^{O} w_o \cdot F_o(x_j^l)
\]

\(^1n \) can be set larger than 2 to allow for deeper and wider local representation. Particularly, \( n = 1 \) results in a canonical feed-forward CNN and \( n = i-1 \) for each \( x_i^l \) leads to dense connection of DenseNet [24].
where $w_o$ here represents the importance of the $o_{ilh}$ branch (operation) of the connection and is dynamically generated by the hypernetwork rather than a fixed learned parameter as in existing NAS methods [8, 15]. We shall introduce the hypernetwork in Section 3.2. Finally, the output of the cell $x_{l_{out}}$ is aggregated by concatenating the output from all the intermediate nodes. We shall use superscript $l$, subscript $c$ and $o$ to index the cell, connection, and branch respectively.

Under this architecture formulation framework, we can readily adjust the number of candidate branches $O$ and also the specific transformations before training, customizing model capacity and efficiency respectively depending on the difficulty of the task and resource constraints in deployment.

### 3.2 SelectionNet for Weight Recalibration

To support instance level inference control, we further introduce $L$ lightweight hypernetworks SelectionNet accompanying each cell. Each SelectionNet $SNet^l$ receives the same input as the accompanying $l_{th}$ cell, namely two output nodes $x_{l_{out}}^0, x_{l_{out}}^1$ from previous cells, and simultaneously produce $C$ sets of recalibration weights for each connection in the cell:

$$W^l = SNet^l(x_{l_{out}}^0, x_{l_{out}}^1)$$  \quad (3)

where $W^l \in \mathbb{R}^{C \times O}$ is the recalibration weight matrix for the $l_{th}$ cell. The SelectionNet $SNet^l$ dynamically generates these weights with the pipeline of $m = 2$ convolutional blocks, a global average pooling and finally an affine transformation. For the $m$ convolutional transformation, we adopt separable convolution [6] which contains a point-wise ($1 \times 1$) convolution and a depth-wise convolution of stride 2 and kernel size $5 \times 5$. The stride reduces the parameter size and computation of $SNet^l$, and the larger kernel size for depth-wise convolution here incurs negligible overhead while extracts features for the immediate weight generation with a larger local receptive field.

The recalibration weights given by the SelectionNet is reminiscent of convolutional attention mechanism [22, 26, 27], where attention weights are determined dynamically by summarizing information of the immediate input and then exploited to recalibrate the relative importance of different input dimensions, e.g. channels in SENet [22]. In ISBNet, the recalibration weights are introduced to the branch. Particularly, each candidate operation of the connection is coupled with a rescaling weight.

The gumbel-softmax [19, 20] technique and the reparameterization trick [28] is introduced to further recalibrate these weights generated by the SelectionNet, to enable the efficient gradient-based optimization for the whole network during training, and more importantly, ensure a sparse selection of important branches during inference. More specifically, each set of coupling weights $W^l_c \in \mathbb{R}^O$ for the $c_{th}$ connection in the $l_{th}$ cell ($C^l_c$) after the following recalibration of the gumbel-softmax follows concrete distribution [20] controlled by a temperature parameter $\tau$:

$$\bar{w}^l_{c,o} = \frac{\exp((w^l_{c,o} + G^l_{c,o}) / \tau)}{\sum_{o' = 1}^O \exp((w^l_{c,o} + G^l_{c,o}) / \tau)}, \tau > 0$$  \quad (4)

where $\bar{w}^l_{c,o}$ is then directly used for branch recalibration as is in Equation 2 and $G^l_{c,o} = -\log(-\log(U^l_{c,o}))$ here is a gumbel random variable coupling with $o_{ilh}$ branch by sampling $U^l_{c,o}$ from Uniform(0,1) [19]. The concrete distribution [20] has the following property that: (1) $\bar{w}^l_{c,o} = \frac{1}{O}$, as $\tau \rightarrow +\infty$, and (2) $p(\lim_{\tau \rightarrow 0} \bar{w}^l_{c,o} = 1) = \exp(w^l_{c,o}) / \sum_{o' = 1}^O \exp(w^l_{c,o'})$. Therefore, high temperature leads to dense uniform branch selection and low temperature tends to sparsely sample branches following the corresponding categorical distribution parameterized by $softmax(W^l_c)$.

### 3.3 Optimization and Inference for ISBNet

With the continuous relaxation of the gumbel-softmax [19, 20] and the reparameterization [28], the branch selection process of the SelectionNets is made directly differentiable with respect to the weight $w^l_{c,o}$. In particular, the gradient $\frac{\partial \mathcal{L}}{\partial w^l_{c,o}}$ back propagated from Cross-Entropy Loss $\mathcal{L}$ to $\bar{w}^l_{c,o}$ through the backbone network can be directly backpropagated to $w^l_{c,o}$ with low variance [20], and further to the
where \( y \) is the ground truth class label, \( \hat{y} \) the prediction, \( \lambda \) controls the regularization strength and \( \mathcal{R}(\cdot) \) calculates the resource consumption of each operation \( \mathcal{F}_{c,o}(\cdot) \). The operation importance weight \( \hat{w}_{c,o} \) here also represents the probability of the corresponding branch \( \mathcal{F}_{c,o} \) being selected during inference, therefore the regularization term \( \mathbb{E}[\mathcal{R}] \) corresponds to the expectation of the aggregated resource taken for each input instance.

The resource regularizer is readily adjustable depending on deployment constraints, which may include the parameter size, FLOPs, and memory access cost (MAC). In this work, we mainly focus on the inference time, namely FLOPs, which can be calculated beforehand for each branch. \( \mathcal{R}(\mathcal{F}_{c,o}(\cdot)) \) is thus a constant here, which means that the regularizer \( \mathcal{R} \) is also directly differentiable with respect to \( \hat{w}_{c,o} \). We denote ISBNet trained with regularization strength \( \lambda \) as ISBNet-R-\( \lambda \).

During inference, the instance-level selective branching is achieved by selecting branches of top \( k \) largest recalibration weight for each connection \( C^l_c \) whose aggregated weight \( S^l_c \) just surpasses a threshold \( T \). Denoting \( \hat{W}^l_c \) sorted in descending order as \( \hat{W}^l_c \), then:

\[
S^l_c = \min \{ S_k : (S_k = \sum_{o=1}^{k} \hat{w}_{c,o}^l) \land (S_k \geq T) \}
\]

After the selection, the recalibration weight \( \hat{w}_{c,o}^l \) of the selected branch is rescaled by \( \frac{1}{S^l_c} \) to stabilize the scale of the representation. Consequently, the SelectionNet will dynamically select vital branches for each instance depending on the input difficulty and also the FLOPs of each branch, i.e. trading off between \( \mathcal{L} \) and \( \mathcal{R} \) in Equation 5. Furthermore, the resource consumption of each instance can be precisely regulated in a fine-grained manner by scheduling the threshold dynamically for each connection. In this work, the same threshold is shared among all connections for simplicity and ISBNet inference with the threshold \( t \) is denoted as ISBNet-T-\( t \).

Under such inference scheme, the backbone network therefore comprises \((2^O-1)^{L \cdot C}\) possible candidate subnets in total, corresponding to each unique branch selection of all \( L \cdot C \) connections. For a small ISBNet of 10 cells, with 5 candidate operations, 8 connections per cell, there are up to \((2^5-1)^{8 \cdot 10} \approx 2 \cdot 10^{10}\) possible candidate architectures of different branch combination, which is orders of magnitudes larger than the search space of conventional NAS [14, 8, 15].
4 Experiments

In this section, we evaluate the performance of ISBNet in comparison with state-of-the-art expert-designed efficient networks and best-performing NAS architectures. The experimental details can be found in Sec. 4.1. Main results are reported in Sec. 4.2, following which we show the performance of ISBNet in anytime prediction and in Sec. 4.3 the visualizations of the branch selection process.

4.1 Experimental Setup

Dataset CIFAR-10/100 [29] dataset contains 50,000 training images and 10,000 test images of $32 \times 32$ pixels in 10/100 classes. We adopt standard data pre-processing and argumentation pipeline [8, 15] as follows: zero padding the training images with 4 pixels on each side and then randomly cropping back to $32 \times 32$ images; randomly flipping training images horizontally; normalizing training images with channel means and standard deviations; applying standard cutout regularization with cutout [30] length 16.

Candidate Operation Set The candidate operation set includes the following 5 ($O = 5$) operations:

- $3 \times 3$ max-pooling
- $3 \times 3$ avg-pooling
- skip connection
- $3 \times 3$ separable-conv
- $5 \times 5$ separable-conv

In particular, separable-conv stands for a pipeline of operations in the order of ReLU-Conv-BN-ReLU-Conv-BN. Skip connection allows for efficient representation forwarding; pooling layers here are computational light with no parameters; and separable-conv dominates the parameter size and also computation in each connection. The three types of operations support trade-off between representation power and efficiency for the branch selection of each connection.

Temperature Annealing Scheme In the pre-training stage, the temperature $\tau$ is fixed to 3 till full convergence and in the fine-tuning stage, $\tau$ is initialized to 1.0 and then is annealed steadily by $\exp(-0.006) \approx 0.999$ every epoch.

Architecture Details We implement two ISBNet architectures of different size in our experiments: (1) ISBNet(S), a small network with $L = 5$ cells and 15 initial channels; (2) ISBNet(M), a medium network with $L = 10$ cells and 20 initial channels. These two architectures have the same number of $N = 4$ computational nodes in each cell. For both architectures, nodes directly connected to the input nodes are downsampled with stride 2 for $\frac{L}{3}$-th and $\frac{2L}{3}$-th cells. An auxiliary classifier with weight 0.4 is connected to the output of $\frac{2L}{3}$-th cell for additional regularization.

Optimization Details For both training stages, we apply SGD with momentum 0.9 and weight decay $3 \times 10^{-4}$ for 1200 epochs. The learning rate is initialized to 0.025 and 0.005 for the pre-training and fine-tuning stage respectively. We adopt drop-path [31] rate 0.7 for each branch. For both stages, the learning rate is annealed to zero following the standard cosine annealing schedule [32]. We use batch size 256 for ISBNet(S) and 80 for ISBNet(M) to fit the whole network into one GPU.

4.2 ISBNet Performance Evaluation

Overall Results and Discussion. Table 1 summarizes the overall performance of ISBNet under different inference threshold $T$ and resource constraint strength $R$. In terms of training efficiency, our ISBNet only takes 2.2 and 8.3 GPU days for ISBNet(S) and ISBNet(M) respectively, which is up to three orders of magnitudes less GPU hours than conventional evolution-based NAS or RL-based NAS thanks to our efficient network design and the end-to-end gradient-based optimization. As for inference time performance, ISBNet reduces a drastic amount of the parameter size and FLOPs comparing to baseline networks. Specifically, with comparable accuracy, ISBNet(S)-R-0.5-T-0.8 only takes 0.20M parameters and 29.28M FLOPs on average during inference, which is only 8.03% and 30.60% of state-of-the-art efficient network ShuffleNetV2 1.5×; ISBNet(S)-R-0.0-T-0.8 achieves up to 10x and 11x parameter size and FLOPs reduction than DARTS with 1.53% accuracy decrease. The massive parameter size and FLOPs reduction demonstrate that the selective branching mechanism in ISBNet enables extremely efficient instance-level prediction. This is also corroborated by the
Table 1: Statistics and performance of ISBNet compared with other state-of-the-art architectures on CIFAR-10. Searching and Training cost marked with * is measured on GTX 1080Ti under our implementation. Performance of ISBNet are reported with full model / selective branching respectively.

| Architecture          | Test Error (%) | Training / Inference Params (M) | Training / Inference FLOPs (M) | Search Method | Search Space | Training & Search Cost (GPU days) |
|-----------------------|----------------|---------------------------------|-------------------------------|---------------|--------------|----------------------------------|
| ResNet110 [1]         | 6.25           | 42.51                           | 2519.71                       | manual        | –            | 0.53*                            |
| DenseNet-BC [24]      | 3.46           | 25.6                            | 9345.25                       | manual        | –            | 4.03*                            |
| MobileNetV2 1.0 [3]   | 5.56           | 3.00                            | 94.42                         | manual        | –            | 0.13*                            |
| ShuffleNetV2 1.5 [3]  | 6.36           | 2.65                            | 80.99                         | manual        | –            | 0.10*                            |
| NASNet-A [15]         | –              | –                               | –                             | –             | –            | –                                |
| AmoebaNet-A [7]       | –              | –                               | –                             | –             | –            | –                                |
| ENAS [14]             | –              | –                               | –                             | –             | –            | –                                |
| InstaNAS-C10-B [33]  | –              | –                               | –                             | –             | –            | –                                |
| ISBNet(M)-R           | 4.78           | 0.46                            | 77.34                         | gradient      | layer-wise   | –                                |
| ISBNet(S)-No-Selection + cutout | 4.13/5.27, 0.57/0.55 | 4.37/15.22, 0.57/0.48 | 84.65/80.99 | gradient | layer-wise | –                                |
| ISBNet(S)-Gumbel-Softmax-T-0.8 + cutout | 4.13/4.53, 0.57/0.33 | 4.13/4.49, 0.57/0.31 | 84.65/47.91 | gradient | layer-wise | –                                |
| ISBNet(S)-R-0.0-T-0.8 + cutout | 4.13/6.33, 0.57/0.20 | 4.37/15.22, 0.57/0.48 | 84.65/47.91 | gradient | layer-wise | –                                |
| ISBNet(S)-R-0.1-T-0.8 + cutout | 4.13/6.33, 0.57/0.20 | 4.37/15.22, 0.57/0.48 | 84.65/47.91 | gradient | layer-wise | –                                |
| ISBNet(S)-R-0.5-T-0.8 + cutout | 3.26/3.76, 1.86/1.02 | 267.26/119.46 | 84.65/47.91 | gradient | layer-wise | –                                |
| ISBNet(S)-R-0.8-T-0.8 + cutout | 3.26/3.76, 1.86/1.02 | 267.26/119.46 | 84.65/47.91 | gradient | layer-wise | –                                |
| ISBNet(M)-R-0.0-T-0.8 + cutout | 3.60/3.91, 1.86/1.02 | 267.26/119.46 | 84.65/47.91 | gradient | layer-wise | –                                |
| ISBNet(M)-R-0.1-T-0.8 + cutout | 3.26/3.76, 1.86/1.02 | 267.26/119.46 | 84.65/47.91 | gradient | layer-wise | –                                |
| ISBNet(M)-R-0.5-T-0.8 + cutout | 3.26/3.76, 1.86/1.02 | 267.26/119.46 | 84.65/47.91 | gradient | layer-wise | –                                |

The results show that a small ISBNet is able to achieve decent accuracy comparable to the best NAS searched models, meanwhile with far less amount of inference parameters and FLOPs. This questions the necessity of current laborious architecture search of NAS [2, 7]. In this paper, we propose a selective branching mechanism evocative of convolutional attention [22, 26] via the introduction of the hypernetworks SelectionNets, which leads to larger model capacity. With 19.30% and 8.54% more parameters and FLOPs, ISBNet integrated with SelectionNets achieves 0.41% noticeably higher accuracy. Furthermore, when trained with gumbel-softmax, SelectionNets enables the network to efficiently select proper branches and customize its architecture on a per-input basis during inference, with negligible accuracy decrease. Gumbel-softmax is necessary because SelectionNets trained with softmax alone suffers from catastrophic accuracy decrease, from 4.37% to 15.22%, while with limited parameter size and FLOPs reduction with inference threshold 0.6.

Accuracy-FLOPs Trade-off. We further evaluate the performance of ISBNet in the setting of anytime prediction [18] where the network is required to output a prediction within a prescribed budget dynamically for each input instance. Figure 2 illustrates the performance of ISBNet in comparison with baselines of anytime prediction MSDNet [18], conditional prediction SkipNet [7] and model compression NetworkSlimming [34]. The results demonstrate that ISBNet obtains significantly better performance than all baselines, where none of the (budget, accuracy) trade-off points from ISBNet is dominated. In particular, ISBNet achieves much higher accuracy than SkipNet and MSDNet in the budget range from 43.95M to 84.65M FLOPs.
4.3 Visualization of Selective Branching

**Ratio of Selective Branching.** We visualize in Figure 3 the average recalibration weight and branch selection ratio of representative cells in ISBNet(S)-R-0.0-T-0.8, which indicates the evolution of the expectation and the average in the final model of the number of each branch being selected during inference respectively. An obvious stratified pattern can be observed that one separable-convolution branch gradually dominates the connection in lower-layer cells while in high-layer cells, the branch selection tends to be more uniform and diversified. This pattern demonstrates that features extracted in lower layers share similar branch of transformation where branch pruning can be deployed to reduce the parameter size; while the instance-aware efficient inference requires the diversities of branch option ascending the layers. Our further experiments show that the average number of branches selected in the last cell is 1.1, indicating that only one branch is required for the inference of most instances. Somewhat surprisingly, we also find that for a given input instance, the set of selected branches differs significantly when inference with different random seeds; yet the SelectionNets are quite confident about each branch selection with weights larger than 0.9 generally. We conjecture that this phenomenon mainly results from the tendency towards the categorical sampling of the candidate branches with gumbel-softmax. During the temperature annealing training stage, the architecture gradually adapts to such stochasticity meanwhile maintaining its accuracy.

![Figure 3: Average Recalibration Weight during the temperature annealing training stage, and inference time Branch Selection Ratio of the 5 branches for each connection in the first/last cell of ISBNet(S)-R-0.0-T-0.8.](image)

**Qualitative Difference between Instances.** Denoting instances that the network is confident with in prediction as *easy* instance and uncertain as *hard* instance, we visualize the clustering of *easy* and *hard* instances in Figure 4 to better understand the selective branching mechanism. We find that the certainty of the prediction made by ISBNet mainly depends on the image quality. In general, easy instances are more salient (clear with high contrast) while hard instances are more inconspicuous (dark with low contrast). We also compute the accuracy and average FLOPs of each cluster. On average, easy instances achieve much higher classification accuracy with 11.2% fewer FLOPs compared with hard instances. This shows that computation could be greatly saved without sacrificing accuracy by selective bypassing unimportant branches for relatively easy instances.

![Figure 4: Visualization of *easy* and *hard* instances of model ISBNet(M)-R-0.0-T-0.8 on CIFAR-10. Easy instances are clearer and brighter in general while hard instances are darker and blurry.](image)

5 Conclusion

In this work, we present ISBNet, a novel network framework combining advantages from both efficient network design and neural architecture search. To achieve instance-aware efficient inference, a series of accompanying lightweight hypernetworks are introduced to the backbone network to dynamically determine importance weights for selective branching. We have successfully integrated gumbel-softmax and reparameterization trick to the branch selection process, which enables accessible and tractable gradient-based end-to-end training, and more importantly, extremely efficient inference. The inference efficiency is further promoted with the resource-aware regularization.

Extensive experiments and visualizations have been conducted, whose results validate the instance-level inference efficiency by selective branching. Particularly, with ISBNet we have achieved 91.97% parameter size and 69.40% FLOPs reduction over state-of-the-art efficient network ShuffleNetV2.
References

[1] 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, pages 770–778, 2016.

[2] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8697–8710, 2018.

[3] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017.

[4] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. arXiv preprint arXiv:1602.07360, 2016.

[5] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6848–6856, 2018.

[6] 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.

[7] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. arXiv preprint arXiv:1802.01548, 2018.

[8] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

[9] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), pages 116–131, 2018.

[10] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578, 2016.

[11] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. Accelerating neural architecture search using performance prediction. arXiv preprint arXiv:1705.10823, 2017.

[12] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In Proceedings of the European Conference on Computer Vision (ECCV), pages 19–34, 2018.

[13] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In International Conference on Machine Learning, pages 549–558, 2018.

[14] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. arXiv preprint arXiv:1802.03268, 2018.

[15] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926, 2018.

[16] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. arXiv preprint arXiv:1812.03443, 2018.

[17] 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.
[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] Eric Jung, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144, 2016.

[20] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712, 2016.

[21] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. SqueezeNext: Hardware-aware neural network design. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 1638–1647, 2018.

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7132–7141, 2018.

[23] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4):229–256, 1992.

[24] 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.

[25] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. arXiv preprint arXiv:1708.05344, 2017.

[26] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), pages 3–19, 2018.

[27] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In European Conference on Computer Vision, pages 483–499. Springer, 2016.

[28] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

[29] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[30] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

[31] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648, 2016.

[32] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983, 2016.

[33] An-Chieh Cheng, Chieh Hubert Lin, Da-Cheng Juan, Wei Wei, and Min Sun. Instanas: Instance-aware neural architecture search. arXiv preprint arXiv:1811.10201, 2018.

[34] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision, pages 2736–2744, 2017.