Learning Personalized Modular Network Guided by Structured Knowledge

Xiaodan Liang
Sun Yat-sen University
xdliang328@gmail.com

Abstract

The dominant deep learning approaches use a "one-size-fits-all" paradigm with the hope that underlying characteristics of diverse inputs can be captured via a fixed structure. They also overlook the importance of explicitly modeling feature hierarchy. However, complex real-world tasks often require discovering diverse reasoning paths for different inputs to achieve satisfying predictions, especially for challenging large-scale recognition tasks with complex label relations. In this paper, we treat the structured commonsense knowledge (e.g. concept hierarchy) as the guidance of customizing more powerful and explainable network structures for distinct inputs, leading to dynamic and individualized inference paths. Give an off-the-shelf large network configuration, the proposed Personalized Modular Network (PMN) is learned by selectively activating a sequence of network modules where each of them is designated to recognize particular levels of structured knowledge. Learning semantic configurations and activation of modules to align well with structured knowledge can be regarded as a decision-making procedure, which is solved by a new graph-based reinforcement learning algorithm. Experiments on three semantic segmentation tasks and classification tasks show our PMN can achieve superior performance with the reduced number of network modules while discovering personalized and explainable module configurations for each input.

1. Introduction

Recent successes of deep neural networks in common tasks [12, 19, 8] have been accompanied by the increasingly complex and deep network architectures tailored for each task. Most existing works follow the "one-size-fits-all" paradigm and hope that all relationships between inputs and targets can be learned via implicit feature propagation along with the fixed network routine, regardless of whether the inputs are too easy or difficult for the network capacity. However, different examples actually need specialized network structures or feature propagation routines with respect to their distinct complexities and specific targets. This can be well motivated by the feature visualization study in [27] where interpretable feature hierarchy can be learned in current neural networks. For example, in terms of image recognition, for easy images (e.g. most common dog), network compression [22, 36, 34] can be pursued by pruning some redundant network modules. In contrast, for difficult images (e.g. rare concepts, such as violin), more sophisticated structures or external constraints [39, 30, 7] should be exploited to improve the model capability. The varying requirement of the model complexity by different instances indicates the potential benefits of dynamic networks, where most appropriate and explainable network structures can be discovered for different inputs.

Some existing input-dependent networks [36, 23] are mainly designed for compressing networks by dynamically executing different modules of a network model. However, their learned policies for selecting specific modules are solely driven by the final reward (i.e. classification accuracy). Such policy learning lacks the essential structure interpretability to explain why a particular module is activated and results in heavy exploration cost to find an optimal selection sequence. On the contrary, an important merit of the human perception system [29] is its ability to adaptively allocate time and inspection for visual recognition and exploit external knowledge for better decision-making. For example, at the first single glimpse, it is sufficient to recognize some coarse classes (e.g. animals, human or scenes). Based on this impression, more time and attention are devoted to further understand among fine-grained concepts (e.g. cat or dogs) by associating their characteristics with those of semantic correlated concepts (e.g. parents in concept hierarchy).

To mimic this dynamic reasoning system, we propose to learn dynamic module configurations to infer personalized reasoning paths for each input, named as Personalized Modular Network (PMN), as shown in Figure 1. Moreover, it is beneficial to bridge visual feature hierarchy with commonsense human knowledge which can be formed as various undirected graphs consisting of rich relationships among concepts. For example, "Shetland Sheepdog" and "Husky" share one superclass "dog" due to some common characteristics. Beyond only inferring the binary selection of each

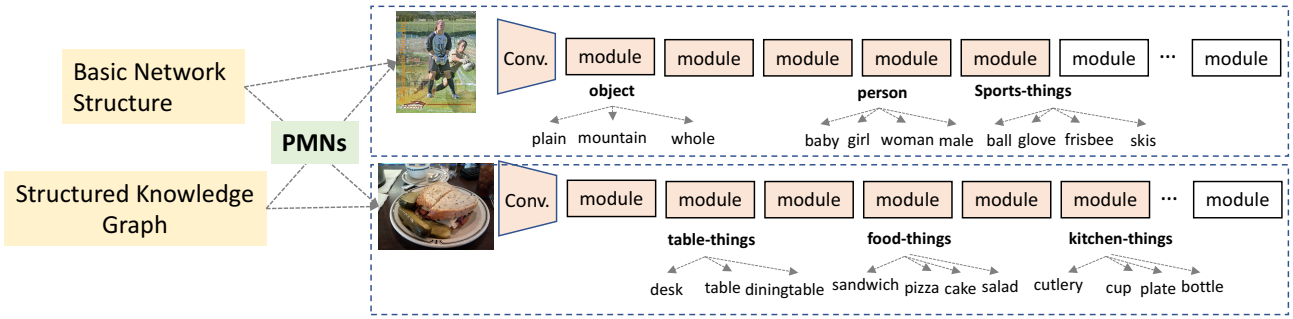

Figure 1. Our PMN aims to learn different module computations for each input and each module is designated to recognize concepts of distinct knowledge levels (e.g. object or person), given a basic network structure and the prior structured knowledge.

module as previous works did, our PMN aims to further determine which particular levels of structured knowledge each module is capable enough to infer, and reconfigure the selected modules to make them semantic coherent with structured knowledge.

Learning to sequentially select modules for addressing particular levels of knowledge can be formulated as a sequential decision-making procedure. We propose a new graph-based reinforcement learning algorithm, which specifies varying knowledge level set as the particular action space for each network module, driven by a graph search function over structured knowledge. Particularly, the action space of each network module is contains the finer-levels of knowledge (e.g cat or dog) relative to activated levels (e.g. animals) by former modules, which enforces early modules characterize easy knowledge while later modules with more feature abstraction are responsible for more fine-grained knowledge. Taking features produced by the module for each image as inputs, a recurrent policy network is learned to reason over the evolved action space. And the reward function is designed to encourage both early-stopping the network propagation and obtaining higher recognition accuracy for balancing both efficiency and accuracy.

We demonstrate the superior performance achieved by our PMN on three segmentation tasks (COCO-Stuff, PASCAL-Context, and ADE20k) and two classification tasks (CIFAR-10 and CIFAR-100). More interestingly, our PMN discovers meaningful patterns of module configurations for different images, and also reduce the number of activated modules for easy samples to obtain good computation efficiency.

2. Related Work

Instance-specific Computation. Conditional computation methods have been proposed to dynamically execute different modules of a network specified for each instance [4, 3, 36]. Learning with reinforcement learning [3, 23, 10], sparse activations are usually estimated to selectively turn on and off a subset of modules based on the input. Their rewards driven only by final accuracy are accumulated after a sequence of decisions computed after each layer, resulting in overhead policy execution. In contrast, our PMN introduces the structured knowledge graph to guide module selections for reaching good interpretability, and a new graph-based reinforcement learning is proposed to evolve action space for each module. Moreover, the graph-based reinforcement learning also makes all routing decisions in a single step for reducing overhead cost and computations. Most recently, Liang et al. [20] explicitly incorporate concept hierarchy into network construction by adding more modules upon the basic network, leading to heavier computation. On the contrary, our PMN fully exploits the capability of existing module in characterizing different concepts, which can selectively reduce the module usage for easy examples or reconfigure outputs of modules for difficult examples.

Dynamic feature computation in vision and NLP applications. Dynamic feature computation has also been explored in vision and natural language applications [6, 32, 15], actively deciding which frames, regions or text entity modules to execute. There also exists some early prediction models, a type of conditional computation models that exit once a criterion is satisfied at early layers. BranchyNet [35] is composed of branches at each layer to make early classification decisions. Figurnov et al. [11] apply Adaptive Computation Time (ACT) to each spatial position of multiple image blocks. This method identifies instance-specific ResNet configurations, but only allows configurations that use early and contiguous blocks. Instead, our PMN allows different semantic configurations of modules towards aligning with structured knowledge. Our approach, posed as the general dynamic network learning, can be easily incorporated for these downstream tasks.

3. Approach

We now present the proposed Personalized Modular Networks (PMN). As illustrated in Figure 2, given an input x, our PMN is to find the best semantic configurations of a minimum number of modules in one predefined network, such

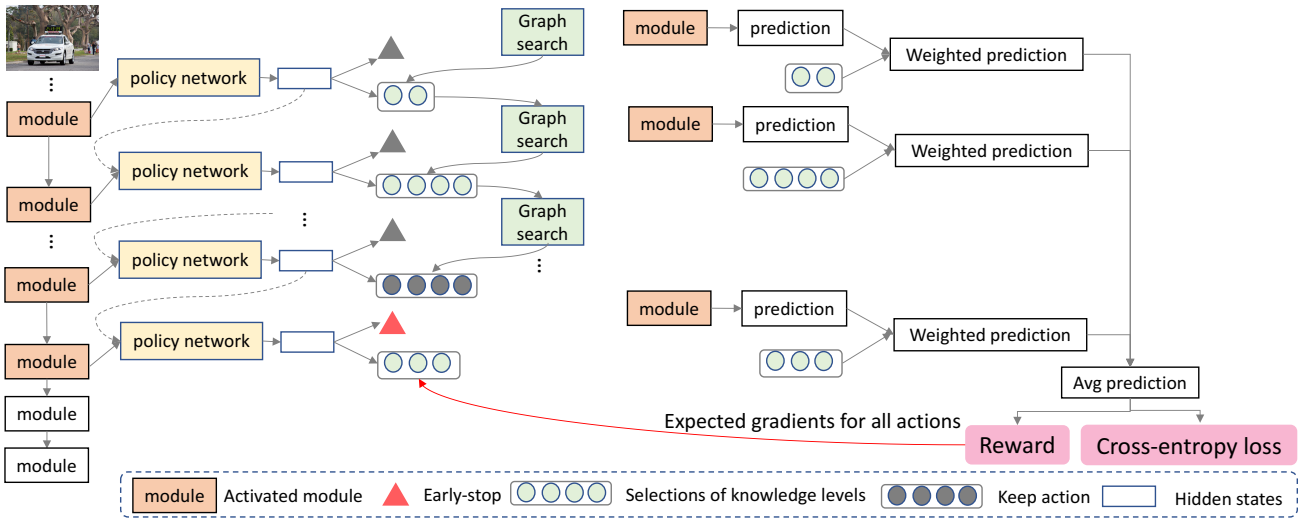

Figure 2. Personalized Modular Networks (PMN) which discovers dynamic module configurations for each input via a graph-based reinforcement learning. The features of each module are passed a recurrent policy network to produce probabilities of *early-stop* action and selections of adaptive knowledge levels which are determined by a graph searching function. Features from activated modules are then passed through a prediction network to obtain final predictions, which are then softly weighted by selections of knowledge levels. Finally, expected gradients are back-propagated through all actions based on a reward function.

that each selected module can be designated to characterize specific levels of structured knowledge. Treating the task of finding this configuration as a search problem becomes intractable for deeper models as the number of potential configurations grows exponentially with the number of modules, and it is non-trivial to directly adopt a supervised learning framework. We thus propose a graph-based reinforcement learning to derive optimal module configurations, which specifies evolving action space for each module following structured knowledge.

3.1. Structured Knowledge Graph

Here, we explore the semantic concept graph as one kind of external structured knowledge , formulated as $\mathcal{G} =< \mathcal{N}, \mathcal{E} >$, where $\mathcal{N}$ consists of all concepts $\{n_i\}$ in a predefined knowledge graph and each graph edge (n_i, n_j) indicates n_i (e.g. *chair*) is the parent concept of n_j (e.g. *armchair*). Our PMN thus learns to configure the most appropriate knowledge levels to be recognized given a specific module by searching over a group of parent concepts (e.g *chair*) that has at least two child concepts within $\mathcal{N}$. The available knowledge level set (action set) for each module is thus equal to the number of parent concepts. The action selection of each module is a multi-classification problem as multiple levels of structured knowledge can be characterized in the same module when the extracted features have enough discriminative capability.

3.2. Graph-based Reinforcement Learning

Action Space. Given a network structure S, we regard network layers/blocks as the set of configurable modules $\{s_k\}_1^K \in S$. Taking ResNets [12] as the example, each residual block that is bypassed by identity skip-connections is one module. The configurations of $\{s_k\}_1^K \in S$ indicate taking actions to each module s_k, including early-stop action T_k, keep action D_k or knowledge level selections $\mathbf{a}_k$ among a set of available parent concepts $\Omega_k = \{n_{1,k}, \ldots, n_{M_k,k}\}$ shown as yellow boxes in Figure 3. Specifically, the *early-stop* action corresponds to early-stopping network execution if extracted features are enough to make predictions; the *keep* action just propagates over this module as normal networks; the *knowledge-level selection* action indicates selecting a subset of parent nodes to enforce this module to recognize their children concepts.

Recurrent Policy Network. We develop a recurrent policy network to produce actions for each module. First, a binary policy vector is estimated to indicate *early-stop* or continue; if continue, a multi-choice policy vector is estimated where knowledge levels with larger than 0.5 probabilities are selected while a *keep* action is selected if none of the levels is activated. Unlike standard reinforcement learning, we train the policy to predict all actions of all modules at once. This is essentially a Markov Decision Process (MDP) given the input state and can also be viewed as contextual bandit [18] or associative reinforcement learning [33]. Formally, we denote the recurrent policy network as $F_{\mathbf{W}}$ parameterized by the weights $\mathbf{W}$. $F_{\mathbf{W}}$ sequentially predicts probabilities of all

actions, so that we can sample actions following the graph hierarchy routine. $F_{\mathbf{W}}$ takes the features x_k produced by each module s_k and the updated hidden states h_{k-1} from the previous module s_{k-1} as inputs, and outputs the probabilities for one-dim early-stop action T_k and M_k-dim actions in knowledge level set Ω_k. For each module s_k, we first sample its early-stop action T_k from one-dim Bernoulli distribution:

$$q_k = F_{\mathbf{W}}(x, h_{k-1}), \tag{1}$$

$$\pi_{\mathbf{W}}(T_k|x, h_{k-1}) = q_k^{T_k}(1-q_k)^{1-T_k}, \tag{2}$$

where q_k is its action probability after a Sigmoid function. $T_k = 0$ and $T_k = 1$ indicate network computation is early-stopped at the module s_k or continues respectively. If $T_k = 1$, we then define a policy of modular configuration $\mathbf{a}_k$ as a M_k-dimensional Bernoulli distribution:

$$\Omega_k = \phi(\mathcal{G}, \{\mathbf{a}_1, \ldots, \mathbf{a}_{k-1}\}), \tag{3}$$

$$\mathbf{p}_k = F_{\mathbf{W}}(x, h_{k-1}, \Omega_k), \tag{4}$$

$$\pi_{\mathbf{W}}(\mathbf{a}_k|x, \Omega_k, h_{k-1}) = \prod_{j=1}^{M_k} \mathbf{p}_{k,j}^{\mathbf{a}_{k,j}}(1-\mathbf{p}_{k,j})^{1-\mathbf{a}_{k,j}}, \tag{5}$$

where the available knowledge level set Ω_k is obtained by the graph search functioning ϕ that takes the whole knowledge graph $\mathcal{G}$ and action histories $\{\mathbf{a}_1, \ldots, \mathbf{a}_{k-1}\}$ of previous modules as inputs. The j-th entry of the vector $\mathbf{p}_k \in [0, 1]$ represents the likelihood of its corresponding knowledge level $n_{k,j}$ being activated in this module. The action $\mathbf{a}_k \in \{0, 1\}$ is selected based on $\mathbf{p}_k$. Here, $\mathbf{a}_{k,j} = 1$ and $\mathbf{a}_{k,j} = 0$ indicate this module is responsible for recognizing the child categories (e.g. *cat*,*dog*) for the j-th knowledge level (e.g. *animal*) or not, respectively. If $\sum_j \mathbf{a}_{k,j} = 0$ that indicates none of levels is activated, then *keep* action D_k is selected for this module by treating it as a normal block.

Graph Searching Function. Given a action history $\{\mathbf{a}_1, \ldots, \mathbf{a}_{k-1}\}$ of previous modules and the whole graph $\mathcal{G}$, the graph searching function $\phi(\cdot)$ finds an evolved action space Ω_k, as illustrated in Figure 3. Intuitively, we encourage coarse knowledge levels (e.g. *animals*) are recognized earlier than more fine-grained levels (e.g. *mammal*) since categories in more fine-grained levels require features with more powerful discriminative capabilities in deeper layers. It is noteworthy that not all modules will recognize knowledge levels. Thus, given actions $\{\mathbf{a}_1, \ldots, \mathbf{a}_{k-1}\}$ where some of them may be empty, we denote their accumulated activated knowledge levels Ω'_k, that is, a list of graph nodes are activated. We thus expand Ω'_k into Ω_k by bringing in the 2-hop children nodes of activated graph nodes following $\mathcal{G}$. This graph searching function has several merits: a) each module is also encouraged to recognize knowledge levels that are already addressed in previous modules. Its underlying motivation is that we allow the flexibility for the case that later modules have superior recognition capability with more powerful features; b) new fine-grained knowledge levels are made available for a later module to enforce it embed more elaborated semantics; c) the inclusion of 2-hop children nodes (children and their children) of activated nodes will allow skipping over some useless intermediate nodes in a deeper graph, rather than strictly only activating their immediate children.

Reward Function. We only consider knowledge-level actions before the early-stopping action $T_m = 0$ happens at m-th module, to encourage both correct predictions and minimal module usage, the reward function is computed based on all non-empty actions $\mathbf{a}'$ in $\mathbf{a} = \{\mathbf{a}_k\}$:

$$\hat{\mathcal{Y}} = \frac{1}{|\mathbf{a}'|} \sum_{\mathbf{a}_k \in \mathbf{a}'} \beta(\mathbf{a}_k) F_\Phi(s_k), \tag{6}$$

$$\mathbf{R}(\mathbf{a}, \mathbf{T}) = \lambda(1 - (\frac{m}{K})^2) + \mathcal{A}(\hat{\mathcal{Y}}, \mathcal{Y}), \tag{7}$$

where $\hat{\mathcal{Y}}$ indicates final predictions by averaging the weighted predictions $\beta(\mathbf{a}_k)F_\Phi(s_k)$ of each module s_k. $F_\Phi(\cdot)$ is a linear layer for the classification task or a convolution layer for semantic segmentation task, which takes features of each module as inputs and outputs predictions for $|\mathcal{N}|$ concepts in the graph $\mathcal{G}$. $\beta(\mathbf{a}_k)$ outputs distinct weights for each nodes conditioned on $\mathbf{a}_k$, that is, 1 for concepts under activated knowledge levels and 0.8 for other concepts. Each selected module is thus able to contribute to recognize for all concepts and just gives higher confidences about children concept predictions of its assigned knowledge levels. $\mathcal{A}(\hat{\mathcal{Y}}, \mathcal{Y})$ is the accuracy function between the prediction $\hat{\mathcal{Y}}$ and groundtruth $\mathcal{Y}$. Here, $(\frac{m}{K})^2$ measures the percentage of module utility. A higher reward will be given to the policy that uses less modules and achieves higher accuracy. The hyper-parameter λ controls the trade-off between efficiency (module usage) and accuracy.

Thus, our PMN works as follows: the recurrent policy network $F_{\mathbf{W}}$ decides which knowledge levels are selected for predictions for each module and which module to early-stop network computation. The prediction is generated by running the prediction network F_Φ, conditioning on predicted actions.

3.3. Personalized Modular Network Optimization

To learn the optimal parameters $\mathbf{W}$ of the policy network $F_{\mathbf{W}}$, we maximize the expected reward:

$$J = \mathbb{E}_{\{\mathbf{T}, \mathbf{a}\} \sim \pi_{\mathbf{W}}}[\mathbf{R}(\mathbf{a}, \mathbf{T})]. \tag{8}$$

We utilize policy gradient [33] to compute the gradients of J:

$$\nabla_{\mathbf{W}} J = \mathbb{E}[\mathbf{R}(\mathbf{a}, \mathbf{T}) \nabla_{\mathbf{W}} (\log \pi_{\mathbf{W}}(T_k, \mathbf{a}_k|x, \Omega_k, h_{k-1})], \tag{9}$$

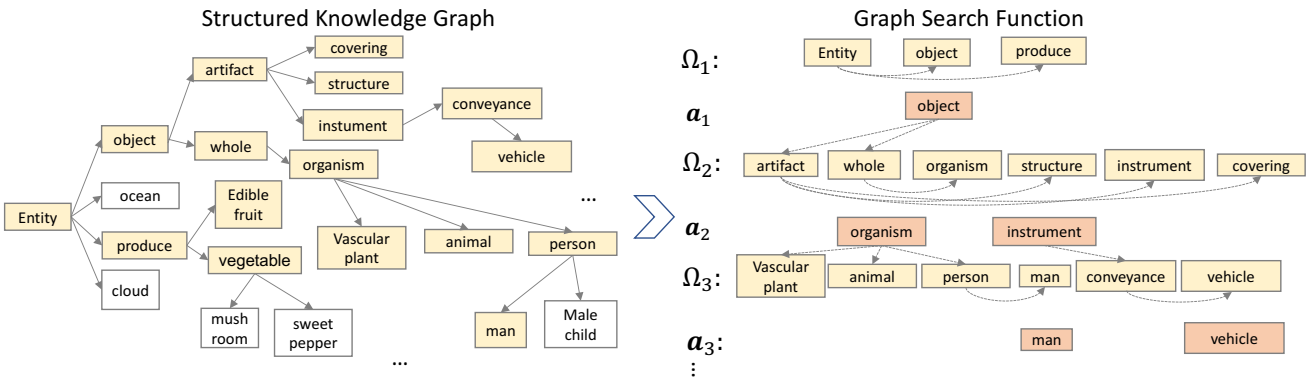

Figure 3. Graph search function which outputs an available knowledge level set Ω_k (i.e. parent concepts), conditioning on activated concepts $\mathbf{a}_{k-1}$ and structured knowledge $\mathcal{G}$.

We approximate the expected gradient in Eqn. 9 with Monte-Carlo sampling using all samples in a mini-batch. These gradient estimates are unbiased, but exhibit high variance. To reduce variance, we utilize a self-critical baseline $\mathbf{R}(\tilde{\mathbf{a}}, \tilde{\mathbf{T}})$ as in [28] and rewrite Eqn. 9 as:

$$\nabla_{\mathbf{W}} J = \mathbb{E}[\mathbf{B} \nabla_{\mathbf{W}} (\log \pi_{\mathbf{W}}(T_k, \mathbf{a}_k | x, \Omega_k, h_{k-1})], \quad (10)$$

where $\mathbf{B} = \mathbf{R}(\mathbf{a}, \mathbf{T}) - \mathbf{R}(\tilde{\mathbf{a}}, \tilde{\mathbf{T}})$, and $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{T}}$ are defined as the maximally probable configuration under the current policy, e.g., $\tilde{\mathbf{a}}_{k,j} = 1$ if $\mathbf{p}_{\mathbf{k},\mathbf{j}} > 0.5$ and otherwise $\tilde{\mathbf{a}}_{k,j} = 0$. We further encourage exploration by introducing a parameter α to bound the probability distribution $\mathbf{q} = \{q_k\}$ and $\mathbf{p} = \{\mathbf{p}_k\}$ and prevent it from saturating, by creating a modified distribution $\mathbf{q}'$ and $\mathbf{p}'$:

$$\mathbf{q}' = \alpha \cdot \mathbf{q} + (1 - \alpha) \cdot (1 - \mathbf{q}) \quad (11)$$

$$\mathbf{p}' = \alpha \cdot \mathbf{p} + (1 - \alpha) \cdot (1 - \mathbf{p}). \quad (12)$$

This bounds the distribution in the range $1 - \alpha \leq \mathbf{p}' \leq \alpha$ and $1 - \alpha \leq \mathbf{q}' \leq \alpha$. Policy gradient methods are typically extremely sensitive to their initialization. We thus first pre-train the prediction network F_Φ using standard supervised learning with cross-entropy loss. Then recurrent policy network F_r and prediction network F_Φ are jointly optimized. The parameters of the prediction network F_Φ are trained with the cross-entropy loss $\mathcal{L}(\hat{\mathcal{Y}}, \mathcal{Y})$ over prediction $\hat{\mathcal{Y}}$ and the groundtruth $\mathcal{Y}$.

4. Experiments

The proposed PMN is generally applicable to various applications. Here we choose the popular image recognition-related tasks as the running examples to validate its superiority. Experiments are conducted on both semantic segmentation tasks on Coco-Stuff [5], Pascal-Context dataset [26] and ADE20K dataset [40], and image-level classification task on CIFAR-10 and CIFAR-100 [17].

Method	Class acc.	acc.	mean IoU
FCN [25]	38.5	60.4	27.2
DeepLabv2 (ResNet-101) [8]	45.5	65.1	34.4
DAG RNN + CRF [31]	42.8	63.0	31.2
OHE + DC + FCN [14]	45.8	66.6	34.3
DSSPN (ResNet-101) [20]	47.0	68.5	36.2
Fixed (16/26)	43.3	64.1	30.9
Random (16/26)	44.1	64.7	31.5
Distribute (16/26)	45.1	64.9	31.4
Policy (18/26)	46.2	65.7	34.9
Policy-adaptive (18/26)	46.3	66.1	35.8
Our PMN (16/26)	**48.1**	**69.9**	**38.4**

Table 1. Comparison on Coco-Stuff test set (%). All our models are based on ResNet-101. "(16/26)" means using 16 modules on average for all test images, compared 26 modules in full ResNet-101.

4.1. Semantic Segmentation

Dataset. We focus on the dense prediction tasks on three public datasets that all aim at recognizing over large-scale categories, which poses more realistic challenges than other small segmentation sets. Specifically, **Coco-Stuff dataset [5]** contains 10,000 complex images from COCO with dense annotations of 91 thing (e.g. book, clock, vase) and 91 stuff classes (e.g. flower, wood, clouds), where 9,000 images are for training and 1,000 for testing. **ADE20k dataset [40]** is annotated with 150 semantic concepts, and includes 20,210 images for training and 2,000 for validation. **PASCAL-Context dataset [26]** consists of 59 object classes and one background, which has 4,998 images for training and 5105 for testing. We use standard evaluation metrics of pixel accuracy (pixAcc) and mean Intersection of Union (mIoU).

Network details. We conduct our experiments using Pytorch framework, 2 GTX TITAN X 12GB cards on a single

Method	mean IoU
FCN [25]	37.8
CRF-RNN [39]	39.3
ParseNet [24]	40.4
BoxSup [9]	40.5
HO CRF [1]	41.3
Piecewise [21]	43.3
VeryDeep [37]	44.5
DeepLab-v2 [8]	45.7
Our PMN (14/26)	**46.8**

Table 2. Comparison on PASCAL-Context test set(%). "(14/26)" means using 14 modules on average for all test images, compared 26 modules in full ResNet-101.

server. We use the Imagenet-pretrained ResNet-101 [12] networks as our pre-defined network architecture following the procedure of [8] and employ *output stride = 8* to give final prediction. Given basic ResNet-101 organized into four segments (i.e., [3, 4, 23, 3]), we treat the later 23 residual blocks and 3 residual blocks as configurable modules, resulting in 26 modules in total that our PMN needs to infer over. Note that features from 3-th ConvBlock and 4-th ConvBlock are with 512-dim and 2048-dim, respectively. In order to employ a recurrent policy, the 2048-dim features from 4-th ConvBlock are first transformed into 512-dim using a linear layer and ReLU layer. The features of each module are passed into global average pooling layer to obtain a 512-dim vector and then fed into the recurrent policy network. Our policy network $F_{\mathbf{W}}$ stacks a LSTM [13] with 512 hidden units and a linear layer that outputs the probabilities of the maximum number of parent concepts and the early-stop action. $F_{\mathbf{W}}$ then samples actions for both early-stop and knowledge-level selection actions in an autoregressive fashion: the hidden states for decoding actions in the previous step are fed as inputs into the next step. At the first step, $F_{\mathbf{W}}$ receives an empty embedding as input. Due to the usage of graph searching function, only probabilities of activated action space will be used for each module. We employ a shared Atrous Spatial Pyramid Pooling (ASSP) [8] module with pyramids of {6,12,18,24} to obtain pixel-wise predictions from 512-dim features for each activated module. Final pixel-wise predictions are produced by averaging weighted predictions from activated modules with their specific attentive knowledge levels, as described in Section 3.2. We set α to 0.8. In terms of reward function, we empirically set λ as 0.5 to balance the strengths of module usage term and prediction accuracy term, and $\mathcal{A}(\hat{\mathcal{Y}}, \mathcal{Y})$ is computed as the pixel accuracy metric.

Structured knowledge construction. In all experiments, a general structured knowledge graph (i.e. concept hierarchy) is employed by combining labels from three datasets, following [20]. Starting from the label hierarchy

Method	mean IoU	pixel acc.
SegNet [2]	21.64	71.00
DilatedNet [38]	32.31	73.55
DeepLabv2 (ResNet-101) [8]	38.97	79.01
DSSPN (ResNet-101) [20]	43.03	81.21
Our PMN (19/26)	**43.80**	**81.33**

Table 3. Comparison on ADE20K val set(%).

Method	mean IoU
PMN w/o overlap (13/26)	35.5
PMN (1-hop children) (22/26)	38.1
PMN w/o weighting (16/26)	37.6
Our PMN (16/26)	**38.4**

Table 4. More ablation studies for our graph search function on Coco-Stuff dataset.

of COCO-Stuff [5] that includes 182 concepts and 26 superclasses, we manually merge concepts from the rest two dataset together by using WordTree as [20]. It results in 340 concepts in the final concept hierarchical graph and its maximal graph depth is twelve.

Training details. We fix the moving means and variations in batch normalization layers of ResNet-101 during finetuning. We optimize the objective function with respect to the weights at all layers by the standard SGD procedure. Inspired by [8], we use the "poly" learning rate policy and set the base learning rate to 2.5e-3 for newly initialized layers and power to 0.9, and set the learning rate as 2.5e-4 for pretrained layers. Our learning procedure has two steps: first pretrain network for 20 epochs by directly appending the ASPP prediction layer on the final block to get good parameter initialization of prediction networks; then jointly train the policy network and prediction network of the whole PMN for 40 epochs on Coco-Stuff and PASCAL-Context, and 100 epochs for ADE20K dataset. Momentum and weight decay are set to 0.9 and 0.0001 respectively. For data augmentation, we adopt random flipping, random cropping and random resize between 0.5 and 2 for all datasets. Due to the GPU memory limitation, the batch size is used 4. The input crop size is set as 513×513 for all datasets. During testing, the actions are selected when its probability is larger than 0.5, different from the Bernoulli sampling used during training.

4.1.1 Comparison with the state-of-the-arts

Table 1, 2, 3 report the result comparison with recent state-of-the-art segmentation models on Coco-Stuff, Pascal-Context and ADE20K dataset, respectively. We regard "DeepLabv2 (ResNet-101) [8]" as our fair baseline. Our PMN can achieve superior performance with less modular usages on all three datasets, which demonstrates the effectiveness of inferring

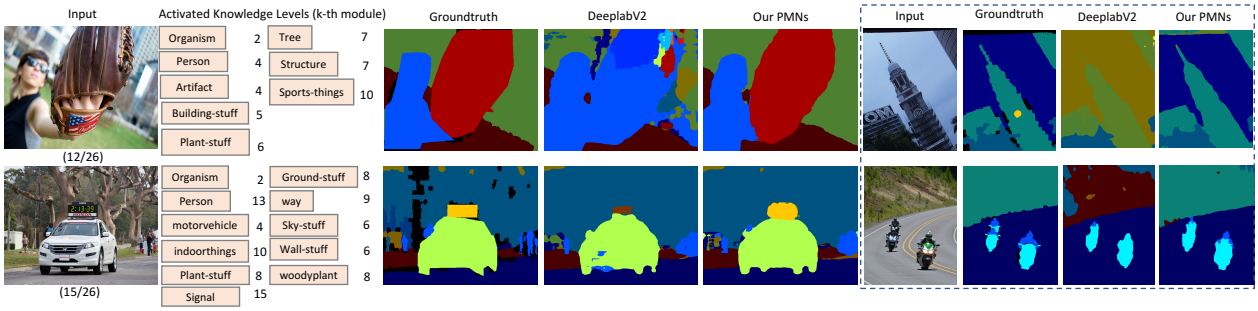

Figure 4. Qualitative comparison results of our PMN with the baseline Deeplabv2 [8]. We also show the activated knowledge levels (parent concepts) for k-th module for left two images. For each activated knowledge level, we only show the earliest module that is selected

meaningful and individualized module configuration for each image guided by structured knowledge. We present the averaged modular usage for test images on each dataset, and can observe it reduces about 40.7% (11/26), 48.2% (13/26) and 29.6% (8/26) module computation in 4-th ConvBlock and 5-th ConvBlock on Coco-Stuff, PASCAL-Context and ADE20k, respectively. The very recent DSSPN [20] includes a network layer for each parent concept, but it is hard to scale up for large-scale concept set and results in redundant predictions for pixels that unlikely belongs to a specific concept. On the contrary, our PMN specifies different computation overheads and designates each activated module to handle more suitable knowledge levels for each input, achieving a good balance between accuracy and efficiency.

More qualitative comparisons are shown in Figure 4. Beside the better prediction performance, our PMN is also able to provide a deeper understanding of feature representation learned in each module. It can be observed that different modules are designated to recognize specific knowledge levels and personalized inference paths are activated for distinct inputs to achieve final prediction.

4.1.2 Ablation Studies

We analysis main contributions of our PMN on the Coco-Stuff dataset, as reported in Table 1 and 4.

Learned policies vs. heuristics. We compare our policy learning with other alternative methods: (1) *Fixed*, which keeps only the first 16 modules active and appends an ASPP prediction layer on the final module; The reason we choose 16 is that it is the resulting averaged module usage by our PMN for fair comparison; (2) *Random*, which keeps 16 randomly selected modules active, and the final prediction is obtained by averaging predictions from ASPP layers on each module; (3) *Distribute*, which evenly distributes 16 modules and averages predictions like Random. We can thus compare different feature combination policies of different modules. The results highlight the advantage of our instance-specific and knowledge-aware policy learning by the proposed PMN.

The effect of structure-knowledge guided policies. We also test policy learning strategies without using external knowledge: 1) *Policy*, which only learns to determine the *early-stop* action of modules, and combines predictions from all modules; 2) *Policy-adaptive*, which decides both the early-stop action and which module should contribute to final predictions. The superior results achieved by our knowledge-guided policy learning verify again the benefits of selecting modules that reveal the properties of specific knowledge levels rather than making the decision only based on final accuracy.

Graph searching function. We further evaluate the importance of key settings in our graph search function which is crucial for policy learning as it depicts the adaptive action space: 1) “PMN w/o overlap" indicates a variant that the activated knowledge levels should not appear in the new action space Ω_k, posed as an aggressive searching strategy. Our results show more prediction refinements from later modules can improve the performance; 2) “PMN (1-hop children)" that only includes 1-hop children of selected actions $\mathbf{a}_{k-1}$ into Ω_k, results in activating more modules; 3) “PMN w/o weighting" directly averages predictions of all activated modules. Our full PMN achieves better performance by adaptively combining different predictions. The underlying reason is that the non-confident predictions on certain categories by specific modules may damage the final performance.

4.2. Image Classification Task

We further evaluate on CIFAR-10 (10 classes) and CIFAR-100 [17] (100 classes), which consist of 60,000 32Ã32 colored images, with 50,000 images for training and 10,000 for testing. Performance is measured by image-level classification accuracy. For fair comparison with the recent dynamic BlockDrop [36] that also explored the dynamic module configuration, we experiment with two ResNets: ResNet-32 and ResNet-110 which start with a convolutional layer followed by 15 and 54 residual blocks, respectively. These residual blocks are treated as configurable modules in

Method (CIFAR-10/-100)	ResNet32	BlockDrop [36]	Our PMN	ResNet101	BlockDrop [36]	Our PMN
Module usage	15/15	6.9/13.1	7.8/13.4	54/54	16.9/30.2	18.4/34.5
Error	7.7/30.7	8.8/31.3	6.9/28.7	6.8/27.8	6.4/26.3	5.9/23.8

Table 5. Comparisons of our PMN with BlockDrop [36] on both CIFAR-10 and CIFAR-100. The first three results are based on ResNet32 and the rest results are on ResNet101.

our PMN. The structured knowledge graph with 148 graph nodes is generated by mapping 100 classes into WordTree, as shown in Supplementary Material. The prediction layer with 10/100 neurons is applied, and all other settings are same with those for segmentation tasks. These models are first pretrained to match state-of-the-art performance on the corresponding datasets when running without our PMN and then jointly trained with our policy network. For training, We use a learning rate of 1e-4, a weight decay of 0.0005 and momentum of 0.9 and train models with a mini-batch size of 128 on two GPUs using a cosine learning rate scheduling [16] for 400 epochs. Observed from Table 5, our PMN outperforms the original ResNet-32 and ResNet-101 models and dynamic network BlockDrop [36] using less or comparable number of modules on average. It demonstrates well the advantages of learning explainable and dynamic module configurations guided by structured knowledge.

5. Conclusion

We presented a novel Personalized Modular Network (PMN) to learn dynamic and personalized inference paths for different inputs, guided by structured knowledge. A graph-based reinforcement learning was proposed to activate an evolved action space for each module routed by the structured knowledge graph, and automatically designates each module to recognize certain knowledge levels. We conducted extensive experiments on both three benchmarks of semantic segmentation and two image classification tasks, observing considerable gains over existing methods in terms of the efficiency and accuracy. Our PMN also learned the dynamic and meaningful inference paths.

References

[1] A. Arnab, S. Jayasumana, S. Zheng, and P. H. Torr. Higher order conditional random fields in deep neural networks. In *ECCV*, pages 524–540, 2016.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *CVPR*, 2015.

[3] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

[4] Y. Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37, 2013.

[5] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. *arXiv preprint arXiv:1612.03716*, 2016.

[6] Q. Cao, X. Liang, B. Li, G. Li, and L. Lin. Visual question reasoning on general dependency tree. *CVPR*, 2018.

[7] S. Chandra, N. Usunier, and I. Kokkinos. Dense and low-rank gaussian crfs using deep embeddings. In *ICCV*, 2017.

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[9] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*, pages 1635–1643, 2015.

[10] L. Denoyer and P. Gallinari. Deep sequential neural network. *arXiv preprint arXiv:1410.0510*, 2014.

[11] M. Figurnov, M. D. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov. Spatially adaptive computation time for residual networks. *arXiv preprint*, 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] H. Hu, Z. Deng, G.-T. Zhou, F. Sha, and G. Mori. Labelbank: Revisiting global perspectives for semantic segmentation. *arXiv preprint arXiv:1703.09891*, 2017.

[15] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. *CoRR, abs/1704.05526*, 3, 2017.

[16] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017.

[17] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[18] J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, pages 817–824, 2008.

[19] X. Liang, L. Lee, and E. P. Xing. Deep variation-structured reinforcement learning for visual relationship and attribute detection. In *CVPR*, 2017.

[20] X. Liang, H. Zhou, and E. Xing. Dynamic-structured semantic propagation network. *CVPR*, 2018.

[21] G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, pages 3194–3203, 2016.

[22] J. Lin, Y. Rao, J. Lu, and J. Zhou. Runtime neural pruning. In *NIPS*, pages 2178–2188, 2017.

[23] L. Liu and J. Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. *AAAI*, 2018.

[24] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[25] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[26] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[27] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[28] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel. Self-critical sequence training for image captioning. *CVPR*, 2017.

[29] D. Schiebener, J. Morimoto, T. Asfour, and A. Ude. Integrating visual perception and manipulation for autonomous learning of object representations. *Adaptive Behavior*, 21(5):328–345, 2013.

[30] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.

[31] B. Shuai, Z. Zuo, B. Wang, and G. Wang. Scene segmentation with dag-recurrent neural networks. *TPAMI*, 2017.

[32] Y.-C. Su and K. Grauman. Leaving some stones unturned: dynamic feature prioritization for activity detection in streaming video. In *ECCV*, pages 783–800, 2016.

[33] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[34] S. Tan, R. Caruana, G. Hooker, and A. Gordo. Transparent model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

[35] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, pages 2464–2469, 2016.

[36] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris. Blockdrop: Dynamic inference paths in residual networks. *CVPR*, 2018.

[37] Z. Wu, C. Shen, and A. v. d. Hengel. Bridging category-level and instance-level semantic image segmentation. *arXiv preprint arXiv:1605.06885*, 2016.

[38] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016.

[39] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, pages 1529–1537, 2015.

[40] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016.