MRF-NAS: Improving UNet with Neural Architecture Search for Medical Image Segmentation

Zifu Wang ZIFU.WANG@ESAT.KULEUVEN.BE
Matthew B. Blaschko MATTHEW.BLASCHKO@ESAT.KULEUVEN.BE
ESAT-PSI, KU Leuven, Belgium

Editors: Under Review for MIDL 2022

Abstract

UNet (Ronneberger et al., 2015) is widely adopted in medical image segmentation due to its simplicity and effectiveness. However, its manually-designed architecture is applied to a large number of problem settings, either with no architecture optimizations, or with manual tuning, which is time consuming and can be sub-optimal. In this work, firstly, we propose Markov Random Field Neural Architecture Search (MRF-NAS) that extends and improves the recent Adaptive and Optimal Network Width Search (AOWS) framework (Berman et al., 2020) with (i) a more general MRF framework (ii) diverse M-best loopy inference (iii) differentiable parameter learning. This provides the necessary NAS framework to accurately and efficiently explore network architectures that induce loopy inference graphs, including loops that arise from skip connections. With UNet as the backbone, we find an architecture, MRF-UNet, that shows several interesting characteristics. Secondly, through the lens of these characteristics, we identify the sub-optimality of the original UNet architecture and further improve our results with MRF-UNetV2. Experiments show that our MRF-UNets significantly outperform several benchmarks while maintaining low computational costs.

Keywords: Neural Architecture Search, Probabilistic Graphical Models, Medical Image Segmentation

1. Introduction

Neural architecture search (NAS) has greatly improved the performance on various vision tasks, for example, classification (Berman et al., 2020; Tan and Le, 2021), object detection (Xu et al., 2019; Liang et al., 2021) and semantic segmentation (Liu et al., 2019a; Weng et al., 2019; Tan et al., 2020; Yan et al., 2020; Wang et al., 2021) via automating the architecture design process. UNet (Ronneberger et al., 2015) is widely adopted in medical image segmentation due to its simplicity and effectiveness. A natural question is, can we improve its manually-designed architecture with NAS?

AOWS (Berman et al., 2020) is a resource-aware NAS method and it is able to find effective architectures that strictly satisfy resource constraints, e.g. latency or MACs. The main idea of AOWS is to model the search problem as parameter learning and MAP inference over a Markov Random Field (MRF). However, the main limitation of AOWS is the adoption of Viterbi inference. As a result, the approach is only applicable to simple tree-structured graphs where the architecture cannot have skip connections (He et al., 2016). Skip connections are widely adopted in modern neural networks as they can ease the training of deep models via shortening effective paths (Veit et al., 2016); skip connections have also played an important role in the success of medical image segmentation where an encoder and a decoder are connected by long skip connections to aggregate features at different levels, e.g. UNet (Ronneberger et al., 2015).

Besides, MAP assignment over a weight-sharing network is usually sub-optimal due to the discrepancy between the one-shot super-network and stand-alone child-networks (Yang et al., 2020; Yu et al., 2020). Restricting the search to a single MAP solution results in the search method having high variance (Liu et al., 2019b), so one usually needs to repeat the search process with different random seeds and hyper-parameters. Furthermore, parameter learning of AOWS is non-differentiable, and this disconnects AOWS from recent advances in the differentiable NAS community (Chu et al., 2021a; Gu et al., 2021; Yang et al., 2021), despite its advantages in representational capacity and efficient inference.

The contributions of this paper are twofold. Firstly, we propose MRF-NAS, which extends and improves AOWS with (i) a more general framework that shows close connections with other NAS approaches and yields better representation capability, (ii) loopy inference algorithms so we can apply it to more complex search spaces, (iii) diverse M-best inference instead of a single MAP assignment to reduce the search variance and to improve search results, (iv) a novel differentiable parameter learning approach with Gibbs sampling and Long-Short-Burnin-Scheme (LSBS) to save on computational cost. With UNet as the backbone, we find an architecture, MRF-UNet, that shows several interesting characteristics. Secondly, through the lens of these characteristics, we identify the sub-optimality of the original UNet architecture and further improve our results with MRF-UNetV2. We show the effectiveness of our approach on two medical image datasets: CHAOS (Kavur et al., 2021) and PROMISE12 (Litjens et al., 2014). Compared with the benchmarks, our MRF-UNets achieve superior performance while maintaining low computational cost.

2. Related works

To reduce the search cost of NAS, one usually uses some proxy to infer an architecture's performance without training it from scratch. For example, the significance of learnable architecture parameters (Liu et al., 2019b; Weng et al., 2019; Yan et al., 2020; Wang et al., 2021) or validation accuracy from a super-network with shared weights (Bender et al., 2018; Guo et al., 2020; Chu et al., 2021b). Resource-aware NAS (Wu et al., 2019; Yu and Huang, 2019a; Weng et al., 2019; Berman et al., 2020; Wan et al., 2020; Yan et al., 2020; Dai et al., 2021) aims to find architectures that satisfy some resource targets such as FLOPs or latency. AutoSlim (Yu and Huang, 2019a) trains a slimmable network (Yu et al., 2019; Yu and Huang, 2019b) as the super-network and applies a greedy approach to search for channel configurations under different FLOPs targets. AOWS (Berman et al., 2020) models resource-aware NAS as a constrained optimization problem which can then be solved via inference over a chain-structured MRF.

3. Preliminaries

3.1. Markov Random Field

For an arbitrary integer n, let $[n]$ be shorthand for $\{1, 2, ..., n\}$. We have a set of discrete variables $\boldsymbol{x} = \{x_i | i \in [n]\}$ and each x_i takes value in a finite label set $X_i = \{x_i^j | j \in [k_i]\}$.

For a set $S \subseteq [n]$, we use x_S to denote $\{x_i | i \in S\}$, and $X_S = \bigtimes_{i \in S} X_i$, where $\bigtimes$ is the cartesian product.

A Markov Random Field (MRF) is an undirected graph $G = (V, E)$ over these variables, and equipped with a set of factors $\Phi = \{\phi_S | S \subseteq [n]\}$ where $\phi_S : X_S \to \mathbb{R}$, such that $V = [n]$ and an edge $e_{i,j} \in E$ when there exists some $\phi_S \in \Phi$ and $\{i, j\} \subseteq S$ (Koller and Friedman, 2009). It is common to employ a pairwise MRF where $\Phi = \{\phi_S | S \subseteq [n] \text{ and } |S| \leq 2\}$.

A set of factors explicitly defines a probabilistic distribution $\mathbb{P}_\Phi(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(\sum_S \phi_S(x_S)\Big)$ where Z is the normalizing constant. The goal of MAP inference is to find an assignment $\boldsymbol{x}^*$ so as to maximize a real-valued energy function $\mathcal{E}(\boldsymbol{x})$:

$$\boldsymbol{x}^* = \operatorname*{argmax}_{\boldsymbol{x} \in X_V} \mathcal{E}(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{x} \in X_V} \exp\Big(\sum_S \phi_S(x_S)\Big). \tag{1}$$

3.2. AOWS

In NAS, there is a neural network N that has n choice nodes, i.e. $\boldsymbol{x} = \{x_i | i \in [n]\}$, and each node x_i can take some value from a label set X_i, e.g. kernel size $= 3$ or 5. Therefore, we can use $\boldsymbol{x}$ to represent the architecture of a neural network. Let $N(\boldsymbol{x})$ be a neural network whose architecture is $\boldsymbol{x}$. Given some task-specific performance measurement $\mathcal{M}$, e.g. classification accuracy, and a resource measurement $\mathcal{R}$, e.g. MACs or latency, resource-aware NAS can be represented as a constrained optimization problem

$$\max_{\boldsymbol{x}} \mathcal{M}(N(\boldsymbol{x})) \quad \text{s.t. } \mathcal{R}(N(\boldsymbol{x})) \leq R_T \tag{2}$$

where R_T is the resource target. Consider the following Lagrangian relaxation of the problem

$$\min_{\gamma} \max_{\boldsymbol{x}} \mathcal{M}(N(\boldsymbol{x})) + \gamma(\mathcal{R}(N(\boldsymbol{x})) - R_T) \tag{3}$$

with γ a Lagrange multiplier. If the inner maximization problem can be solved effiencently, then the minimization problem in (3) can be solved by binary search over γ since the objective is concave in γ (Srivastava et al., 2019; Berman et al., 2020).

The key idea of AOWS (Berman et al., 2020) is to model (3) as parameter learning and MAP inference over a pairwise MRF such that $\mathcal{M}(N(\boldsymbol{x})) = \sum_i \phi_i$ and $\mathcal{R}(N(\boldsymbol{x})) = \sum_{i,j} \phi_{i,j}$. For $\mathcal{M}(N(\boldsymbol{x}))$, they assume $\phi_i(x_i^j) = -\frac{1}{|T_{i,j}|} \sum_t l(\boldsymbol{w} | x_i^{(t)} = j)$ where $l(\boldsymbol{w} | x_i^{(t)} = j)$ is the training loss when $x_i = j$ is sampled at iteration t, and $|T_{i,j}|$ is the total number of times x_i^j is sampled. For $\mathcal{R}(N(\boldsymbol{x}))$, many resource models have a pairwise form. For example, MACs can be calculated exactly as pairwise sums; latency is usually modeled as a pairwise model due to sequential execution of the forward pass. Once these factors are known, the inner maximization problem can be solved efficiently via Viterbi inference.

4. MRF-NAS

Here we generalize the idea of AOWS (Berman et al., 2020) to a broader setting. We assume that there exists some non-decreasing mapping $\mathcal{F} : \mathbb{R} \to \mathbb{R}$ such that

$$\mathcal{M}(N(\boldsymbol{x})) = \mathcal{F}(\mathcal{E}(\boldsymbol{x})) \tag{4}$$

Therefore, we extend their framework and no longer require $\mathcal{M}(N(\boldsymbol{x})) == \mathcal{E}(\boldsymbol{x})$, but let $\mathbb{P}_\Phi$ be defined by a set of factors Φ such that $\mathbb{P}_\Phi(\boldsymbol{x}_1) \geq \mathbb{P}_\Phi(\boldsymbol{x}_2) \Rightarrow \mathcal{M}(N(\boldsymbol{x}_1)) \geq \mathcal{M}(N(\boldsymbol{x}_2))$. Then NAS becomes MAP inference over a set of properly defined factors

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, \mathcal{M}(N(\boldsymbol{x})) = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, \mathcal{E}(\boldsymbol{x}). \tag{5}$$

For resource-aware NAS, we follow (Berman et al., 2020) to introduce another set of factors

$$\mathcal{R}(N(\boldsymbol{x})) = \mathcal{E}'(\boldsymbol{x}) = \exp\Big(\sum_{i\in V}\phi'_i(x_i) + \sum_{(i,j)\in E}\phi'_{i,j}(x_i,x_j)\Big) \tag{6}$$

and combine these two energy functions as in (3). As discussed in 3.2, many resource models can be represented exactly or approximately as a pairwise model. Following (Berman et al., 2020), we focus on latency. We can populate each element ϕ'_i and $\phi'_{i,j}$ through profiling the entire network on some target hardware and solving a system of linear equations.

Many existing methods show close connections to our formulation. For example, one-shot methods with weight sharing (Bender et al., 2018; Guo et al., 2020; Chu et al., 2021b) define a single factor ϕ_V whose scope includes all nodes in the graph where $\phi_V(\boldsymbol{x})$ is the validation accuracy of the super-network evaluated with architecture $\boldsymbol{x}$. Their formulation imposes no factorization, and therefore the cardinality of $\phi_V(\boldsymbol{x})$ grows exponentially in the order of n, the number of nodes in the graph, which makes MAP inference impossible. On the contrary, AOWS (Berman et al., 2020) and differentiable NAS approaches (Liu et al., 2019b; Weng et al., 2019; Yan et al., 2020; Wang et al., 2021) introduce a set of unary factors $\Phi = \{\phi_i | i \in V\}$ such that in AOWS, $\phi_i(x_i)$ is the averaged losses, and in differentiable NAS approaches, $\phi_i(x_i)$ is the learnable architecture parameter. With no higher order interaction, MAP inference deteriorates to marginal maximization whose solution can be derived easily. However, this model imposes strong local independence and greatly limits the representation capability of the underlying graphical model.

Since we model the resource measurement as a pairwise model, for the ease of joint inference in (3), we also consider $\mathcal{E}(\boldsymbol{x})$ to be pairwise

$$\mathcal{E}(\boldsymbol{x}) = \exp\Big(\sum_{i\in V}\phi_i(x_i) + \sum_{(i,j)\in E}\phi_{i,j}(x_i,x_j)\Big). \tag{7}$$

Moreover, compared with methods that only use unary terms, our pairwise model imposes weaker local independence and has more representation power. Although the inclusion of pairwise terms increases the number of learnable factors from $O(|X|)$ to $O(|X|^2)$, where $|X|$ is the cardinality of factors and is usually less than 20, the added overhead is negligible compared with the number in learnable parameters of modern CNNs.

4.1. Diverse M-best loopy inference

The main limitation of AOWS (Berman et al., 2020) is the adoption of Viterbi inference, which is only applicable to simple chain graphs such as MobileNetV1 (Howard et al., 2017). When the computational graph forms loops, i.e. it includes skip connections (He et al., 2016), we need to resort to loopy inference algorithms. Exact inference over a loopy graph can be very expensive, especially when the graph is densely connected. In the worst case, the complexity of exact inference can be exponential in n when the graph is fully connected. Therefore, sometimes we can only hope for approximate solutions. However, we find in practice that for realistic architectures, fast approximate inference yields excellent performance on par with exact inference. The difference between exact and approximate inference will be discussed in more detail in 6.1.

Furthermore, as shown in Yang et al. (2020); Yu et al. (2020), MAP assignment on a weight-sharing network is usually of poor quality, but we can still find architectures that achieve good performance by examining other top solutions. Therefore, instead of a single MAP assignment $\boldsymbol{x}^*$, we use diverse M-best inference (Batra et al., 2012) to find a set of diverse solutions $\{\boldsymbol{x}^1, ..., \boldsymbol{x}^m\}$ so as to reduce the variance in the search phase (preliminaries of diverse M-best inference are in appendix A).

Diverse M-best inference requires a set of balanced dissimilarity constraints, each with an associated Lagrange multiplier (see Eq. (12)). Rather than searching via the diversity constraints, we can directly perform model selection on the Lagrange multiplier λ^q (Batra et al., 2012). We find that the absolute value of ϕ_i can be very different across factors. Instead of using a single scalar value λ^q for all i, we set it to be a vector $\boldsymbol{\lambda}^q = (\lambda_1^q, ..., \lambda_i^q, ..., \lambda_n^q)$ such that

$$\lambda_i^q = \frac{\max_j \phi_i^q(x_i^j) - \min_j \phi_i^q(x_i^j)}{L}, \tag{8}$$

where $\phi_i^q(\cdot)$ is the modified unary factor. Then we can tune L instead.

4.2. Differentiable parameter learning

In the previous sections, we have discussed how to find optimal solutions if the factors in MRF are already known. Here we propose a differentiable approach to learn these factors so as to close the gap between AOWS and other differentiable NAS approaches. Following the formulation in (Ardywibowo et al., 2020), the goal of differentiable NAS is to maximize the following objective

$$-\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_\Phi(\boldsymbol{x})}[l(\boldsymbol{w}^*|\boldsymbol{x})] \quad \text{s.t. } \boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} l(\boldsymbol{w}|\boldsymbol{x}), \tag{9}$$

where $l(\cdot)$ is some loss function and $\boldsymbol{w}$ encodes connection weights of the neural network. In order to make (9) differentiable, we can approximate it through Monte Carlo with n_{mc} samples and use the Gumbel-Softmax reparameterization trick (Jang et al., 2017; Ardywibowo et al., 2020) to smooth the discrete categorical distribution.

However, there is one more caveat in the aforementioned approach: to sample from the joint probability distribution $\mathbb{P}_\Phi(\boldsymbol{x})$. When we only have unary factors such as in Ardywibowo et al. (2020), sampling from $\mathbb{P}_\Phi(\boldsymbol{x})$ is the same as independently sampling from the marginal probability distribution $\mathbb{P}_{\phi_i}(x_i)$:

$$\mathbb{P}_\Phi(\boldsymbol{x}) = \frac{1}{Z}\exp\Big(\sum_i \phi_i(x_i)\Big) = \prod_i \frac{1}{Z}\exp\Big(\phi_i(x_i)\Big) = \prod_i \mathbb{P}_{\phi_i}(x_i). \tag{10}$$

When we have high-order interactions such as pairwise terms, sampling from the joint usually involves the use of Markov Chain Monte Carlo (MCMC) methods. Here we use Gibbs sampling for simplicity:

$$x_i^t \sim \mathbb{P}_\Phi(x_i|\boldsymbol{x}_{-i}) \tag{11}$$

where the distribution of x_i at t-th iteration is determined by $\boldsymbol{x}_{-i}$, all nodes except i. In an MRF, $\boldsymbol{x}_{-i}$ can be simplified to the Markov blanket of i. Since $\mathbb{P}_\Phi(x_i|\boldsymbol{x}_{-i})$ is just the product of several factors, if we apply the Gumbel-Softmax trick, the sampling process is differentiable with respect to these factors. A graphical illustration of the overall procedure is shown in Figure 1 (in the appendix).

After a burn-in period with n_{burnin} samples, Gibbs sampling will converge to the stationary distribution. The length of burn-in period is theoretically unknown and is often decided empirically. Gibbs sampling can be expensive because every time we update Φ, we will have a new $\mathbb{P}_\Phi$. Therefore, we need to re-enter the burn-in period just to draw n_{mc} samples where $n_{\text{mc}} \ll n_{\text{burnin}}$, and then update Φ again. In order to mitigate this problem, we propose a Long-Short-Burnin-Scheme (LSBS) which is similar to the blending learning and inference approach in Chen et al. (2015). Specifically, at the beginning of each epoch, we run a long burn-in period n_{long}, but at each iteration within that epoch, we only run a short burn-in period n_{short}. We can assume that $\mathbb{P}_{\Phi^t} \approx \mathbb{P}_{\Phi^{t+1}}$ since Φ will only change by a small amount. Starting from a sample $\boldsymbol{x}^t \sim \mathbb{P}_{\Phi^t}$, we can quickly transit to a sample $\boldsymbol{x}^{t+1} \sim \mathbb{P}_{\Phi^{t+1}}$ without running a long burnin period. As a result, as opposed to $n_{\text{long}} + n_{\text{mc}}$, we only need to draw $n_{\text{short}} + n_{\text{mc}}$ samples at each iteration, where $n_{\text{mc}} \approx n_{\text{short}} \ll n_{\text{long}}$.

5. Experiments

5.1. Datasets

CHAOS (Kavur et al., 2021) consists of both CT and MRI scans of abdomen organs (liver, right kidney, left kidney and spleen). We only use MRI scans, and it includes 20 cases and 1270 slices. PROMISE12 (Litjens et al., 2014) contains prostate MRI images, and it has 50 cases and 1377 slices.

5.2. Main results

The main results are in Table 1. We set the latency target to 1.70ms which is the same as the original UNet, and we also include MACs for comparison. Our MRF-UNets outperform benchmarks while being simple and latency-friendly.

Model	CHAOS (%)	PROMISE (%)	MACs (G)	Latency (ms)
UNet	83.64	79.84	2.42	1.70
UNet++	84.58	83.05	5.88	5.11
BiO-Net	84.42	82.61	18.61	3.28
NAS-UNet	84.70	82.23	15.22	4.25
MS-NAS	83.76	81.42	12.14	3.96
BiX-Net	80.25	82.35	6.64	1.87
MRF-UNet	85.17	83.32	2.37	1.70
MRF-UNetV2	**85.45**	**83.56**	**2.33**	**1.68**

Table 1: Dice scores on CHAOS and PROMISE.

5.3. MRF-UNet architecture

In Figure 2 (in the appendix), we illustrate the architecture of MRF-UNet. Interestingly, MRF-UNet shows several characteristics that differ from the original UNet: (i) it has a larger encoder but a smaller decoder (ii) layers that are connected by the long skip connections are shallower (iii) layers that need to process these concatenated feature maps are wider and also have larger kernel size. As a result, there exists a bottleneck pattern in the encoder and an inverted bottleneck pattern in the decoder. Our observations show that the encoder and decoder do not need to be balanced as in the original UNet and many other encoder-decoder architectures (Milletari et al., 2016; Chaurasia and Culurciello, 2017). They also demonstrate that the widely adopted "half resolution, double width" principle might be sub-optimal in an encoder-decoder network. Indeed, feature maps are concatenated by the long skip connections and are processed by the following layer, which form the most computationally extensive part in the whole network. Therefore, their widths should be smaller to reduce complexity. However, layers that need to process this rich information should be wider and have larger receptive fields.

Inspired by these observations, in Figure 3 (in the appendix) we propose MRF-UNetV2 that emphasizes these characteristics. We also note that a recent trend in NAS is to design a more and more complex search space to include as many candidates as possible, but it becomes difficult to interpret the search results. We hope that our observations can inspire practitioners when designing other encoder-decoder architectures.

6. Ablation study

6.1. Exact vs. approximate loopy inference

Inference on a loopy MRF in general is NP-hard, so we cannot always hope for exact solutions. Here we use Max-Product Clique Tree algorithm (MPCT) for exact solutions, and Max-Product Linear Programming (MPLP) for approximate inference (Koller and Friedman, 2009). We compare architectures found by MPCT (exact inference) and MPLP (approximate inference) in Table 2. They usually obtain very similar solutions and their results are almost identical.

The complexity of MPCT is $O(|X|^{|C|})$ where $|X| = 10$ is the cardinality of factors and $|C|$ is the size of the largest clique. Generally, $|C|$ increases when the graph is more densely connected, and in the worst case $|C| = n$ the number of nodes when the graph is fully connected, e.g. DenseNet (Huang et al., 2017). In Figure 5 (in the appendix), we show the size of the largest clique vs. the number nodes for UNet (Ronneberger et al., 2015), UNet+ and UNet++ (Zhou et al., 2020). UNet++, being more densely connected, has a much larger clique size than UNet. The clique size of the original UNet is 5, and MPCT takes several minutes on our recent Macbook Pro. Note that we need to repeat the inference for $m \times n_{\text{iter}} = 100$ times, and it will soon become infeasible if we want to apply MPCT on a deeper UNet or on UNet+/UNet++. On the other hand, MPLP can converge within a few seconds. Since they usually find similar solutions, we use MPLP as a default.

6.2. Diverse solutions

As shown in Yang et al. (2020); Yu et al. (2020), there exists an inconsistency between the true rank of an architecture and its rank on a weight-sharing network, but we can still find architectures that are reasonably good by examining other top solutions. This motivates us to apply diverse M-best inference (Batra et al., 2012). In Figure 6 (in the appendix), we show the results of diverse 5-best. The MAP solution is usually not the best quality and diverse M-best inference can greatly improve our results. It also helps reduce the variance in the search phase since we can evaluate m highly probable solutions at the same time, and the total computational cost is thus decreased from $m \times (\text{cost}_{\text{search}} + \text{cost}_{\text{eval}})$ to $\text{cost}_{\text{search}} + m \times \text{cost}_{\text{eval}}$.

Calibration (Guo et al., 2017) is critical for medical diagnosis and deep ensembles (Lakshminarayanan et al., 2017) have been shown to effectively reduce the calibration error. Compared with a deep ensemble that consists of the same architecture from different initialization, we can form an ensemble with a diverse set of solutions. As shown in Table 3, our diverse ensemble achieves a lower Expected Calibration Error (ECE) on CHAOS.

Should we further increase m if it is so helpful? The answer is no: further increasing m generally does not help us to find a better solution, and the majority of solutions are often of sub-optimal performance so it does not help reduce the variance. In Figure 6 (in the appendix), we show the result of diverse 10-best. We choose a higher $L = 20$ and expect that the best solution will come later, since it leads to a lower dissimilarity target in (8). Indeed, the best architecture is now the 4th one instead of the 3rd, but it does not show a better performance and many solutions are of similar quality. Therefore, we do not benefit from increasing m, while it adds the cost of both inference and evaluation.

Algorithm	MPCT	MPLP
Dice (%)	85.10	**85.17**

Table 2: Evaluating architectures found by MPCT and MPLP on CHAOS.

Model	Deep Ensemble	Diverse Deep Ensemble
ECE (%)	0.7732	**0.7281**

Table 3: Comparing deep ensemble (Lakshminarayanan et al., 2017) with our diverse deep ensemble on CHAOS. Lower is better.

7. Conclusion

In this paper, we propose MRF-NAS that extends and improves AOWS (Berman et al., 2020) with a more general framework based on a pairwise MRF formulation. With diverse M-best loopy inference algorithms and differentiable parameter learning, we find an architecture, MRF-UNet, with several interesting characteristics. Through the lens of these characteristics, we identify the sub-optimality of the original UNet architecture and further improve our results with MRF-UNetV2.

Acknowledgements

We acknowledge support from the Research Foundation - Flanders (FWO) through project number G0A1319N, and funding from the Flemish Government under the Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen programme.

References

Randy Ardywibowo, Shahin Boluki, Xinyu Gong, Zhangyang Wang, and Xiaoning Qian. NADS: Neural Architecture Distribution Search for Uncertainty Awareness. *ICML*, 2020.

Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse M-best solutions in Markov random fields. *ECCV*, 2012.

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. *ICML*, 2018.

Maxim Berman, Leonid Pishchulin, Ning Xu, Matthew Blaschko, and Gerard Medioni. AOWS: Adaptive and Optimal Network Width Search with Latency Constraints. *CVPR*, 2020.

Abhishek Chaurasia and Eugenio Culurciello. LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation. *VCIP*, 2017.

Liang-Chieh Chen, Alexander G Schwing, Alan L Yuille, and Raquel Urtasun. Learning deep structured models. *ICML*, 2015.

Xiangxiang Chu, Xiaoxing Wang, Bo Zhang, Shun Lu, Xiaolin Wei, and Junchi Yan. DARTS-: Robustly Stepping out of Performance Collapse Without Indicators. *ICLR*, 2021a.

Xiangxiang Chu, Bo Zhang, and Ruijun Xu. FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. *ICCV*, 2021b.

Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, and Joseph E Gonzalez. FBNetV3: Joint Architecture-Recipe Search using Predictor Pretraining. *CVPR*, 2021.

Ilke Demir, Krzysztof Koperski, David Lindenbaum, Guan Pang, Jing Huang, Saikat Basu, Forest Hughes, Devis Tuia, and Ramesh Raskar. DeepGlobe 2018: A Challenge to Parse the Earth through Satellite Images. *CVPR Workshop*, 2018.

Xuanyi Dong and Yi Yang. One-shot neural architecture search via self-evaluated template network. *ICCV*, 2019.

Yu-Chao Gu, Li-Juan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. DOTS: Decoupling Operation and Topology in Differentiable Architecture Search. *CVPR*, 2021.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *ICML*, 2017.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *ECCV*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016.

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*, 2017.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *CVPR*, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with Gumbel-softmax. *ICLR*, 2017.

A. Emre Kavur, N. Sinem Gezer, Mustafa Barış, Sinem Aslan, Pierre-Henri Conze, Vladimir Groza, Duc Duy Pham, Soumick Chatterjee, Philipp Ernst, Savaş Özkan, Bora Baydar, Dmitry Lachinov, Shuo Han, Josef Pauli, Fabian Isensee, Matthias Perkonigg, Rachana Sathish, Ronnie Rajan, Debdoot Sheet, Gurbandurdy Dovletov, Oliver Speck, Andreas Nürnberger, Klaus H. Maier-Hein, Gözde Bozdağı Akar, Gözde Ünal, Oğuz Dicle, and M. Alper Selver. CHAOS Challenge - combined (CT-MR) healthy abdominal organ segmentation. *MIA*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: Principles and Techniques.* MIT press, 2009.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 2017.

Tingting Liang, Yongtao Wang, Zhi Tang, Guosheng Hu, and Haibin Ling. OPANAS: One-Shot Path Aggregation Network Architecture Search for Object Detection. *CVPR*, 2021.

Geert Litjens, Robert Toth, Wendy van de Ven, Caroline Hoeks, Sjoerd Kerkstra, Bram van Ginneken, Graham Vincent, Gwenael Guillard, Neil Birbeck, Jindang Zhang, Robin Strand, Filip Malmberg, Yangming Ou, Christos Davatzikos, Matthias Kirschner, Florian Jung, Jing Yuan, Wu Qiu, Qinquan Gao, Philip "Eddie" Edwards, Bianca Maan, Ferdinand van der Heijden, Soumya Ghose, Jhimli Mitra, Jason Dowling, Dean Barratt, Henkjan Huisman, and Anant Madabhushi. Evaluation of prostate segmentation algorithms for MRI: The PROMISE12 Challenge. *MIA*, 2014.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan Yuille, and Li Fei-Fei. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. *CVPR*, 2019a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable Architecture Search. *ICLR*, 2019b.

Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. *3DV*, 2016.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *MICCAI*, 2015.

Shivangi Srivastava, Maxim Berman, Matthew B Blaschko, and Devis Tuia. Adaptive compression-based lifelong learning. *BMVC*, 2019.

Mingxing Tan and Quoc V Le. EfficientNetV2: Smaller Models and Faster Training. *ICML*, 2021.

Mingxing Tan, Ruoming Pang, and Quoc V Le. EfficientDet: Scalable and Efficient Object Detection. *CVPR*, 2020.

Andreas Veit, Michael Wilber, and Serge Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. *NeurIPS*, 2016.

Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E Gonzalez. FBNetV2: Differentiable Neural Architecture Search for Spatial and Channel Dimensions. *CVPR*, 2020.

Xinyi Wang, Tiange Xiang, Chaoyi Zhang, Yang Song, Dongnan Liu, Heng Huang, and Weidong Cai. BiX-NAS: Searching Efficient Bi-directional Architecture for Medical Image Segmentation. *MICCAI*, 2021.

Yu Weng, Tianbao Zhou, Yujie Li, and Xiaoyu Qiu. NAS-Unet: Neural Architecture Search for Medical Image Segmentation. *IEEE Access*, 2019.

Bichen Wu, Kurt Keutzer, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, and Yangqing Jia. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. *CVPR*, 2019.

Tiange Xiang, Chaoyi Zhang, Dongnan Liu, Yang Song, Heng Huang, and Weidong Cai. BiO-Net: Learning Recurrent Bi-directional Connections for Encoder-Decoder Architecture. *MICCAI*, 2020.

Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-FPN: Automatic Network Architecture Adaptation for Object Detection Beyond Classification. *ICCV*, 2019.

Xingang Yan, Weiwen Jiang, Yiyu Shi, and Cheng Zhuo. MS-NAS: Multi-Scale Neural Architecture Search for Medical Image Segmentation. *MICCAI*, 2020.

Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. NAS evaluation is frustratingly hard. *ICLR*, 2020.

Yibo Yang, Shan You, Hongyang Li, Fei Wang, Chen Qian, and Zhouchen Lin. Towards improving the consistency, efficiency, and flexibility of differentiable neural architecture search. *CVPR*, 2021.

Jiahui Yu and Thomas Huang. AutoSlim: Towards One-Shot Architecture Search for Channel Numbers. *NeurIPS Workshop*, 2019a.

Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. *ICCV*, 2019b.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *ICLR*, 2019.

Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. *ICLR*, 2020.

Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *TMI*, 2020.

Appendix A. Diverse M-best inference

In MRFs, there exist optimization error (approximate inference), approximation error (limitations of the model, e.g. a pairwise MRF can only represent pairwise interactions), and estimation error (factors are learnt from a finite dataset). In the context of NAS, in order to reduce the cost of searching, we often resort to proxies on a weight-sharing network (Guo et al., 2020; Dong and Yang, 2019) and they can be inaccurate. Instead of giving all our hope to a single MAP solution, diverse M-best inference (Batra et al., 2012) aims to find a diverse set of highly probable solutions.

Given some dissimilarity function $\Delta(\boldsymbol{x}^p, \boldsymbol{x}^q)$ between two solutions and a dissimilarity target k^q, we denote $\boldsymbol{x}^1$ as the MAP, $\boldsymbol{x}^2$ the second-best solution and so on until $\boldsymbol{x}^m$ the mth-best solution. Then for each $2 \leq p \leq m$, we have the following constrained optimization problem

$$\boldsymbol{x}^p = \underset{\boldsymbol{x} \in X_V}{\operatorname{argmax}} \, \mathcal{E}(\boldsymbol{x}) \tag{12}$$

$$\text{s.t. } \Delta(\boldsymbol{x}^p, \boldsymbol{x}^q) \geq k^q \quad \text{for } q = 1, ..., p-1. \tag{13}$$

Therefore, we are interested in a diverse set of solutions $\{\boldsymbol{x}^1, ..., \boldsymbol{x}^p, ..., \boldsymbol{x}^m\}$ such that each $\boldsymbol{x}^p$ maximizes the energy function, and is at least k^q-units away from each of the $p-1$ previously found solutions. It is well known (Batra et al., 2012) that if we consider a pairwise MRF and choose Hamming distance as the dissimilarity function, we can turn (12) into a new MAP problem such that pairwise factors remain the same and unary factors become $\phi_i^p(x_i^j) = \phi_i(x_i^j) - \sum_{q=1}^{p-1} \lambda^q \cdot \mathbb{1}(x_i^{;q} = x_i^j)$ where λ^q is the Lagrange multiplier and $x_i^{;q}$ is the assignment of x_i in the q-th solution.

Appendix B. Experimental setups

B.1. Search Space

We use UNet (Ronneberger et al., 2015) as the backbone. Generally speaking, UNet and other encoder-decoder networks usually contain three types of operations: Normal, Down and Up. We search for both size and width of the convolution kernel and summarize the search space in Table 4. We have about 4×10^{23} configurations in total.

For a fair comparison, for manually designed architectures, e.g. UNet (Ronneberger et al., 2015), UNet++ (Zhou et al., 2020) and BiO-Net (Xiang et al., 2020), we use their templates and implement the Normal/Up/Down operations in the same way as our search space, but fix the kernel size to be 3. For automatically found architectures, e.g. NAS-UNet (Weng et al., 2019), MS-NAS (Yan et al., 2020), and BiX-Net (Wang et al., 2021), we do not make any modification and use their implementations directly. However, there exist discrepancies. For instance, we use transposed convolution for Up operation while NAS-UNet (Weng et al., 2019) uses dilated transposed convolution and BiX-Net (Wang et al., 2021) uses bilinear interpolation.

B.2. Implementation Details

For the latency model, we use a NVIDIA GeForce RTX 3090. We first measure the real-time latency using input tensor of size $(batch\_size, 3, 256, 256)$ with $batch\_size = 16$. After

Type	Size	Width	Cardinality
Normal	size = [3, 5], stride = 1	[0.5, 0.75, 1.0, 1.25, 1.5]	10
Down	size = 3, stride = 2	[0.5, 0.75, 1.0, 1.25, 1.5]	5
Up	transpose size = 2, stride = 2	[0.5, 0.75, 1.0, 1.25, 1.5]	5

Table 4: Search space with UNet as the backbone.

collecting 10000 random samples where each experiment is repeated for 60 times, we use the first 5000 samples to build the latency model and the remaining samples to evaluate the model. We achieve a mean absolute error of 3%.

In the search phase, we train a super-network using the sandwich rule (Yu and Huang, 2019b) for $T = 50$ epochs. Initially, factors are not updated until the super-network is trained for a warmup period of 10 epochs. Following, the learning rate of weights starts from 0.0005 and is then decreased by a factor of $(1 - \frac{t}{T})^{0.9}$ at each epoch t. We use the Adam optimizer (Kingma and Ba, 2015) with weight decay of 0.0001. The learning rate of factors is fixed at 0.0003 and we also use the Adam optimizer with the same weight decay. For the sampling part, we use $n_{\text{long}} = 10000$, $n_{\text{short}} = 10$ and $n_{\text{mc}} = 1$. The temperature parameter τ in Gumbel-Softmax is fixed at 1. For simplicity, we search on Deepglobe Land (Demir et al., 2018) and the found architectures are evaluated on other medical datasets. Our results can be improved by searching on each dataset individually. In the re-train phase, we use the same hyper-parameters as in the search phase, except that we train for $T = 100$ epochs. We use the same hyper-parameters for both architectures found by our methods as well as the baselines.

As for inference, we choose $m = 5$ and $L = 10$ for diverse M-best inference, and the number of iterations in binary search $n_{\text{iter}} = 20$.

B.3. Computational cost

The overhead of our method comes from Gibbs sampling and inference over a complex loopy graph. Since we run a long burn-in period n_{long} only at the start of each epoch, and a short burn-in period $n_{\text{short}} + n_{\text{mc}}$ at each training iteration, the cost of sampling is negligible compared with a forward-backward pass of the neural network. We have discussed the complexity of loopy inference algorithms in 6.1, and since we use MPLP as a default, the cost of inference is also minimal. In our experiments, the overhead takes up less than 2% of the total search time.

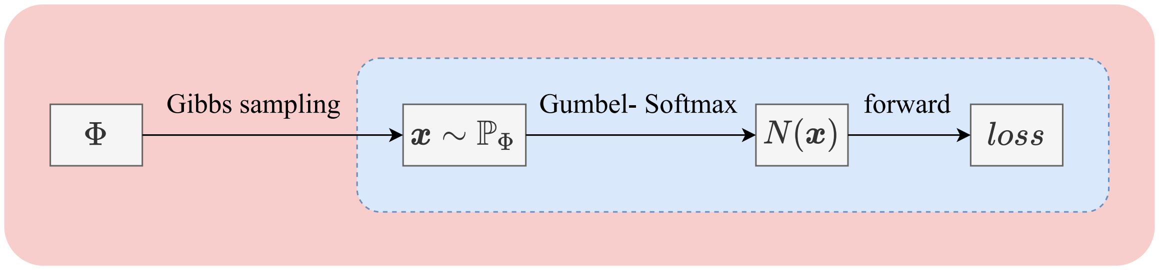

Figure 1: Workflow of our differentiable parameter learning approach. Vanilla differentiable NAS methods (Ardywibowo et al., 2020) with Monte Carlo approximation and Gumbel-Softmax trick are shown in the blue rectangle. In our case, since the joint probability distribution $\mathbb{P}_\Phi(\boldsymbol{x})$ cannot be decomposed as the product of only unary terms, we need to perform an extra step of MCMC, e.g. Gibbs sampling. Best viewed in color.

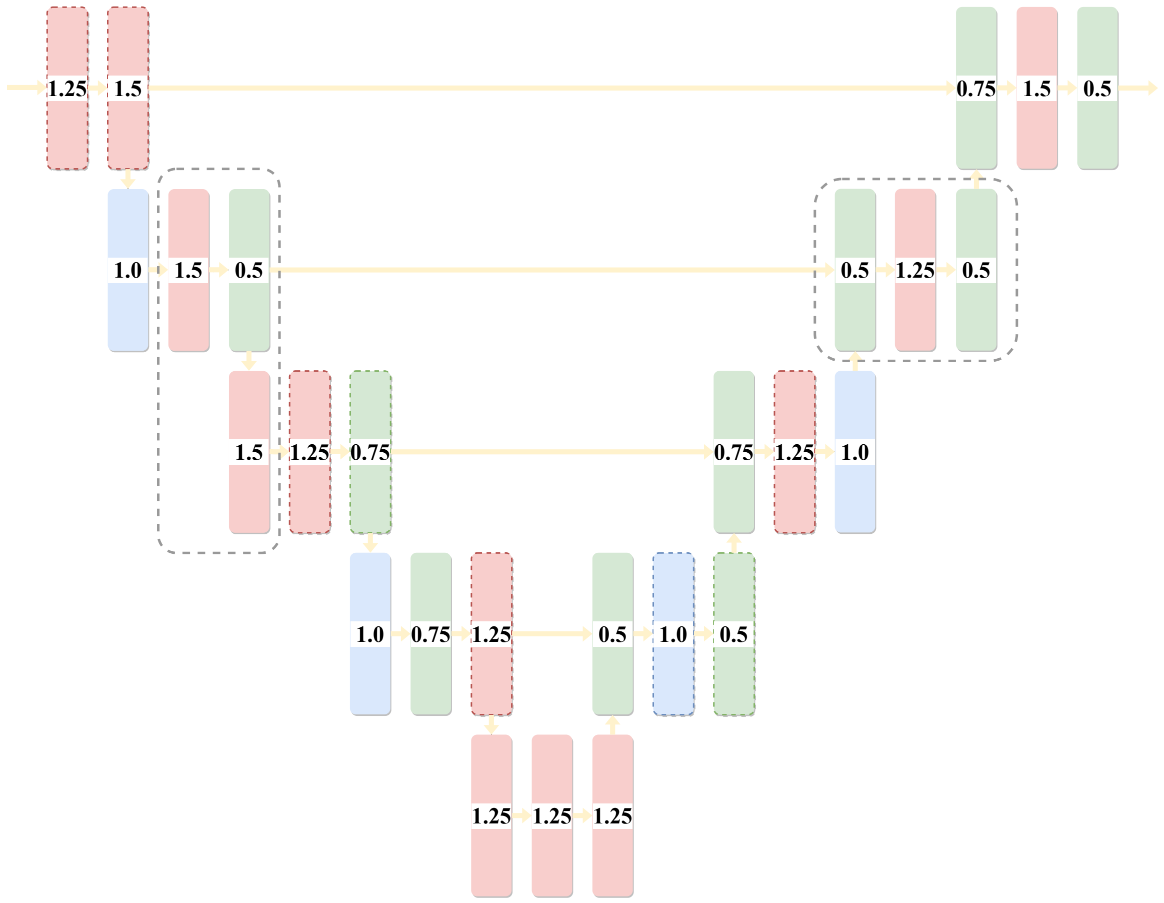

Figure 2: Architecture of MRF-UNet. Numbers inside rectangles are width ratios to the original UNet. Rectangles surrounded by colored dashed lines use 5×5 kernels while others use 3×3 kernels. The gray dashed line on the left highlights an example of a bottleneck block in the encoder, and the gray dashed line on the right shows an example of an inverted bottleneck block in the decoder. Best viewed in color.

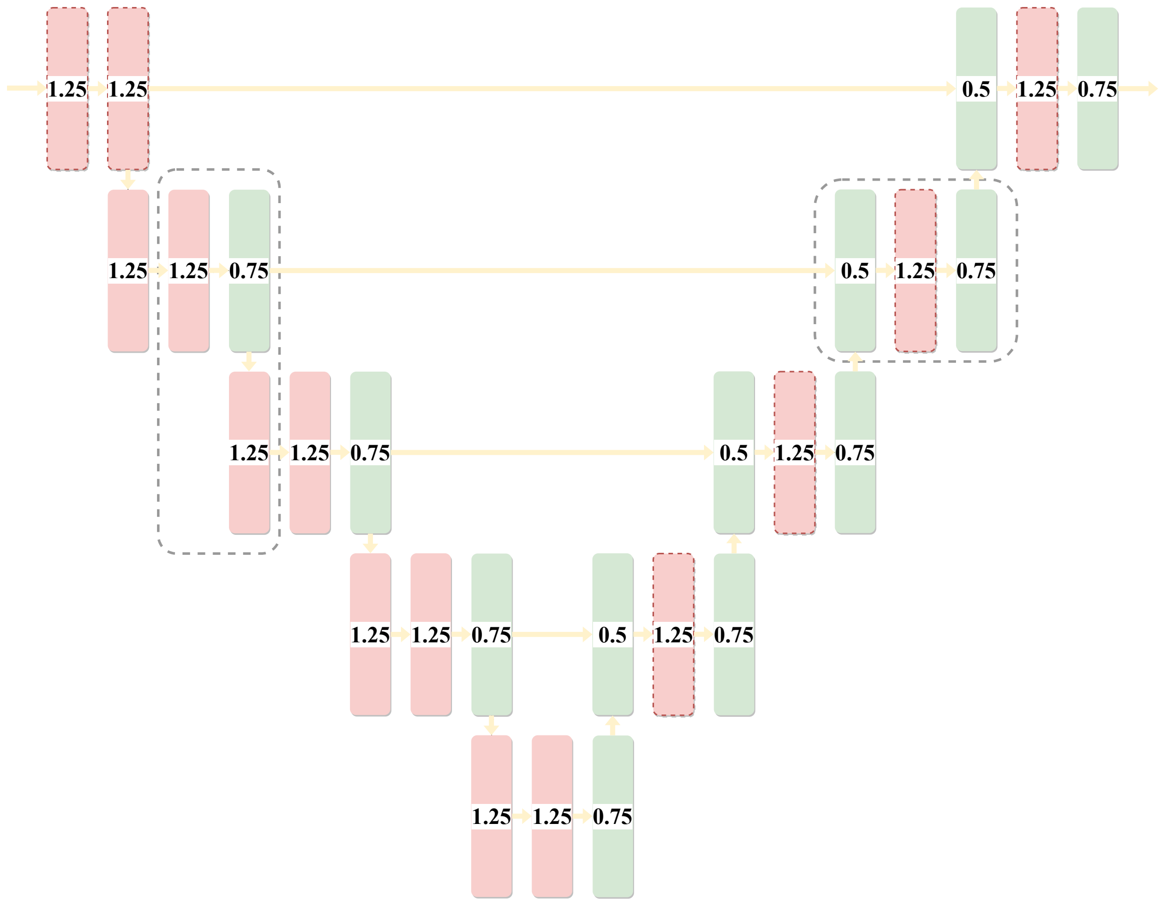

Figure 3: Architecture of MRF-UNetV2. Numbers inside rectangles are width ratios to the original UNet. Rectangles surrounded by colored dashed lines use 5×5 kernels while others use 3×3 kernels. The gray dashed line on the left highlights an example of a bottleneck block in the encoder, and the gray dashed line on the right shows an example of an inverted bottleneck block in the decoder. Best viewed in color.

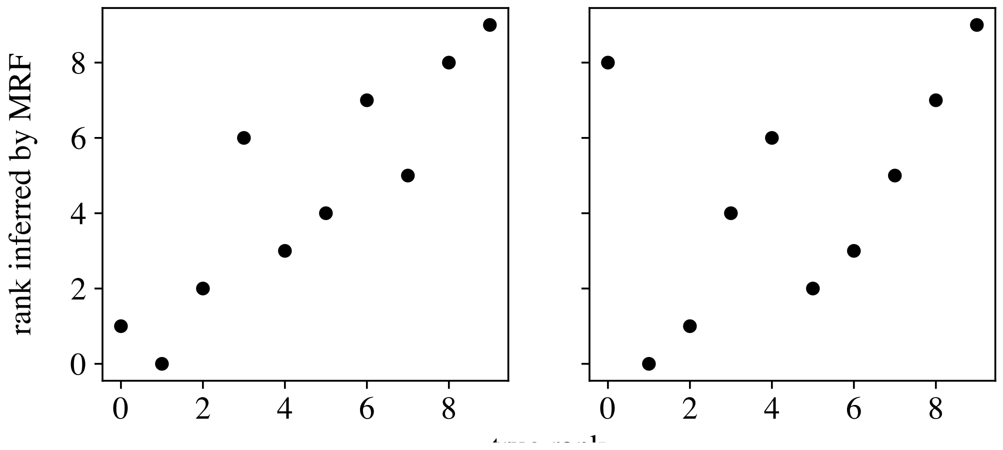

Figure 4: Correlation between the true rank and rank inferred by a pairwise MRF (left) vs. a unary MRF (right) on 10 architectures in our search space.

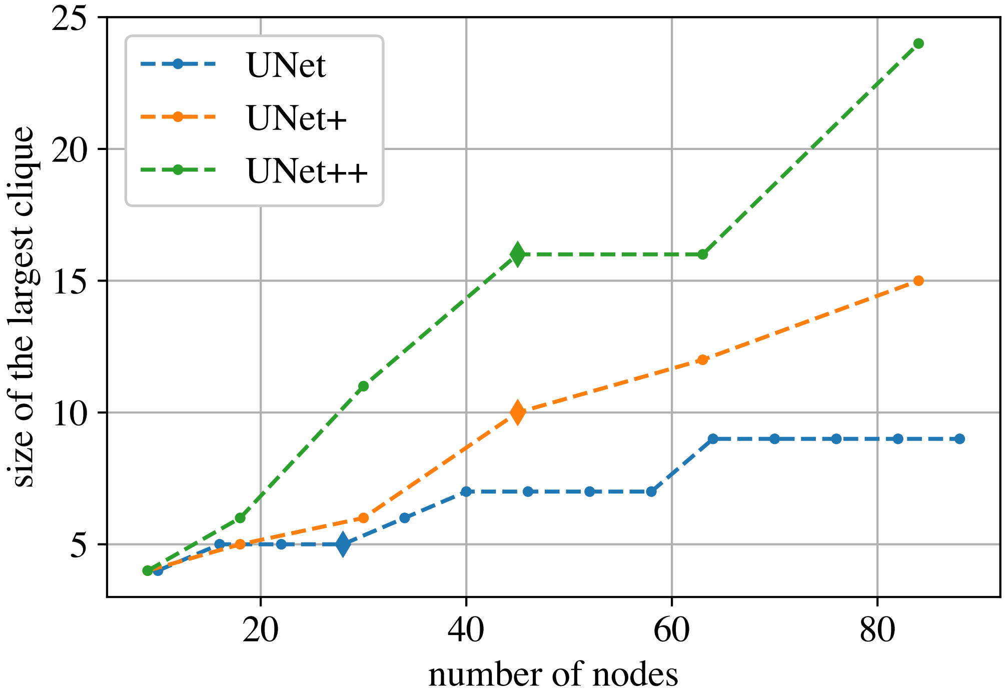

Figure 5: Size of the largest clique vs. the number of nodes for UNet (Ronneberger et al., 2015), UNet+ (Zhou et al., 2020) and UNet++ (Zhou et al., 2020); diamonds indicate the original depth. Best viewed in color.

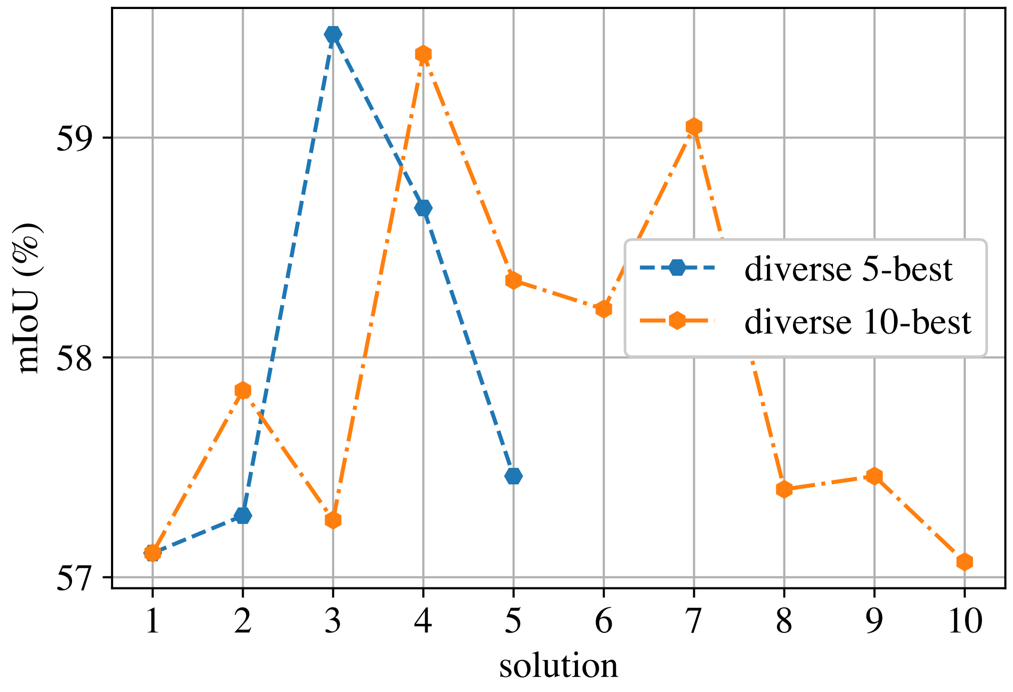

Figure 6: Evaluating solutions obtained from diverse 5-best and diverse 10-best inference. Best viewed in color.