MRF-UNets: Searching UNet with Markov Random Fields

Zifu Wang ✉ and Matthew B. Blaschko

ESAT-PSI, KU Leuven, Leuven, Belgium
zifu.wang@kuleuven.be

Abstract. UNet [27] is widely used in semantic 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) method [4] with (i) a more general MRF framework (ii) diverse M-best loopy inference (iii) differentiable parameter learning. This provides the necessary NAS framework to 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 on three aerial image datasets and two medical image datasets while maintaining low computational costs. The code is available at: https://github.com/zifuwanggg/MRF-UNets.

Keywords: Neural Architecture Search, Probabilistic Graphical Models, Semantic Segmentation

1 Introduction

Neural architecture search (NAS) has greatly improved the performance on various vision tasks, for example, classification [4,8,30], object detection [34,23,10] and semantic segmentation [32,35,31,18] via automating the architecture design process. UNet [27] is widely adopted to a large number of problem settings such as aerial [9,28] and medical image segmentation [24,11,20] due to its simplicity and effectiveness, but either with no architecture optimization, or with simple manual tuning. A natural question is, can we improve its manually-designed architecture with NAS?

AOWS [4] is a resource-aware NAS method and it is able to find effective architectures that strictly satisfy resource constraints, e.g. latency or the number of floating-point operations (FLOPs). The main idea of AOWS is to model the search problem as parameter learning and maximum a posteriori (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 [14]. Skip connections are widely adopted in modern neural networks as they can ease the training of deep models via shortening effective paths [29]; skip connections have also played an important role in the success of semantic segmentation where an encoder and a decoder are connected by long skip connections to aggregate features at different levels, e.g. UNet.

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 [36,42]. Restricting the search to a single MAP solution also results in the search method having high variance [25], 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 [25,7,37], despite its advantages in 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 and (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 three aerial image datasets: DeepGlobe Land, Road and Building [9] and two medical image datasets: CHAOS [20] and PROMISE12 [24]. Compared with the benchmarks, our MRF-UNets achieve superior performance while maintaining low computational cost.

2 Related Works

Neural architecture search (NAS) [44,25] is a technique for automating the design process of neural network architectures. The early attempt [44] trains a RNN with reinforcement learning and costs thousands of GPU hours. In order to reduce the search cost, one usually uses some proxy to infer an architecture's performance without training it from scratch. For example, the significance of learnable architecture parameters [25,7,37,32,35,15,31,18] or validation accuracy from a super-network with shared weights [3,4,13,6].

Resource-aware NAS [38,4,40,8] focuses on architectures that achieve good performance while satisfy resource targets such as FLOPs or latency. FBNet [8] inserts a differentiable latency term into the loss function to penalize networks that consume high latency. However, the found architectures are not guaranteed to strictly satisfy the constraints. AutoSlim [38] trains a slimmable network [41,39] as the super-network and applies a greedy heuristics to search for channel configurations under different FLOPs targets. AOWS [4] models resource-aware NAS as a constrained optimization problem which can then be solved via inference over a chain-structured MRF. Nevertheless, their method can only be applied to simple search spaces which do not include skip connections.

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 $\varPhi = \{\phi_S | S \subseteq [n]\}$ where $\phi_S : X_S \to \mathbb{R}$, such that $V = [n]$ and an edge $e_{i,j} \in E$ when there exists some $\phi_S \in \varPhi$ and $\{i, j\} \subseteq S$ [21]. It is common to employ a pairwise MRF where $\varPhi = \{\phi_S \big| S \subseteq [n] \text{ and } |S| \leq 2\}$. A set of factors $\varPhi$ explicitly defines a probabilistic distribution $\mathbb{P}_{\varPhi}(\boldsymbol{x}) = \frac{1}{Z} \exp\Big(\sum_S \phi_S(x_S)\Big)$ 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 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 [3,25] on a weight-sharing network and they can be inaccurate. Instead of giving all our hope to a single MAP solution, diverse M-best inference [2] aims to find a diverse set of highly probable solutions.

Given some dissimilarity function $\varDelta(\boldsymbol{x}^p, \boldsymbol{x}^q)$ 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

$$\boldsymbol{x}^p = \operatorname*{argmax}_{\boldsymbol{x} \in X_V} \mathcal{E}(\boldsymbol{x}) \tag{2}$$

$$\text{s.t. } \varDelta(\boldsymbol{x}^p, \boldsymbol{x}^q) \geq k^q \quad \text{for } q = 1, ..., p-1. \tag{3}$$

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. If we consider a pairwise MRF and choose Hamming distance as the dissimilarity function, we can turn Eq. (2) 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 [2].

3.3 AOWS

Having introduced the notations for MRFs, here we illustrate how AOWS models the NAS problem as a MRF. 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. latency or FLOPs, resource-aware NAS can be represented as a constrained optimization problem

$$\max_{\boldsymbol{x}} \mathcal{M}(N(\boldsymbol{x})) \quad \text{s.t. } \mathcal{R}(N(\boldsymbol{x})) \leq R_T \tag{4}$$

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

$$\min_{\gamma} \max_{\boldsymbol{x}} \mathcal{M}(N(\boldsymbol{x})) + \gamma(\mathcal{R}(N(\boldsymbol{x})) - R_T) \tag{5}$$

with γ a Lagrange multiplier. If the inner maximization problem can be solved effienciently, then the minimization problem in Eq. (5) can be solved by binary search over γ since the objective is concave in γ [28,4].

The key idea of AOWS [4] is to model Eq. (5) 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, FLOPs 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 [4] to a broader setting. We assume that there exists some non-decreasing mapping $\mathcal{F} : \mathbb{R} \to \mathbb{R}$ such that

$$\mathcal{M}(N(\boldsymbol{x})) = \mathcal{F}(\mathcal{E}(\boldsymbol{x})) \tag{6}$$

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

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, \mathcal{M}(N(\boldsymbol{x})) = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, \mathcal{E}(\boldsymbol{x}). \tag{7}$$

For resource-aware NAS, we follow [4] to introduce another set of factors

$$\mathcal{R}(N(\boldsymbol{x})) = \mathcal{E}'(\boldsymbol{x}) = \exp\Big(\sum_{i\in V} \phi'_i(x_i) + \sum_{(i,j)\in E} \phi'_{i,j}(x_i, x_j)\Big) \tag{8}$$

and combine these two energy functions as in Eq. (5). As discussed in section 3.3, many resource models can be represented exactly or approximately as a pairwise model. Following [4], 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 [3,13,6] 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 [4] and differentiable NAS approaches [25,7,37] 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 Eq. (5), we also consider $\mathcal{E}(\boldsymbol{x})$ to be pairwise

$$\mathcal{E}(\boldsymbol{x}) = \exp\Big(\sum_{i\in V} \phi_i(x_i) + \sum_{(i,j)\in E} \phi_{i,j}(x_i, x_j)\Big). \tag{9}$$

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 neural networks.

4.1 Diverse M-best Loopy Inference

The main limitation of AOWS [4] is the adoption of Viterbi inference, which is only applicable to simple chain graphs such as MobileNetV1 [16]. When the computational graph forms loops, i.e. it includes skip connections [14], 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 section 6.1.

Furthermore, 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 [36,42]. Therefore, instead of a single MAP assignment $\boldsymbol{x}^*$, we use diverse M-best inference [2] to find a set of diverse solutions $\{\boldsymbol{x}^1, ..., \boldsymbol{x}^m\}$ so as to reduce the variance in the search phase.

Diverse M-best inference requires a set of balanced dissimilarity constraints, each with an associated Lagrange multiplier. In Eq. (2), it is crucial to choose a dissimilarity target k^q, and rather than searching via the diversity constraints, we can directly perform model selection on the Lagrange multiplier λ^q [2]. 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

$$\lambda_i^q = \frac{\max_j \phi_i^q(x_i^j) - \min_j \phi_i^q(x_i^j)}{L}, \tag{10}$$

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 [1], the goal of differentiable NAS is to maximize the following objective

$$-\mathbb{E}_{\boldsymbol{x}\sim\mathbb{P}_{\Phi}(\boldsymbol{x})}[l(\boldsymbol{w}^*|\boldsymbol{x})] \quad \text{s.t. } \boldsymbol{w}^* = \operatorname*{argmin}_{\boldsymbol{w}} l(\boldsymbol{w}|\boldsymbol{x}), \tag{11}$$

where $l(\cdot)$ is some loss function and $\boldsymbol{w}$ encodes connection weights of the neural network. In order to make Eq. (11) differentiable, we can approximate it through Monte Carlo with n_{mc} samples and use the Gumbel-Softmax reparameterization trick [19] 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 [1], sampling from $\mathbb{P}_{\Phi}(\boldsymbol{x})$ is the same as independently sampling from the marginal probability distribution $\mathbb{P}_{\phi_i}(x_i)$:

$$\mathbb{P}_{\Phi}(\boldsymbol{x}) = \frac{1}{Z}\exp\Big(\sum_i \phi_i(x_i)\Big) = \prod_i \frac{1}{Z}\exp\Big(\phi_i(x_i)\Big) = \prod_i \mathbb{P}_{\phi_i}(x_i). \tag{12}$$

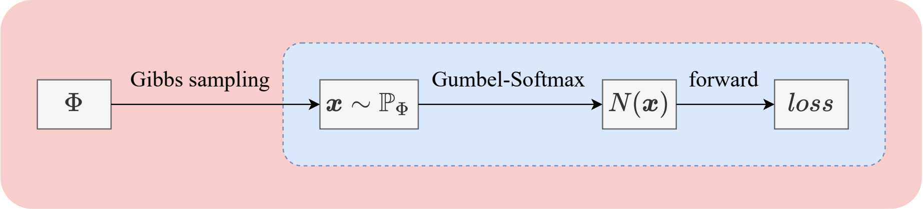

Fig. 1: Workflow of our differentiable parameter learning approach. Vanilla differentiable NAS methods [1] with Monte Carlo approximation and Gumbel-Softmax trick are shown in the blue rectangle. In our case, since the joint probability distribution $\mathbb{P}_{\Phi}(\boldsymbol{x})$ cannot be decomposed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

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:

$$x_i^t \sim \mathbb{P}_{\Phi}(x_i|\boldsymbol{x}_{-i}) \tag{13}$$

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 Fig. 1.

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). 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

We choose five semantic segmentation datasets with diverse contents. Specifically, DeepGlobe challenge [9] provides three aerial image datasets: Land, Road and Building. Land is a multi-class (urban, agriculture, rangeland, forest, water and barren) segmentation dataset and it contains 803 satellite images focusing on rural areas. Road is a binary segmentation dataset and it consists of 6226 images captured over Thailand, Indonesia, and India. Building is also a binary segmentation task with 10593 images taken from Las Vegas, Paris, Shanghai, and Khartoum.

CHAOS [20] is medical image dataset including both computed tomography (CT) and magnetic resonance imaging (MRI) scans of abdomen organs (liver, right kidney, left kidney and spleen). We only use MRI scans, and it has 20 cases and 1270 2D-slices. PROMISE [24] contains prostate MRI images, and it has 50 cases and 1377 2D-slices. For all datasets, only training sets are available. We split 60%/20%/20% of the training set as train/val/test set. For simplicity, we resize all images to 256 × 256.

5.2 Search Space

We use UNet [27] 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 1. For the original UNet that has 26 layers, our search space has about 4×10^{23} configurations in total.

For a fair comparison, for manually designed architectures, e.g. UNet [27], UNet++ [43] and BiO-Net [33], 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 [32], MS-NAS [35], and BiX-Net [31], we do not make any modification and use their implementations directly. However, there exist discrepancies. For instance, we use transposed convolution for up-sampling while NAS-UNet [32] uses dilated transposed convolution and BiX-Net [31] uses bilinear interpolation.

Table 1: Search space with UNet as the backbone.

Type	Size	Width
Normal	3, 5	0.5, 0.75, 1.0, 1.25, 1.5
Down	3	0.5, 0.75, 1.0, 1.25, 1.5
Up	2	0.5, 0.75, 1.0, 1.25, 1.5

5.3 Implementation Details

In the search phase, we train a super-network using the sandwich rule [39] for $T = 50$ epochs. Initially, factors are not updated until the super-network is trained for a warmup period of 10 epochs. The learning rate of network 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 with weight decay of 0.0001. The learning rate of MRF factors is fixed at 0.0003 and we also use the Adam optimizer with the same weight decay. For the sampling, 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 inference, we choose $m = 5$ and $L = 10$ for diverse M-best inference, and the number of binary search iterations is $n_{\text{iter}} = 20$. For simplicity, we search on Deepglobe Land [9] and the found architectures are evaluated on other 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.

5.4 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 will discuss the cost of loopy inference algorithms in section 6.1, and since we use approximate inference algorithm 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.

5.5 MRF-UNets Architecture

MRF-UNet shows several interesting 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 [45,26,5]. 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, we propose MRF-UNetV2 to emphasize these characteristics. As shown in Fig. 2, MRF-UNetV2 has a simpler architecture that is easier for implementation, and we show that it can sometimes outperform MRF-UNet in Table 2 and Table 3. We note that a recent trend in NAS is to design a more and more complex search space to include as many candidates as possible [30], but it becomes difficult to interpret the search results. We hope that our observations can inspire practitioners when designing other encoder-decoder architectures.

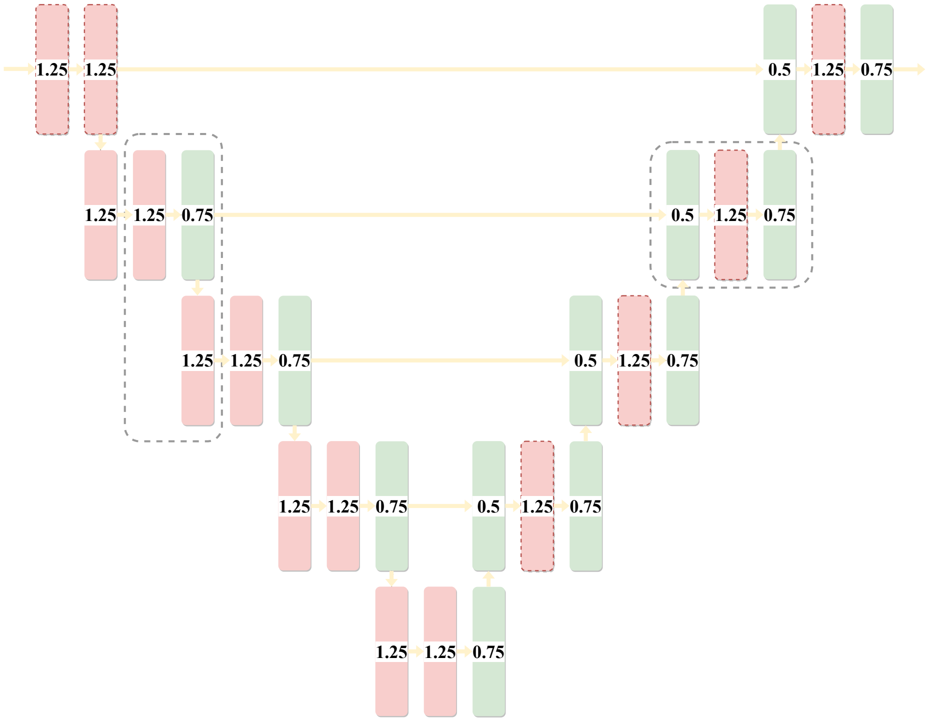

Fig. 2: Architecture of MRF-UNetV2. Numbers inside rectangles are width ratios to the original UNet. Rectangles surrounded by colored dashed lines use 5×5 kernels while others use 3×3 kernels. The gray dashed line on the left highlights an example of a bottleneck block in the encoder, and the gray dashed line on the right shows an example of an inverted bottleneck block in the decoder.

5.6 Main Results

Our benchmarks include manually designed architectures: UNet [27], UNet++ [43], BiO-Net [33] and architectures found with NAS: NAS-UNet [32], MS-NAS [35], BiX-Net [31]. The main results are in Table 2 and Table 3. We set the latency target to 1.70ms which is the same as the original UNet, and we also include FLOPs for comparison. Our MRF-UNets outperform benchmarks over five datasets with diverse semantics, while require less computational resources.

Table 2: mIoU (%) on DeepGlobe challenge. mean±std are computed over 5 runs.

Model	Land (%)	Road (%)	Building (%)	FLOPs (G)	Latency (ms)
UNet	58.41 ± 0.52	57.05 ± 0.13	75.12 ± 0.09	4.84	1.70
UNet++	59.53 ± 0.21	56.96 ± 0.08	74.27 ± 0.21	11.76	5.11
BiO-Net	58.10 ± 0.37	57.06 ± 0.22	74.29 ± 0.15	37.22	3.28
NAS-UNet	58.11 ± 0.91	57.73 ± 0.56	74.46 ± 0.19	30.44	4.25
MS-NAS	58.75 ± 0.68	57.34 ± 0.35	74.61 ± 0.17	24.28	3.96
BiX-Net	57.96 ± 0.94	57.74 ± 0.08	74.63 ± 0.12	13.28	1.87
MRF-UNet	**59.64**±0.44	57.81 ± 0.17	75.50 ± 0.09	4.74	1.68
MRF-UNetV2	58.56 ± 0.25	**57.90**±0.23	**75.84**±0.13	4.66	1.70

Table 3: Dice scores (%) on CHAOS and PROMISE. mean±std are computed over 5 runs.

Model	CHAOS (%)	PROMISE (%)	FLOPs (G)	Latency (ms)
UNet	91.16 ± 0.23	84.60 ± 0.68	4.84	1.70
UNet++	91.46 ± 0.15	86.29 ± 0.35	11.76	5.11
BiO-Net	91.80 ± 0.42	86.04 ± 0.77	37.22	3.28
NAS-UNet	91.30 ± 0.65	85.04 ± 0.90	30.44	4.25
MS-NAS	91.47 ± 0.35	85.42 ± 0.72	24.28	3.96
BiX-Net	91.22 ± 0.39	84.35 ± 0.91	13.28	1.87
MRF-UNet	92.03 ± 0.31	**86.76**± 0.32	4.74	1.68
MRF-UNetV2	**92.14**±0.24	86.61 ± 0.36	4.66	1.70

6 Ablation Study

6.1 Exact vs. Approximate Loopy Inference

Without our loopy inference extension, AOWS fails on more complex loopy graphs. However, inference on a loopy MRF is a NP-hard problem [21], 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 [21]. We compare architectures found by MPCT and MPLP in Table 4. They usually obtain very similar solutions and their results are almost identical. The complexity of MPCT and in general of exact inference is $O(|X|^{|C|})$ where $|X|$ is the cardinality of factors and in our experiments it is 10, 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 [17]. In Fig. 3, we show the size of the largest clique vs. the number nodes for UNet [27], UNet+ [43] and UNet++ [43]. UNet++, being more densely connected, has a much larger clique size than UNet. The clique size of the original UNet is 5, and MPCT already takes several minutes on our MacBook Pro (14-inch, 2021). 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 deeper UNet or

UNet+/UNet++. Nevertheless, MPLP can converge within a few seconds. Since they usually find similar solutions, we use MPLP as a default.

Table 4: Evaluating architectures found by MPCT and MPLP on DeepGlobe Land. mean±std are computed over 5 runs.

Algorithm	MPCT	MPLP
mIoU (%)	59.56 ± 0.57	**59.64**±0.44

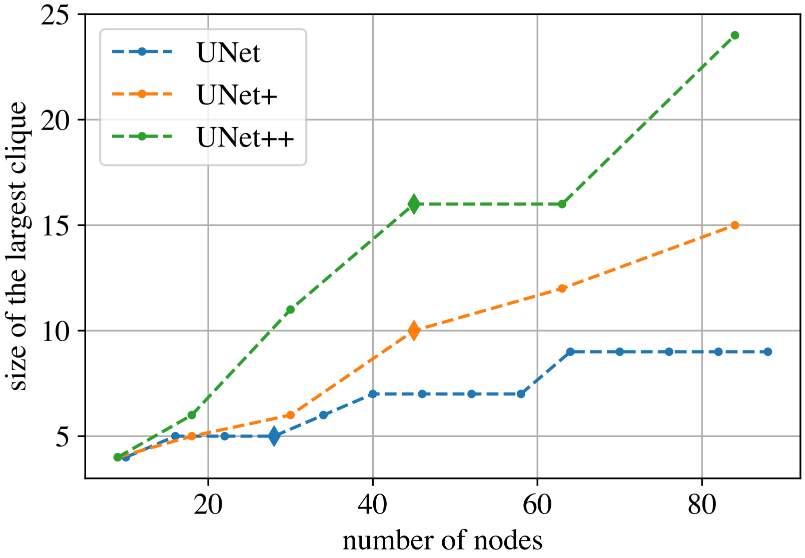

Fig. 3: Size of the largest clique vs. the number of nodes for UNet [27], UNet+ [43] and UNet++ [43]. Diamonds indicate the original architectures whose depth is 5.

6.2 Diverse Solutions

As shown in [36,42], 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 [2]. In Table 5, we show the results of diverse 5-best. The MAP solution is 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 [12] is critical for medical diagnosis and deep ensembles [22] 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 7, 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 Table 6, 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 Eq. (10). Indeed, the best architecture is now the 5th 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.

Table 5: Evaluating architectures found by diverse 5-best on DeepGlobe Land ($L = 10$). mean±std are computed over 5 runs.

Solution	1	2	3	4	5
mIoU (%)	57.37±0.61	57.41±0.32	**59.64**±0.44	59.43±0.46	56.59±0.70

Table 6: Evaluating architectures found by diverse 10-best on DeepGlobe Land ($L = 20$). mean±std are computed over 5 runs.

Solution	1	2	3	4	5
mIoU (%)	57.02±0.53	58.37±0.41	56.94±0.55	59.37±0.41	**59.56**±0.22
Solution	6	7	8	9	10
mIoU (%)	57.86±0.41	59.45±0.92	57.82±0.51	57.79±0.48	57.84±0.39

Table 7: Comparing deep ensemble [22] with our diverse deep ensemble on CHAOS. Lower is better. mean±std are computed over 5 runs.

Model	Deep Ensemble	Diverse Deep Ensemble
ECE (%)	0.7091 ± 0.0140	**0.6872**±0.0126

6.3 Pairwise Formulation and Differentiable Parameter Learning

Except diverse M-best loopy inference, we make other two modifications: pairwise formulation and differentiable parameter learning. In table 8, we compare the architectures found by our our MRF-NAS vs. MRF-NAS without pairwise factors and MRF-NAS without differentiable parameter learning.

Table 8: Evaluating architectures found by A1: MRF-NAS w/o pairwise factors, A2: MRF-NAS w/o differentiable parameter learning and MRF-UNet on DeepGlobe Land. mean±std are computed over 5 runs.

Architecture	A1	A2	MRF-UNet
mIoU (%)	59.17±0.58	59.33±0.31	**59.64**±0.44

7 Conclusion

In this paper, we propose MRF-NAS that extends and improves AOWS [4] with a more general framework based on a pairwise Markov Random Field (MRF) formulation, which leads to applying various statistical techniques for MAP optimization. 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 and propose MRF-UNetV2 with a simpler architecture that can further improve our results. MRF-UNets, albeit requiring less computational resources, outperform several SOTA benchmarks over three aerial image datasets and two medical image datasets that contain diverse contents. This demonstrates that the found architectures are robust and effective.

Acknowledgements

We acknowledge support from the Research Foundation - Flanders (FWO) through project numbers G0A1319N and S001421N, and funding from the Flemish Government under the Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen programme.

References

1. Ardywibowo, R., Boluki, S., Gong, X., Wang, Z., Qian, X.: NADS: Neural Architecture Distribution Search for Uncertainty Awareness. ICML (2020)
2. Batra, D., Yadollahpour, P., Guzman-Rivera, A., Shakhnarovich, G.: Diverse M-best solutions in Markov random fields. ECCV (2012)
3. Bender, G., Kindermans, P.J., Zoph, B., Vasudevan, V., Le, Q.: Understanding and simplifying one-shot architecture search. ICML (2018)

4. Berman, M., Pishchulin, L., Xu, N., Blaschko, M., Medioni, G.: AOWS: Adaptive and Optimal Network Width Search with Latency Constraints. CVPR (2020)
5. Chaurasia, A., Culurciello, E.: LinkNet: Exploiting Encoder Representations for Efficient Semantic Segmentation. VCIP (2017)
6. Chu, X., Zhang, B., Xu, R.: FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. ICCV (2021)
7. Chu, X., Zhou, T., Zhang, B., Li, J.: Fair DARTS: Eliminating Unfair Advantages in Differentiable Architecture Search. ECCV (2020)
8. Dai, X., Wan, A., Zhang, P., Wu, B., He, Z., Wei, Z., Chen, K., Tian, Y., Yu, M., Vajda, P., Gonzalez, J.E.: FBNetV3: Joint Architecture-Recipe Search using Predictor Pretraining. CVPR (2021)
9. Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., Hughes, F., Tuia, D., Raskar, R.: DeepGlobe 2018: A Challenge to Parse the Earth through Satellite Images. CVPR Workshop (2018)
10. Ding, M., Huo, Y., Lu, H., Yang, L., Wang, Z., Lu, Z., Wang, J., Luo, P.: Learning Versatile Neural Architectures by Propagating Network Codes. ICLR (2022)
11. Eelbode, T., Bertels, J., Berman, M., Vandermeulen, D., Maes, F., Bisschops, R., Blaschko, M.B.: Optimization for medical image segmentation: Theory and Practice When Evaluating With Dice Score or Jaccard Index. TMI (2020)
12. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. ICML (2017)
13. Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J.: Single path one-shot neural architecture search with uniform sampling. ECCV (2020)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CVPR (2016)
15. He, Y., Yang, D., Roth, H., Zhao, C., Xu, D.: DiNTS: Differentiable Neural Network Topology Search for 3D Medical Image Segmentation. CVPR (2021)
16. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv (2017)
17. Huang, G., Liu, Z., Maaten, L.v.d., Weinberger, K.Q.: Densely connected convolutional networks. CVPR (2017)
18. Huang, Z., Wang, Z., Yang, Z., Gu, L.: AdwU-Net: Adaptive Depth and Width U-Net for Medical Image Segmentation by Differentiable Neural Architecture Search. MIDL (2022)
19. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with Gumbel-softmax. ICLR (2017)
20. Kavur, A.E., Gezer, N.S., Barış, M., Aslan, S., Conze, P.H., Groza, V., Pham, D.D., Chatterjee, S., Ernst, P., Özkan, S., Baydar, B., Lachinov, D., Han, S., Pauli, J., Isensee, F., Perkonigg, M., Sathish, R., Rajan, R., Sheet, D., Dovletov, G., Speck, O., Nürnberger, A., Maier-Hein, K.H., Akar, G.B., Ünal, G., Dicle, O., Selver, M.A.: CHAOS Challenge - combined (CT-MR) healthy abdominal organ segmentation. MIA (2021)
21. Koller, D., Friedman, N.: Probabilistic graphical models: Principles and Techniques. MIT press (2009)
22. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. NeurIPS (2017)
23. Liang, T., Wang, Y., Tang, Z., Hu, G., Ling, H.: OPANAS: One-Shot Path Aggregation Network Architecture Search for Object Detection. CVPR (2021)

24. Litjens, G., Toth, R., Ven, W.v.d., Hoeks, C., Kerkstra, S., Ginneken, B.v., Vincent, G., Guillard, G., Birbeck, N., Zhang, J., Strand, R., Malmberg, F., Ou, Y., Davatzikos, C., Kirschner, M., Jung, F., Yuan, J., Qiu, W., Gao, Q., Edwards, P. Maan, B., Heijden, F.v.d., Ghose, S., Mitra, J., Dowling, J., Barratt, D., Huisman, H., Madabhushi, A.: Evaluation of prostate segmentation algorithms for MRI: The PROMISE12 Challenge. MIA (2014)
25. Liu,H.,Simonyan,K.,Yang,Y.: DARTS: Differentiable Architecture Search. ICLR (2019)
26. Milletari, F., Navab, N., Ahmadi, S.A.: V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. 3DV (2016)
27. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI (2015)
28. Srivastava, S., Berman, M., Blaschko, M.B., Tuia, D.: Adaptive compression-based lifelong learning. BMVC (2019)
29. Veit, A., Wilber, M., Belongie, S.: Residual Networks Behave Like Ensembles of Relatively Shallow Networks. NeurIPS (2016)
30. Wan, X., Ru, B., Esperança, P.M., Li, Z.: On Redundancy and Diversity in Cell-based Neural Architecture Search. ICLR (2022)
31. Wang, X., Xiang, T., Zhang, C., Song, Y., Liu, D., Huang, H., Cai, W.: BiX-NAS: Searching Efficient Bi-directional Architecture for Medical Image Segmentation. MICCAI (2021)
32. Weng, Y., Zhou, T., Li, Y., Qiu, X.: NAS-Unet: Neural Architecture Search for Medical Image Segmentation. IEEE Access (2019)
33. Xiang, T., Zhang, C., Liu, D., Song, Y., Huang, H., Cai, W.: BiO-Net: Learning Recurrent Bi-directional Connections for Encoder-Decoder Architecture. MICCAI (2020)
34. Xu, H., Yao, L., Zhang, W., Liang, X., Li, Z.: Auto-FPN: Automatic Network Architecture Adaptation for Object Detection Beyond Classification. ICCV (2019)
35. Yan, X., Jiang, W., Shi, Y., Zhuo, C.: MS-NAS: Multi-Scale Neural Architecture Search for Medical Image Segmentation. MICCAI (2020)
36. Yang, A., Esperança, P.M., Carlucci, F.M.: NAS evaluation is frustratingly hard. ICLR (2020)
37. Ye, P., Li, B., Li, Y., Chen, T., Fan, J., Ouyang, W.: β-DARTS: Beta-Decay Regularization for Differentiable Architecture Search. CVPR (2022)
38. Yu, J., Huang, T.: AutoSlim: Towards One-Shot Architecture Search for Channel Numbers. NeurIPS Workshop (2019)
39. Yu, J., Huang, T.S.: Universally slimmable networks and improved training techniques. ICCV (2019)
40. Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P.J., Tan, M., Huang, T., Song, X., Pang, R., Le, Q.: BigNAS: Scaling Up Neural Architecture Search with Big Single-Stage Models. ECCV (2020)
41. Yu, J., Yang, L., Xu, N., Yang, J., Huang, T.: Slimmable neural networks. ICLR (2019)
42. Yu, K., Sciuto, C., Jaggi, M., Musat, C., Salzmann, M.: Evaluating the search phase of neural architecture search. ICLR (2020)
43. Zhou, Z., Siddiquee, M.M.R., Tajbakhsh, N., Liang, J.: UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. TMI (2020)
44. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. ICLR (2017)
45. Çiçek, , Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. MICCAI (2016)