International Journal of Computer Vision
https://doi.org/10.1007/s11263-023-01920-9

# FlowNAS: Neural Architecture Search for Optical Flow Estimation

**Zhiwei Lin[1] · Tingting Liang[1] · Taihong Xiao[2] · Yongtao Wang[1] · Ming-Hsuan Yang[2]**

Received: 3 November 2022 / Accepted: 19 September 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

**Abstract**
Recent optical flow estimators usually employ deep models designed for image classification as the encoders for feature extraction and matching. However, those encoders developed for image classification may be sub-optimal for flow estimation. In contrast, the decoder design of optical flow estimators often requires meticulous design for flow estimation. The disconnect between the encoder and decoder could negatively affect optical flow estimation. To address this issue, we propose a neural architecture search method, FlowNAS, to automatically find the more suitable and stronger encoder architecture for existing flow decoders. We first design a suitable search space, including various convolutional operators, and construct a weight-sharing super-network for efficiently evaluating the candidate architectures. To better train the super-network, we present a Feature Alignment Distillation module that utilizes a well-trained flow estimator to guide the training of the super-network. Finally, a resource-constrained evolutionary algorithm is exploited to determine an optimal architecture (i.e., sub-network). Experimental results show that FlowNAS can be easily incorporated into existing flow estimators and achieves state-of-the-art performance with the trade-off between accuracy and efficiency. Furthermore, the encoder architecture discovered by FlowNAS with the weights inherited from the super-network achieves 4.67% F1-all error on KITTI, an 8.4% reduction of RAFT baseline, surpassing state-of-the-art handcrafted GMA and AGFlow models, while reducing the model complexity and latency. The source code and trained models will be released at https://github.com/VDIGPKU/FlowNAS.

**Keywords** Optical flow estimation · Neural architecture search · Knowledge distillation

## 1 Introduction

Optical flow estimation aims to measure per-pixel 2D motion between consecutive video frames, which is widely used in vision tasks, e.g., action recognition (Sun et al., 2018c), object tracking (Tan et al., 2022; Biswas et al., 2022), and video understanding (Fortun et al., 2015). One key component for accurate optical flow estimation is constructing a discriminative cost volume for matching features.

Recently, deep neural networks have been applied to optical flow estimation by extracting more effective features (Dosovitskiy et al., 2015; Sun et al., 2018b; Teed and Deng, 2020). The flow estimation can be further refined with the pyramid coarse-to-fine decoder architecture design. While most flow estimation methods focus on constructing better cost volumes (Yang and Ramanan, 2019; Xiao et al., 2020; Jiang and Learned-Miller, 2021) or designing a more delicate decoder (Jiang et al., 2021a), we show that the encoder architectures are also essential for two reasons. First, the cost volume and flow decoder rely on the feature representation extracted by the encoder. A stronger encoder

Communicated by Jianfei Cai.

Zhiwei Lin, Tingting Liang and Taihong Xiao have equally contributed to this work.

✉ Yongtao Wang
wyt@pku.edu.cn

Zhiwei Lin
zwlin@pku.edu.cn

Tingting Liang
tingtingliang@pku.edu.cn

Taihong Xiao
txiao3@ucmerced.edu

Ming-Hsuan Yang
mhyang@ucmerced.edu

[1] Wangxuan Institute of Computer Technology, Peking University, Beijing, China

[2] University of California, Merced, CA, USA

Published online: 25 October 2023

can provide feature representations with more semantic and motion information for cost volume construction or flow estimation. Second, the encoder constitutes a considerable parameter component of a flow network model. For example, the model parameters of the encoder in RAFT (Teed and Deng, 2020) amount to 58% of the entire network. However, recent state-of-the-art flow estimation methods still adopt encoders designed for image classification to extract features of input images. Due to the natural differences between image classification and flow estimation tasks, the encoder directly adopted from that designed for classification tasks is sub-optimal, likely negatively affecting flow estimation. Thus, finding an encoder architecture suitable for flow estimation is very important.

In a separate line of research, to reduce human efforts in designing neural networks, Neural Architecture Search (NAS) has been successfully applied to various high-level vision tasks (Bender et al., 2018; Tan et la., 2020; Liu et al., 2019a; Saikia et al., 2019). Nevertheless, much less attention has been paid to using NAS for low-level vision tasks. This can be attributed to several factors. In general, NAS requires selecting network components (e.g., kernel size of convolution in a particular layer) through many possible architectures. This entails heavy computational loads (e.g., early NAS algorithms (Zoph and Le, 2017) require thousands of GPU hours to find an architecture on the CIFAR dataset (Krizhevsky et al., 2009)). In addition to the heavy computational requirements, existing NAS methods are designed for specific tasks with different priors. To our knowledge, NAS has not been employed to determine optimal architecture for flow estimation. Since the decoder architecture design involves much prior knowledge of optical flow estimation, changes in the decoder architectures may cause negative effects (See Sect. 4.6). Furthermore, the modules designed in existing flow decoders are highly coupled with their flow estimation pipelines. It is challenging to separate independent decoder operation modules and construct a search space for flow decoder search. Nevertheless, these two problems exist outside the encoder design of flow estimators.

In this paper, we present a neural network architecture search method, FlowNAS, specifically for better encoder designs within the context of optical flow estimation. We leverage human knowledge in flow estimation as priors in architecture search and design. To handle the issue of computational overheads, we first determine a suitable search space by exploring the effective convolutional operators and then construct a super-network that comprises all weight-sharing sub-network modules. We then propose the baseline FlowNAS following (Gou et al., 2020) within two steps: (1) constraint-free super-network pre-training, (2) resource-constrained sub-network search. However, due to the large number of sub-networks sharing weights and thus causing interference, the weights inherited directly from super-network are often sub-optimal. Hence, retraining the discovered architecture from scratch is usually required, introducing additional computational overheads. Furthermore, due to the sub-optimal weights of the super-network, there exists a low correlation problem between the performance of weight-inherited and retrained sub-networks, i.e., the best-performing sub-network determined by the resource-constrained search is not necessarily the best architecture for the overall performance of flow estimation.

To address all the issues mentioned above, in contrast to existing NAS algorithms that use sophisticated pruning or sampling strategies (Cai et al., 2020; Yu et al., 2020; Wang et al., 2021) to mitigate interference between sub-networks, we propose a Feature Alignment Distillation (FAD) method to fully utilize human priors in flow estimation and achieve much better performance. We take pre-trained weights of handcrafted optical flow estimator, e.g., RAFT (Teed and Deng, 2020), as our teacher model and super-network as our student model. In each training iteration, we sample one sub-network from the super-network, and the extracted feature map pyramid is trained under the guidance of the teacher model. Specifically, Channel-wise Alignment is applied to the feature map of both teacher and student to make the distillation process independent of the number of channels. It is worth mentioning that using existing open-source weights as teacher guidance will not limit the performance of FlowNAS. Instead, it leads to fast convergence and performance improvement of the super-network. By using FAD, for any discovered architecture, inheriting weights directly from the super-network can achieve similar or even better performance than retraining from scratch, reducing the overall training time by twice. Meanwhile, we avoid the low correlation problem since we do not have to retrain the discovered sub-networks from scratch to get their actual performance. Although we introduce an additional teacher model into FlowNAS, it brings marginal computational overheads during training ($\sim 0.2\times$ training cost), since we only need to calculate the features of the encoder for the teacher model. The teacher model is discarded during inference, and no additional computational cost is introduced.

With the evolutionary algorithm, we simultaneously obtain outstanding model architectures and their corresponding weights. In practice, for any existing decoder of optical flow estimation, FlowNAS can find a more lightweight and powerful encoder that can directly replace the original one. Experimental results demonstrate that our searched encoder outperforms handcrafted encoders with several different kinds of decoders (Fig. 1). The main contributions of this work are:

- We analyze the importance of the encoders of flow estimators and propose a neural architecture search

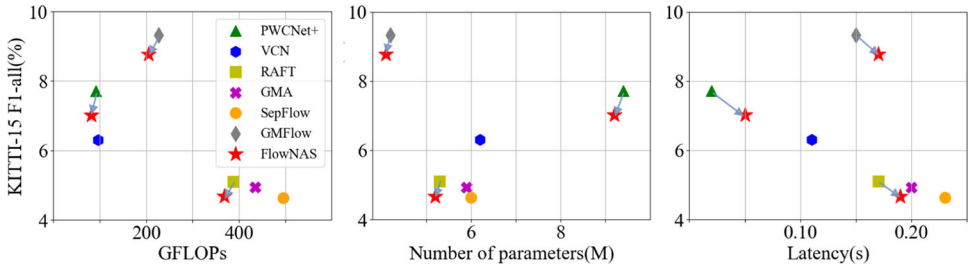


**Fig. 1 Model parameters, GFLOPs, latency *vs.* KITTI F1-all error.** FlowNAS achieves better accuracy-efficiency trade-offs than handcrafted approaches

framework, FlowNAS, to automatically design a stronger encoder for optical flow estimation.
To our knowledge, this is the first work introducing NAS to this challenging task.

- We present an efficient search space for FlowNAS and propose a Feature Alignment Distillation module to enhance the representation capability of the super-network by effectively leveraging prior information.
- We show that FlowNAS can be easily incorporated with existing flow estimators for better performance.
For example, FlowNAS-RAFT achieves an F1-all error of 4.67% on KITTI, an 8.4% error rate reduction of RAFT (Teed and Deng, 2020), surpassing existing models including GMA (Jiang et al., 2021a) and AGFlow (Luo et al., 2022) while reducing model complexity and latency.

## 2 Related Work

### 2.1 Deep Models for Optical Flow

Recent optical flow estimation methods have been inspired by the success of deep models for vision tasks. FlowNet (Dosovitskiy et al., 2015) uses convolutional neural networks to extract and correlate features extracted from two frames for flow estimation. As it requires a large amount of annotated samples for training, a synthetic dataset is created to fascinate the training process. In (Ilg et al., 2017), FlowNet2.0 stacks multiple basic FlowNet modules for iterative refinement and achieves better results. Exploiting a coarse-to-fine refinement paradigm, SpyNet (Ranjan and Black, 2017) uses a spatial pyramid network that warps images at different scales to deal with large motions. On the other hand, PWC-Net (Sun et al., 2018b) extracts the feature through pyramid processing and builds a cost volume at each level, where the estimated flow is iteratively refined.

Aside from the pyramid architecture design, numerous methods have developed effective cost volume modules for feature matching to estimate optical flow. VCN (Yang and Ramanan, 2019) presents a cost volume scheme by decoupling the 4D convolution into a 2D spatial filter and a 2D winner-take-all filter. In (Xiao et al., 2020), LCV shows the performance of flow estimation methods can be improved by learning an effective cost volume for feature matching based on Cayley representations. In (de Jong et al., 5555), a grid search is performed to determine the highest response in the spatiotemporal frequency domain for analyzing how deep neural networks estimate optical flow. DCVNet (Jiang and Learned-Miller, 2021) proposes dilated cost volumes to capture small and large displacements.

Another line of work on flow refinement focuses on designing effective decoder modules. In (Hur and Roth, 2019), IRR introduces an iterative residual refinement scheme for joint estimation of occlusion and flow. RAFT (Teed and Deng, 2020) proposes a lightweight recurrent decoder by sharing weights across the iterative refinement process, which facilitates feature matching of all pairs of pixels for accurate flow estimation. Recently, GMA (Jiang et al., 2021a) has introduced a global motion aggregation to capture long-range self-similarity in the reference frames.

In contrast to the methods mentioned above, where the model architectures are hand-crafted based on some principles, we propose to learn better feature representations by searching for a better network architecture specifically for optical flow estimation.

### 2.2 Neural Architecture Search

**NAS for High-level Vision Tasks:** Neural Architecture Search (NAS) aims to replace the efforts of human experts in the architecture design of deep neural networks with machines. It has achieved significant success in numerous high-level vision tasks such as classification (Bender et al., 2018; Brock et al., 2018; Chu et al., 2021; Liu et al., 2019b; Chu et al., 2020; Cai et al., 2019), object detection (Tan et la., 2020; Liang et al., 2021), and semantic segmentation (Liu et al., 2019a; Zhang et al., 2019). Early NAS methods (Liu et al., 2018; Liu et al., 2018b; Real et al., 2019) explore thousands of candidate architectures from scratch (on a smaller proxy task) and select the most promising regions in the search space based on a predefined metric. Recent approaches (Bender et al., 2018; Brock et al., 2018; Gou et al., 2020; Chu et al., 2021) amortize the cost by training a single super-network. The sub-networks of these methods can be efficiently ranked by using shared weights to estimate their

relative accuracy. To speed up the search process, gradient-based approaches (Liu et al., 2019b; Chu et al., 2020) are proposed for continuous relaxation of the search space, which enables differentiable optimization in architecture search. One main issue with super-network-based methods is the low-rank correlation between sub-networks and the super-network. To improve the rank correlation, FairNAS (Chu et al., 2021) uses a sampling strategy to train sub-networks. ReNAS (Xu et al., 2021) proposes to train an architecture performance predictor to give the final performance of an architecture. However, FairNAS needs to retrain the sub-networks from scratch after obtaining the best sub-network architecture. ReNAS (Xu et al., 2021) trains a performance predictor for architecture search. Nevertheless, it requires a large number of ground truth architecture-performance pairs (423 pairs in (Xu et al., 2021)) to train the predictor, which is a substantial computational overhead. CTNAS (Chen et al., 2021) devises a scheme to evaluate architectures via pairwise comparisons. Although reducing the requirement of ground truth architecture-performance pairs, it still needs 100 pairs to achieve a favorable result.

All the approaches mentioned above require retraining: first, training a super-network to determine the optimal architecture (sub-network) configuration, and then retraining this architecture from scratch to obtain the final result, introducing additional training costs. To alleviate these issues, several pruning and training schemes (Cai et al., 2020; Yu et al., 2020; Wang et al., 2021) have been developed to improve the performance of the super-network, where each sub-network performs on par with its stand-alone performance. OFA (Cai et al., 2020) pre-trains a single whole network and then progressively distills it to obtain a smaller network, and BigNAS (Yu et al., 2020) utilizes sandwich rule as well as distillation to handle a wider set of models. Recently, AttentiveNAS (Wang et al., 2021) uses a Pareto front sampling strategy to optimize the super-network better.

**NAS for Low-level Vision Tasks:** Advances in NAS have led to numerous applications in low-level vision tasks (Zhang et al., 2020; Li et al., 2020b; Cheng et al., 2020; Liu et al., 2021). Currently, gradient-based differentiable architecture search (DARTS)-type methods (Liu et al., 2019b) are often used. LEAStereo (Cheng et al., 2020) presents an end-to-end NAS framework for deep stereo matching by incorporating task-specific human knowledge into the architecture framework. It designs a task-specific architecture search space and uses differentiable optimization for architecture search. HiNAS (Zhang et al., 2020) uses primitive search space (e.g., $3 \times 3$ separable convolution) to address synthetic Gaussian noise removal for image restoration. In (Liu et al., 2021), RUAS designs an unrolling-type architecture search to handle low-light image enhancement. On the other hand, AutoDispNet (Saikia et al., 2019) constructs a cell-based search space for disparity estimation and introduces a hyper-parameter optimization method to train the searched architecture.

We note that existing NAS methods for low-level vision mainly require two-stage training: super-network training and sub-network retraining, introducing an additional $1\times$ computational overhead. In our work, we first employ NAS to determine model architectures for flow estimation and exploit the potential of NAS simultaneously. Specifically, we propose FAD to train a stronger super-network. In this training approach, we can remove the retraining stage. The weights inherited from the super-network can achieve similar or even better performance than retraining from scratch for any searched architecture.

**NAS for Model Compression:** In addition to constructing a better network architecture, NAS can be applied to prune model parameters. NAT (Gou et al., 2022) exploits graph convolutional networks to build the architecture optimization process and removes the redundant operations. DCP/DKP (Liu et al., 2022) introduces additional discrimination-aware losses to guide the channel/kernel selection. On the other hand, NPPM (Gao et al., 2021) prunes channels for CNNs by maximizing the accuracy of sub-networks with the help of a performance prediction network. Similar to ReNAS (Xu et al., 2021) and CTNAS (Chen et al., 2021), NPPM (Gao et al., 2021) also requires ground truth architecture-performance pairs for network training. The aforementioned model compression methods take an existing arbitrary architecture as the input and then output the pruned version of the architecture. They can only remove redundant parameters or structures by reducing kernel sizes or channels. In contrast, FlowNAS discovers an optimal architecture from scratch and must carefully design a suitable search space for optical flow estimation. Thus, we allow sub-networks to choose more diverse structures, including larger filters and expanded channels. FlowNAS and model compression methods are not mutually exclusive. Both can be utilized together to obtain a more lightweight model. For instance, FlowNAS can be employed to search for an optimized architecture. The searched architecture can then be further pruned using existing model compression techniques to reduce the model size.

## 3 NAS for Optical Flow Estimation

In this section, we present the proposed NAS method for optical flow. We construct a compact and efficient search space by leveraging task-specific human knowledge. By utilizing the pre-trained weights of the handcrafted architecture, the trained super-network can achieve a representation model without requiring additional retraining. Figure 2 shows the overall pipeline of FlowNAS.

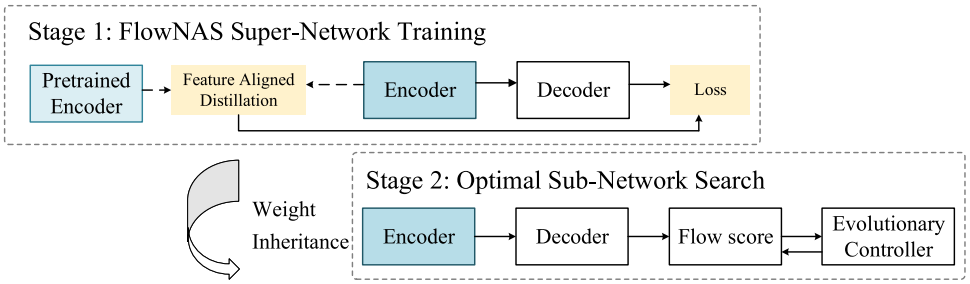


**Fig. 2 Illustration of FlowNAS pipeline.** (1) super-network training. (2) optimal architecture search. The searched architecture and its weights inherited from the super-network can achieve better results than its stand-alone training

**Table 1**
Search Space of FlowNAS

| Block | Width | Depth | Kernel Size | Expansion Ratio | Stride |
|---|---|---|---|---|---|
| First Conv2d | {64} | {1} | {7} | – | 2 |
| SepConv-1 | {56,64} | {1,2} | {3,5} | {1} | 1 |
| SepConv-2 | {64,72} | {1,2,3} | {3,5} | {1,2,4} | 1 |
| SepConv-3 | {88,96} | {1,2,3} | {3,5} | {4,5,6} | 2 |
| SepConv-4 | {96,104,112} | {1,2,3} | {3,5} | {4,5,6} | 1 |
| SepConv-5 | {112,120,128} | {2,3,4} | {3,5} | {6} | 2 |
| SepConv-6 | {128,136} | {1,2} | {3,5} | {6} | 1 |
| Last Conv2d | {128} | {1} | {1} | – | 1 |

Width refers to the number of channels. Depth is the number of repeated blocks. The expansion ratio denotes the channel expansion ratio in the first $1\times1$ convolution of each SepConv Block

### 3.1 Problem Formalization

Assuming the weights of the super-network as $W$ and the architectural configurations as $\mathcal{A} = \{\alpha_i\}$, we formulate the problem as:

$$\begin{aligned} W^* &= \arg\min_{W} \mathbb{E}_{\alpha_i \in \mathcal{A}}[Error_{train}(C(W, \alpha_i))]. \\ \alpha^* &= \arg\min_{\alpha_i \in \mathcal{A}} Error_{val}(C(W^*, \alpha_i)), \\ &\text{s.t. } Param(\alpha_i) < P, \ \forall i. \end{aligned} \tag{1}$$

where $C(W, \alpha_i)$ denotes a selection scheme that selects part from super-network $W$ to form a sub-network with architectural configuration $\alpha_i$, and $P$ is the parameter upper bound. The overall training objective is to optimize $W$ to ensure each supported sub-network maintains the same error rate level as its stand-alone performance. The selection scheme aims to find the sub-network $C(W^*, \alpha^*)$ with the lowest error that satisfies the resource constraint.

### 3.2 Search Space

**Encoder Design:** We note that the deep models for optical flow estimation are usually based on the encoder-decoder structure (Teed and Deng, 2020; Jiang et al., 2021a; Zhang et al., 2021). While most flow estimation methods focus on constructing better cost volumes or designing more effective decoders, we show the importance of encoder design for extracting features while integrating and optimizing different decoders in one super-network is not the focus of this paper. Motivated by the success of RAFT (Teed and Deng, 2020) and CNN models (He et al., 2016; Xie et al., 2017), we propose a CNN model of a sequence of units with gradually reduced feature maps and expanded channels. Each unit consists of a sequence of convolution layers. Similar to recent NAS approaches (Liu et al., 2019b; Gou et al., 2020), we treat each unit as a searching cell, and the entire search space is a regular stack of these searching cells.

**Cell Level Search Space:** We allow each searching cell to use an arbitrary number of convolution blocks (denoted as dynamic depth) and each block to use an arbitrary number of channels (denoted as dynamic width) and kernel sizes (denoted as dynamic kernel size). For example, the depth of each cell is chosen from {1, 2, 3, 4}. While for each convolution operation in the block, the width ranges from 56 to 136 with a stride 8, the kernel size is chosen from {3, 5}, and the expansion ratio is selected from {1, 2, 4, 5, 6}. The summary of the search space is shown in Table 1. With 6 cells, we have roughly $((10\times2\times5)^1+(10\times2\times5)^2+(10\times2\times5)^3+(10\times2\times5)^4)^6 \approx 10^{48}$ different neural network architectures. For each convolution operation in a convolution block, the weights of kernel sizes and channels in different sub-network architectural configurations are shared. In practice, we pre-allocate a weight tensor with the maximum number of kernel sizes and channels. Then, we slice out the corresponding sub-tensor for convolution for a specific architectural configuration choice. As such, all these sub-networks share the same weights from $W$. We only require 8.1M parameters to store all of them.

Without sharing, the total model size will be computationally prohibitive.

We summarize the essential convolution operations for dense image prediction into three types: standard convolution, separable convolution (Chollet, 2017), and shuffle convolution (Zhang et al., 2018). A straightforward approach is to combine them all in the search space, which will be expanded by over 1000 times, further increasing the optimization difficulty of the super-network. Instead, we conduct simple experiments to compare these convolutions and choose the most suitable one, separable convolution, for optical flow estimation. More details can be found in Sect. 4.3.

### 3.3 Baseline FlowNAS

Training a super-network can be a multi-objective problem, where each objective corresponds to one sub-network. A naive training approach optimizes the super-network by enumerating all sub-networks in each update step. However, it is computationally prohibitive for a vast search space, as considered in our model. Another naive training approach is to sample one sub-network at each update step, thereby alleviating the prohibitive cost issues. Similar to (Gou et al., 2020), we propose a baseline FlowNAS model consisting of constraint-free training and resource-constrained search. Since the Sintel and KITTI datasets do not provide official validation subsets, we split the respective training sets into training and validation subsets as carried out by FlowNet (Dosovitskiy et al., 2015) and VCN (Yang and Ramanan, 2019). For constraint-free training, we use the training subset to train the super-network. During the resource-constrained search, we evaluate the performance of the sub-networks with the validation subsets.

**Constraint-free Training:** At each update step, we randomly sample one sub-network architecture configuration $\alpha_i$ from the search space. According to the configuration $\alpha_i$, we slice out the sub-tensors of weights in super-network $W$ to compute the flow predictions. The weights of sub-tensors are then updated by backpropagation.

**Resource-constrained Search:** We conduct the resource-constrained sub-network search with the evolutionary algorithm. Specifically, we randomly sample $N$ sub-network architecture configurations that satisfy the resource constraint to form a population. For each configuration, we slice out the sub-tensors of weights in super-network $W$ and evaluate the performance of this configuration on the validation set. Note that evaluating a sub-network requires only inference without training. We rank the sub-networks by their performance. Then, we repeatedly generate another $N$ new sub-networks satisfying the resource constraint through crossover and mutation on top $k$ performing sub-networks. Similar to SPOS (Gou et al., 2020), crossover means that two randomly selected sub-networks are crossed to produce a new one, and mutation denotes that a randomly selected sub-network mutates its every cell with probability $p$ to construct a new sub-network. Finally, the new sub-networks are added to the population. We repeat the above process $T$ times until convergence. In our experiments, we set $N = 50$, $k = 10$, $p = 0.1$, and $T = 20$.

However, with such a large number of sub-networks sharing weights and thus interfering with each other, weights directly inherited from super-network are often sub-optimal (Gou et al., 2020; Chu et al., 2021). Hence, retraining the discovered architecture from scratch is usually required, introducing additional computational overhead. Furthermore, the rank correlation between the performance of weight-inherited and retrained sub-networks is low due to the sub-optimal weights. This leads to the sub-optimal of discovered architecture by the resource-constrained search. In the following, we introduce the *Feature Alignment Distillation* method to address this issue.

### 3.4 Feature Alignment Distillation

A super-network comprises numerous sub-networks of different sizes. Existing methods typically adopt sophisticated training strategies (Cai et al., 2020; Yu et al., 2020; Wang et al., 2021) to alleviate the interference among sub-networks. However, applying them directly to flow estimation has two limitations: (1) it requires 2 to 3 times more training time to pre-train or sample more than one sub-network in each update step; and (2) such training strategies are designed for image classification, and may not be optimal for dense image prediction such as flow estimation. More analysis can be found in Sect. 4.3.

As our goal is to enhance the representation strength of the super-network and prevent interference among sub-networks, we exploit human knowledge to guide the training process. Specifically, we use pre-trained weights of a handcrafted flow estimator as a teacher to guide our super-network training, making the best use of human priors. To this end, we propose the Feature Alignment Distillation (FAD) method in this work. FAD can guide the feature outputs of all sub-networks sampled from the super-network to converge into the same feature space, which reduces the decoder's training instability and facilitates the whole super-network's training process.

As shown in Fig. 3, FlowNAS takes a handcrafted estimator with the pre-trained weights, e.g., RAFT (Teed and Deng, 2020), as a teacher model, and a super-network as the student model. The feature pyramids extracted by the teacher model and the sub-network are denoted as $\{f_i^T \in \mathcal{R}^{c_i^T \times h \times w} \mid i = 1, 2, ..., n\}$ and $\{f_i^S \in \mathcal{R}^{c_i^S \times h \times w} \mid i = 1, 2, ..., n\}$. Since each sub-network has dynamic width in the search space, the channel number of $f_i^S$ varies and does not align with $f_i^T$. Thus,

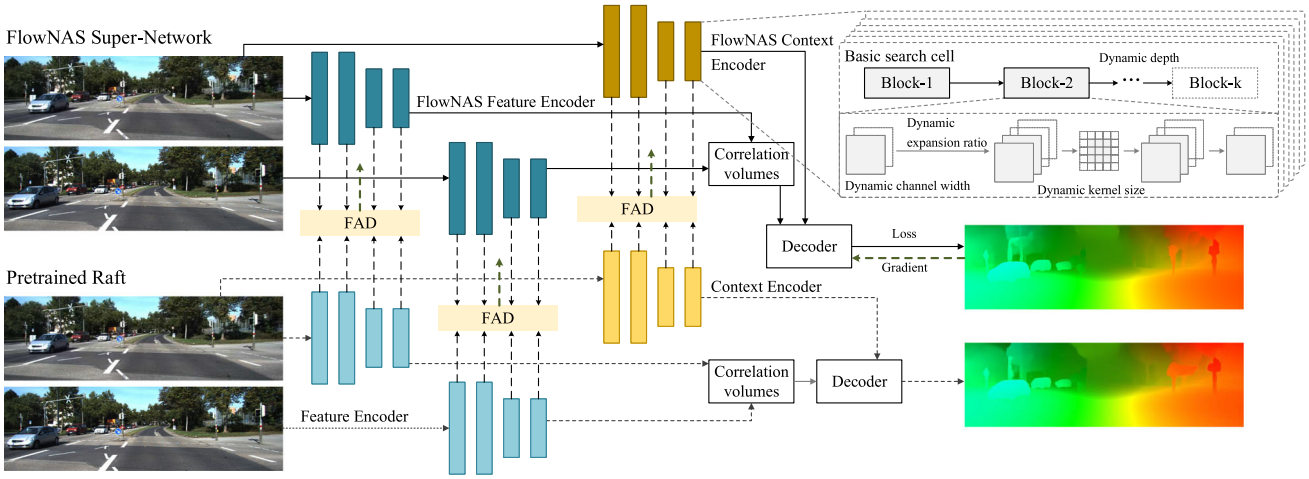


**Fig. 3 Overview of FlowNAS super-network training with Feature Alignment Distillation.** It uses pre-trained weights of handcrafted flow estimator (below) as a teacher to guide super-network (above) training

we apply a channel-wise alignment operation $g(\cdot)$ to $f_i^T$ and $f_i^S$ to align the number of the channel of two features. We then perform $L_2$ distance to estimate the difference between $g(f_i^T)$ and $g(f_i^S)$.

**Distillation Supervision:** The distillation loss computes the $L_2$ distance of $g(f_i^T)$ and $g(f_i^S)$ with exponentially increasing weight $\gamma^{N-i}$:

$$\mathcal{L}_{\mathcal{D}} = \sum_{i=1}^{N} \gamma^{N-i} \cdot L_2(g(f_i^T), g(f_i^S)). \tag{2}$$

The final training loss for FlowNAS is:

$$\mathcal{L} = \mathcal{L}_{flow} + \lambda \mathcal{L}_{\mathcal{D}}, \tag{3}$$

where $\mathcal{L}_{flow}$ is the original loss function for the flow method, and $\lambda$ is the weight to balance two losses. $\lambda$ is set to 1 in our experiments.

### 3.5 Channel-Wise Alignment

We present four types of channel-wise alignment operations in this section.

**Dynamic Channel Projection:** A straightforward approach to align the dynamic width of the student with the teacher is to use a weight-sharing dynamic linear layer $W_i \in \mathcal{R}^{c_i^S \times c_i^T}$ (Yu et al., 2019) to adjust the channel dimension of the sampled sub-network to be the same as the teacher model, where $g(f_i)$ is

$$g(f_i) = \begin{cases} W_i^T \cdot f_i & \text{, if } f_i \text{ comes from the student,} \\ f_i & \text{, if } f_i \text{ comes from the teacher.} \end{cases} \tag{4}$$

**Spatial Attention:** Attention modules play a critical role in knowledge transfer. In (Zagoruyko and Komodakis, 2017), it significantly improves the performance of the student model by forcing it to mimic the attention maps of the teacher in image classification. Similar to (Zagoruyko and Komodakis, 2017), we conduct a non-local module (Wang et al., 2018) for each layer of extracted feature map of the teacher and student model, reducing the channel dimension to 4.

**Channel Maximize:** Another simple scheme is to compress the width of both student and teacher into one without any learnable parameters. We provide two types of channel compression methods: maximize and average. The Channel Maximize operation selects a channel with maximum activation for each feature point along the spatial dimension ($x$, $y$), where $g(f_i)$ can be expressed as:

$$g(f_i)_{x,y} = \max_{k=1,2,\ldots,c_i} |f_{k,x,y}|. \tag{5}$$

**Channel Average:** Instead of focusing on the maximum activated channel, the average operation estimates the global average, where $g(f_i)$ can be formulated as:

$$g(f_i)_{x,y} = \frac{1}{c_i} \sum_{k=1}^{c_i} |f_{k,x,y}|. \tag{6}$$

We choose the Channel Maximize operation as the final channel-wise method for FlowNAS. More details can be found in Sect. 4.3.2.

## 4 Experimental Results

We present experimental results on the Sintel (Butler et al., 2012) and KITTI (Geiger et al., 2013) benchmarks to demon-

**Table 2** Results on Sintel and KITTI datasets

| Training Data | Method | Sintel (trainval) | | KITTI-15 (trainval) | | Sintel (test) | | KITTI-15 (test) | Params (M) | GFLOPs |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all | Clean | Final | F1-all | | |
| - | FlowFields (Bailer et al., 2015) | – | – | – | – | 3.75 | 5.81 | 15.31 | – | – |
| - | FlowFields++ (Schuster et al., 2018) | – | – | – | – | 2.94 | 5.49 | 14.82 | – | – |
| S | DCFlow (Xu et al., 2017) | – | – | – | – | 3.54 | 5.12 | 14.86 | – | – |
| S | MRFlow (Wulff et al., 2017) | – | – | – | – | 2.53 | 5.38 | 12.19 | – | – |
| C + T | HD3 (Yin et al., 2019) | 3.84 | 8.77 | 13.17 | 24.0 | – | – | – | – | – |
| | PWC-Net (Sun et al., 2018b) | 2.55 | 3.93 | 10.35 | 33.7 | – | – | – | 9.4 | 90.8 |
| | LiteFlowNet2 (Hui et al., 2019) | 2.24 | 3.78 | 8.97 | 25.9 | – | – | – | – | – |
| | VCN (Yang and Ramanan, 2019) | 2.21 | 3.68 | 8.36 | 25.1 | – | – | – | 6.2 | 96.5 |
| | MaskFlowNet (Zhao et al., 2020) | 2.25 | 3.61 | – | 23.1 | – | – | – | – | – |
| | FlowNet2 (Ilg et al., 2017) | 2.02 | 3.54 | 10.08 | 30.0 | 3.96 | 6.02 | – | – | – |
| | RAFT (Teed and Deng, 2020) | 1.43 | 2.71 | 5.04 | 17.4 | – | – | – | 5.3 | 388 |
| | SCV (Jiang et al., 2021b) | 1.29 | 2.95 | 6.80 | 19.3 | – | – | – | – | – |
| | SeparableFlow (Zhang et al., 2021) | 1.30 | 2.59 | 4.60 | 15.9 | – | – | – | 6.0 | 495 |
| | GMA (Jiang et al., 2021a) | 1.30 | 2.74 | 4.69 | 17.1 | – | – | – | 5.9 | 435 |
| | AGFlow (Luo et al., 2022) | 1.31 | 2.69 | 4.82 | 17.0 | – | – | – | – | – |
| | GMFlow (Xu et al., 2022) | 1.08 | 2.48 | 7.77 | 23.4 | – | – | – | 4.2 | 226 |
| | FlowNAS-PWCNet-S | 1.83 | 3.44 | – | – | – | – | – | 9.3 | 78.2 |
| | FlowNAS-PWCNet-K | – | – | 10.00 | 33.3 | – | – | – | 9.3 | 78.2 |
| | FlowNAS-RAFT-S | 1.31 | 2.68 | – | – | – | – | – | 5.2 | 368 |
| | FlowNAS-RAFT-K | – | – | 4.88 | 17.1 | – | – | – | 5.2 | 368 |
| | FlowNAS-GMFlow-S | 1.01 | 2.42 | – | – | – | – | – | 4.1 | 208 |
| | FlowNAS-GMFlow-K | – | – | 7.42 | 22.9 | – | – | – | 4.1 | 208 |
| C+T+S+K+H | LiteFlowNet2$^2$ (Hui et al., 2019) | (1.30) | (1.62) | (1.47) | (4.8) | 3.48 | 4.69 | 7.74 | – | – |
| | PWC-Net+ (Sun et al., 2018a) | (1.71) | (2.34) | (1.50) | (5.3) | 3.45 | 4.60 | 7.72 | 9.4 | 90.8 |
| | VCN (Yang and Ramanan, 2019) | (1.66) | (2.24) | (1.16) | (4.1) | 2.81 | 4.40 | 6.30 | 6.2 | 96.5 |
| | MaskFlowNet (Zhao et al., 2020) | – | – | – | – | 2.52 | 4.17 | 6.10 | – | – |

**Table 2** continued

| Training Data | Method | Sintel (trainval) | | KITTI-15 (trainval) | | Sintel (test) | | KITTI-15 (test) | Params (M) | GFLOPs |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all | Clean | Final | F1-all | | |
| | RAFT (Teed and Deng, 2020) | (0.77) | (1.27) | – | – | 1.94/1.61* | 3.18/2.86* | 5.10 | 5.3 | 388 |
| | SCV (Jiang et al., 2021b) | (0.86) | (1.75) | – | – | 1.72/1.77* | 3.60/3.88* | 6.17 | – | – |
| | SeparableFlow (Zhang et al., 2021) | (0.69) | (1.10) | (0.69) | (1.60) | 1.50 | 2.67 | 4.64 | 6.0 | 495 |
| | GMA (Jiang et al., 2021a) | (0.62) | (1.06) | (0.56) | (1.20) | –/1.39* | –/2.47* | 4.93 | 5.9 | 435 |
| | AGFlow (Luo et al., 2022) | (0.65) | (1.07) | (0.58) | (1.20) | –/1.43* | –/2.47* | 4.89 | – | – |
| | GMFlow (Xu et al., 2022) | – | – | – | – | 1.74 | 2.90 | 9.32 | 4.2 | 226 |
| | Baseline FlowNAS-RAFT-S | (0.66) | (1.12) | – | – | 1.68/1.60* | 3.10/2.74* | – | 5.1 | 354 |
| | Baseline FlowNAS-RAFT-K | – | – | (0.90) | (2.42) | – | – | 4.98 | 5.1 | 354 |
| | FlowNAS-PWCNet-S | (1.67) | (2.25) | – | – | 3.23 | 4.41 | – | 9.3 | 78.2 |
| | FlowNAS-PWCNet-K | – | – | (1.35) | (5.0) | – | – | 7.02 | 9.3 | 78.2 |
| | FlowNAS-RAFT-S | (0.77) | (1.25) | – | – | 1.88/1.60* | 3.18/2.98* | – | 5.2 | 368 |
| | FlowNAS-RAFT-K | – | – | (0.81) | (2.30) | – | – | 4.67 | 5.2 | 368 |
| | FlowNAS-GMFlow-S | (0.71) | (1.30) | – | – | 1.68 | 2.83 | – | 4.1 | 208 |
| | FlowNAS-GMFlow-K | – | – | (1.52) | (5.11) | – | – | 8.77 | 4.1 | 208 |

C+T denotes results after pre-training on FlyingChairs (C) and FlyingThings (T)
C+T+S+K+H indicates the results trained on combining Sintel, KITTI, and HD1K. *Results are evaluated with the "warm-start" strategy mentioned in RAFT (Teed and Deng, 2020). FlowNAS-RAFT-S/K denotes the sub-network searched on Sintel or KITTI

strate the effectiveness of FlowNAS. We show that the proposed method achieves state-of-the-art accuracy-efficiency trade-offs. In addition, we conduct ablation studies to discuss the effect of search space selection and FAD.

### 4.1 Implementation Details

**Basic Setups:** We use RAFT (Teed and Deng, 2020) as our baseline method and use the same training schedule if not otherwise specified. Specifically, we pre-train the network on FlyingChairs (Dosovitskiy et al., 2015) for 100k iterations and then on FlyingThings (Mayer et al., 2016) for another 100k iterations. We then finetune the network for Sintel evaluation on the combination of FlyingThings (Mayer et al., 2016), Sintel (Butler et al., 2012), KITTI-2015 (Menze and Geiger, 2015) and HD1K (Kondermann et al., 2016) for 100k iterations. Finally, we finetune the network on KITTI-2015 for an additional 50k iterations before evaluation.

Our primary evaluation metric is an average end-point error (AEPE), the mean pixel-wise flow error. In addition, KITTI uses the F1-all (%) metric, which refers to the percentage of optical flow vectors whose end-point error is greater than 3 pixels or over 5% of ground truth.

**FlowNAS Setups:** To search for the architecture with the best generalization ability and avoid overfitting, we split Sintel and KITTI training sets into training and validation subsets as performed by FlowNet (Dosovitskiy et al., 2015) and VCN (Yang and Ramanan, 2019). In the following, *the referred training set and validation set of Sintel and KITTI are the corresponding set after splitting*. In addition, *the original training set officially provided is denoted as trainval set*. For fair comparisons with other methods, we report the final results of FlowNAS trained on the trainval set after selecting the best architecture. During the resource-constrained search, the parameter upper bound $P$ is set to 5.3M in this paper. To prevent finding local minimal solutions, we combine the FAD loss and the prediction error of the sub-network as regularization terms. The entire architecture search optimization takes about 1.4 GPU days for Sintel and 0.25 GPU days for KITTI on an RTX 8000 GPU, which is negligible compared to the overall training time.

### 4.2 Main Results

Table 2 shows the performance of FlowNAS against state-of-the-art approaches on Sintel and KITTI-2015. As FlowNAS can be easily incorporated with existing flow estimators to find a better encoder, we apply it to PWC-Net, RAFT, and GMFlow. We use the resource-constrained search described in Sect. 3.3 to find the two most suitable sub-network architectures on Sintel and KITTI validation sets separately. The searched architectures are distinguished by the "-S" and "-K".

**Baseline FlowNAS:** We train the searched architecture of baseline FlowNAS from scratch using the same protocol as (Teed and Deng, 2020). Compared with RAFT (Teed and Deng, 2020), the baseline FlowNAS-RAFT has already achieved better results on both Sintel and KITTI-2015. The AEPE of Sintel is reduced from 2.86 to 2.74, and the F1-all of KITTI is reduced from 5.10% to 4.98%. The results demonstrate the necessity of redesigning the encoder for optical flow estimation and the effectiveness of the baseline FlowNAS.

**FlowNAS:** FlowNAS equipped with FAD further improves the accuracy of the baseline FlowNAS without retraining the sub-network. Table 2 shows, on the training set of FlyingChairs (C) + FlyingThings (T), by using the encoder searched by FlowNAS, FlowNAS-PWCNet-S achieves the APEP of 1.83 on clean pass and 3.44 on the final pass of Sintel, improving the performance of PWCNet by 28.2% (from 2.55 to 1.83) and 12.5% (from 3.93 to 3.44). FlowNAS-RAFT-S obtains 1.31 AEPE on the clean pass and 2.68 on the final pass of Sintel, which is comparable to RAFT-based state-of-the-art methods, GMA (Jiang et al., 2021a) and AGFlow (Luo et al., 2022), and lower than RAFT (Teed and Deng, 2020) by 8.4% (from 1.43 to 1.31) and 0.1% (from 2.71 to 2.68). Furthermore, GMFlow (Xu et al., 2022) achieves state-of-the-art performance on Sintel. Nevertheless, FlowNAS-GMFlow-S improves its performance even further, from 1.08 to 1.01 on the clean pass and 2.48 to 2.42 on the final pass, achieving new state-of-the-art results. For KITTI, FlowNAS-RAFT-K achieves an AEPE of 4.88 and F1-all score of 17.1% on the KITTI trainval set, which significantly surpasses SCV (Jiang et al., 2021b) by 28.1% (from 6.80 to 4.89) and 11.4% (from 19.3 to 17.1), respectively.

For online evaluation on KITTI, FlowNAS-RAFT-K improves RAFT by 8.4% (from 5.10% to 4.67%) on F1-all, outperforming the state-of-the-art handcrafted GMA (Jiang et al., 2021a), SCV (Jiang et al., 2021b) and AGFlow (Luo et al., 2022). FlowNAS-RAFT-K achieves comparable results with SeparableFlow on KITTI F1-all error while reducing the number of parameters from 6.0M to 5.2M. It is worth noting that SeparableFlow and GMA share the same encoder as RAFT, with changes made only on the decoder. Thus, the performance gain from a better encoder (FlowNAS) is comparable to that of the existing state-of-the-art decoder (GMA and SeparableFlow), as shown in Table 3. On the synthetic dataset Sintel, for three baseline decoders of optical flow estimation, PWCNet, RAFT, and GMFlow, FlowNAS achieves a lower AEPE than handcrafted encoders, proving that FlowNAS can search for a better encoder architecture.

Overall, the results demonstrate the excellent cross-dataset generalization of FlowNAS and the strong representation ability of the discovered architectures. Meanwhile, the results show that the encoder design is essential for optical flow estimation, and there is much room for improvement with existing modules.

**Table 3** Comparison of the effects brought by decoder and encoder

| Training Data | Encoder | Decoder | KITTI-15 (test) F1-all | Params | GFLOPs |
|---|---|---|---|---|---|
| C+T+S+K+H | RAFT (Teed and Deng, 2020) | RAFT (Teed and Deng, 2020) | 5.10 | 5.3M | 388 |
| | RAFT (Teed and Deng, 2020) | GMA (Jiang et al., 2021a) | 4.93 | 5.9M | 435 |
| | RAFT (Teed and Deng, 2020) | SeparableFlow (Zhang et al., 2021) | **4.63** | 6.0M | 495 |
| | FlowNAS | RAFT (Teed and Deng, 2020) | 4.67 | **5.2**M | **368** |

C+T+S+K+H indicates the results trained on combining Sintel, KITTI, and HD1K
The performance improvement brought by a better encoder is comparable to that of the existing state-of-the-art decoder

**Table 4** Ablation experiments of search space

| Training Data | Search Space | Sintel (trainval) | | KITTI-15 (trainval) | |
|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all |
| C + T | Conv | 1.68±0.09 | **2.91**±0.14 | 6.60±0.27 | 20.8±0.51 |
| | SepConv | **1.64**±0.13 | 2.96±0.13 | **6.46**±0.20 | **20.4**±0.41 |
| | ShuffleConv | 1.89±0.16 | 3.16±0.19 | 7.59±0.84 | 23.8±1.56 |
| | Combination | 1.72±0.13 | 2.99±0.18 | 6.59±0.37 | 20.5±0.78 |

The SepConv search space performs the best for the flow estimation task

**Table 5** Ablation experiments for Channel-wise Alignment operation

| Training Data | Feature alignment operation | Sintel (val) | | KITTI-15 (val) | |
|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all |
| C+T+S (train)+K (train)+H | Dynamic Channel Projection | 1.88 | 5.06 | 1.75 | 4.80 |
| | Spatial Attention | 22.3 | 30.1 | 26.8 | 83.0 |
| | Channel Maximize | **1.52** | **3.59** | **1.31** | **3.63** |
| | Channel Average | 1.60 | 3.98 | 1.41 | 3.89 |

The Channel Maximize operation achieves the best results

## 4.3 Ablation Study

In this section, we conduct ablation experiments to analyze the main modules of FlowNAS. Due to the submission limits of Sintel and KITTI, we present more experimental results based on our validation split. We choose RAFT (Teed and Deng, 2020) as our decoder for the ablation study. For illustration purposes, the results of two optimal sub-network architectures on Sintel and KITTI in the following tables (from Table 4 to Table 16) are listed in the same row.

### 4.3.1 Search Space

In Table 4, we evaluate four search spaces: Conv, SepConv, ShuffleConv, and combination. In these experiments, the width, depth, and kernel size are the variable of each search space. We randomly sample 15 architectures for each search space and report their average results where the models are trained from scratch. To save computational overhead, the sampled sub-networks are trained on FlyingChairs (C) + FlyingThings (T).

As SepConv performs best among all search space candidates for the flow estimation task, we choose SepConv as the essential convolution operation for FlowNAS. Furthermore, combining different convolution operations does not lead to better performance than SepConv. The results can be attributed to two factors. First, the search space of combinations is $6^3$ times larger than that of SepConv, which causes insufficient training. Second, the coupling of SepConv and other operations brings difficulties for the evolutionary algorithm to converge into a better local minimum.

### 4.3.2 Feature Alignment Distillation

We conduct the following experiments to validate the proposed FAD module.

**Channel-wise Alignment**: We train FlowNAS super-network with the proposed alignment operations on the combined Sintel, KITTI, and HD1K datasets, and report the performances of their best-searched architectures by directly inheriting weights from super-networks. As shown in Table 5, the operations with learnable parameters (Dynamic Channel Projection and Spatial Attention) are less effective than non-parametric operations (Channel Maximize and Average). It can be attributed that the additional learnable weights from these operations may negatively affect the training process

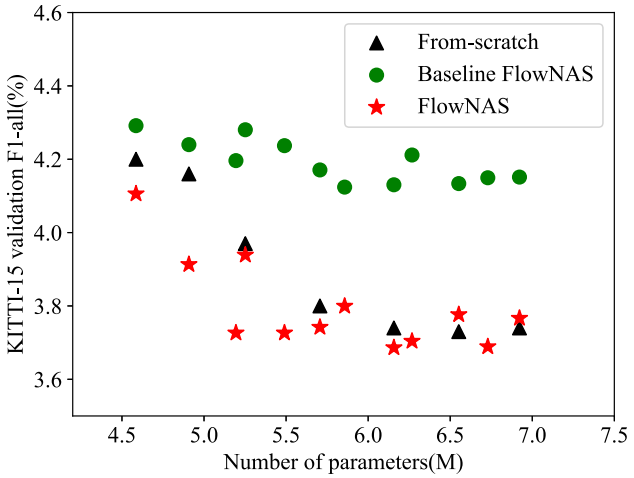


**Fig. 4 KITTI-15 validation results of several random sub-network architectures.** The weights are obtained from From-scratch, baseline FlowNAS, and FlowNAS

of the super-network since the number of parameters is large. For example, the parameters of the Non-local modules are 17% of those of the super-network. Thus, it is prone to overfit the encoder feature. In addition, the Channel Maximize operation performs better than Channel Average. This can be explained that the Channel Maximize operation paying more attention to the most discriminate parts in feature maps (Zagoruyko and Komodakis, 2017) (e.g., the foreground objects) compared to Channel Average. Based on the experiment results and analysis, we choose the Channel Maximize operation as the final channel-wise method for FlowNAS.

**Effectiveness of FAD:** Fig. 4 shows the performance of sub-networks from From-scratch, baseline FlowNAS, and FlowNAS. We randomly sample 12 sub-networks and report their performance with inherited weights from the baseline FlowNAS super-network and FlowNAS super-network. In addition, we randomly sample 6 sub-networks and train them from scratch as their stand-alone performance. For all sampled architectures, FlowNAS reduces their F1-all error by 0.18∼0.51 compared with baseline FlowNAS, demonstrating the effectiveness of FAD for better super-network optimization. Furthermore, super-network weights from FlowNAS achieve similar or even lower F1-all errors than their stand-alone performance, showing the strong representation ability of FlowNAS.

**Table 7** Kendall's Tau (KTau) correlation coefficient

| Dataset | Method | KTau |
|---|---|---|
| Sintel | Baseline FlowNAS | 0.213 |
| | FlowNAS | **0.590** |
| KITTI | Baseline FlowNAS | 0.451 |
| | FlowNAS | **0.768** |

With FAD, FlowNAS achieves a higher rank correlation than Baseline FlowNAS

Table 6 shows the performance of the two best architectures searched by the baseline FlowNAS and FlowNAS, namely baseline FlowNAS-RAFT and FlowNAS-RAFT. When inheriting weights from their corresponding super-networks, the performance of baseline FlowNAS-RAFT and FlowNAS-RAFT are shown in lines 1 and 3. FlowNAS-RAFT achieves a lower F1-all error of 0.42 than the baseline FlowNAS-RAFT, validating the effectiveness of FAD for super-network training. Finally, the performance of FlowNAS-RAFT when trained with or without FAD is shown in lines 4 and 5. It is worth noting that FAD can also improve the training of sub-network by reducing 0.06 F1-all error, demonstrating the effectiveness of FAD for flow estimator training.

In addition to the performance improvement of the super-network, FAD can improve the rank correlation. We randomly sample 24 architecture configurations and calculate the Kendall's Tau (KTau) correlation coefficient between the performance of weight-inherited and retrained sub-networks. Table 7 shows that FAD can improve the KTau rank correlation from 0.213 to 0.590 on Sintel and from 0.451 to 0.768 on KITTI. The proposed FAD uses feature distillation in each encoder block and guides the features from all sub-network blocks to converge into the same feature space. As such, the optimal objective of a sub-network can be factorized into several independently block-wise feature optimization problems (Li et al., 2020a). Thus, the training interference between each block is alleviated, and the rank correlation is improved. In addition, FAD can reduce the training instability of the encoder and accelerate the whole super-network train-

**Table 6** Ablation experiments for distillation

| Training Data | Method | Weight | KITTI-15 (val) | |
|---|---|---|---|---|
| | | | F1-epe | F1-all |
| C+T+S (train)+K (train)+H | Baseline FlowNAS-RAFT | Inherit | 1.52 | 4.05 |
| | Baseline FlowNAS-RAFT | From-scratch | 1.36 | 3.69 |
| | FlowNAS-RAFT | Inherit | **1.31** | **3.63** |
| | FlowNAS-RAFT | From-scratch | 1.38 | 3.84 |
| | FlowNAS-RAFT | From-scratch* | 1.36 | 3.78 |

The entry "from-scratch*" denotes training the sub-network with Feature Alignment Distillation

**Table 8** Ablation experiments for additional supervision

| Training Data | Supervision Loss | Sintel (val) | | KITTI-15 (val) | |
|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all |
| C+T+S (train)+K (train)+H | $\mathcal{L}_{flow}$ | 1.64 | 4.06 | 1.52 | 4.05 |
| | $\mathcal{L}_{flow} + \mathcal{L}_{D}$ | **1.52** | **3.59** | **1.31** | **3.63** |
| | $\mathcal{L}_{simple} + \mathcal{L}_{D}$ | 30.8 | 39.8 | 32.3 | 98.7 |
| | $\mathcal{L}_{flow} + \mathcal{L}_{simple} + \mathcal{L}_{D}$ | 1.90 | 5.51 | 1.50 | 3.94 |

The simple decoder does not help refine the super-network

**Table 9** Performance comparison using two knowledge distillation methods

| Training Data | Knowledge Distillation Method | Sintel (val) | | KITTI-15 (val) | | Training Time |
|---|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all | |
| C+T+S (train)+K (train)+H | None | 1.64 | 4.06 | 1.52 | 4.05 | 1× |
| | Conventional Knowledge Distillation | 1.58 | 4.02 | 1.50 | 4.04 | 1.6× |
| | Feature Alignment Distillation | **1.52** | **3.59** | **1.31** | **3.63** | 1.2× |

FAD achieves better results than conventional knowledge distillation with less training time

**Table 10** Ablation experiments for the teacher model in FAD

| Training Data | Teacher Model | Sintel (val) | | KITTI-15 (val) | |
|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all |
| C+T+S (train)+K (train)+H | None | 1.64 | 4.06 | 1.52 | 4.05 |
| | Largest Sub-network of FlowNAS | 1.68 | 3.99 | 1.75 | 4.12 |
| | PWC-Net+ | 1.67 | 3.90 | 1.69 | 4.09 |
| | RAFT | **1.52** | **3.59** | **1.31** | **3.63** |
| | SeparableFlow | 1.58 | 3.70 | 1.38 | 3.70 |
| | GMA | 1.55 | 3.68 | 1.47 | 3.98 |
| | GMFlow | 1.62 | 3.87 | 1.49 | 4.00 |

Using RAFT as the teacher model can obtain the best results on both Sintel and KITTI validation sets

ing process, enabling the weights from the super-network to be closer to the optimal weights for the sub-networks. Thus, the performance of the sub-networks with inherited weights from the super-network is closer to their stand-alone performance (Fig. 4). This can further improve the rank correlation.

**Other Supervision from the Decoder:** Similar to the existing dense image prediction method (Zhao et al., 2017), we try to refine the initial features of the super-network following. Specifically, we simplify the decoder by directly estimating optical flow based on the similarities between two pyramid feature maps from the super-network, namely the simple decoder. The supervision loss of the simple decoder is denoted as $\mathcal{L}_{simple}$. As shown in Table 8, unlike the additional loss of other tasks, e.g., semantic segmentation, our super-network performs worse when using the simple decoder for supervision. The reason is that the simple decoder does not learn from training data, as opposed to the decoder of recent deep methods that can progressively refine flow estimation based on two pyramids of features and cost volumes.

Aside from the simple decoder, we use the outputs of the original decoder in RAFT to perform knowledge distillation using the conventional knowledge distillation in (Gou et al., 2021). Table 9 shows that performing conventional knowledge distillation achieves worse results than FAD and takes more time for super-network training. Although conventional knowledge distillation is simple, distilling knowledge on feature is shown to be important for representation learning (Romero et al., 2015; Gou et al., 2021), since intermediate feature maps provide more favorable information and supervision for the learning of the student model. In this work, FlowNAS aims to find a better encoder architecture for optical flow estimation and thus relies much on feature representation learning. Compared with conventional knowledge distillation, FAD can simultaneously guide and facilitate the feature learning process of all sub-networks. In addition, the output of a high-performing teacher network is similar to the ground truth. As such, only learning the output of a teacher network in conventional knowledge distillation for the student is similar to training with the ground truth.

**Table 11** Overall comparison with cutting edge handcrafted flow estimators

| Method | Params | GFLOPs | Time | KITTI (test) Fl-all (%) |
|---|---|---|---|---|
| PWCNet+ (Sun et al., 2018a) | 9.4M | **90.8** | **0.02** | 7.72 |
| VCN(Yang and Ramanan, 2019) | 6.2M | 96.5 | 0.11 | 6.30 |
| RAFT (Teed and Deng, 2020) | 5.3M | 388 | 0.17 | 5.10 |
| GMA (Jiang et al., 2021a) | 5.9M | 435 | 0.20 | 4.93 |
| SeparableFlow (Zhang et al., 2021) | 6.0M | 495 | 0.23 | **4.63** |
| FlowNAS-RAFT-K | **5.2M** | 368 | 0.19 | 4.67 |

FlowNAS enables the most advanced accuracy-efficiency trade-off

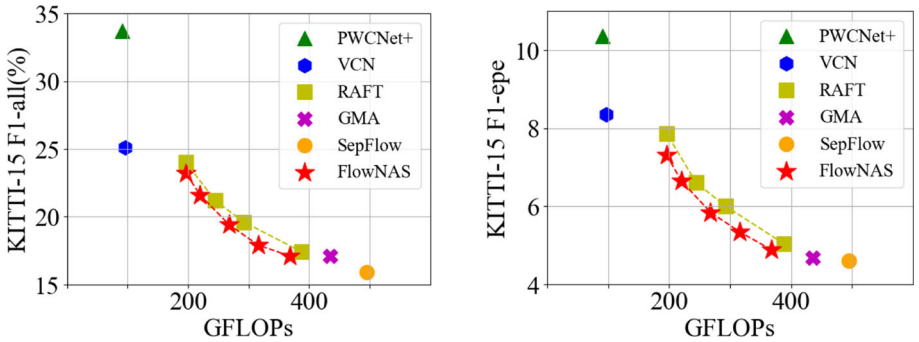


**Fig. 5 Trade-offs between accuracy and GFLOPs**. FlowNAS obtains a better Pareto front

**Selection of the Teacher Model:** The teacher model is an important component in the proposed FAD method. We train FlowNAS super-network with the supervision of different teacher models. Table 10 shows that using the handcrafted estimators (such as PWC-Net+, RAFT, and SeparableFlow) as the teacher can obtain better results than the largest sub-network in our searched space. The reason is that the handcrafted estimators with pre-trained weights as a teacher model can provide stronger human prior knowledge to guide the super-network training. The ablation studies show that while SeparableFlow, GMA, and GMFlow yield better flow estimation results than RAFT, employing RAFT as the teacher model for FlowNAS yields the optimal performance in terms of super-network performance. This result can be explained by the consistency of the architecture between the teacher and student models (Yuan et al., 2021; Yang et la., 2022), i.e., the decoder architecture of FlowNAS is the same as RAFT. When teacher and student models have a similar architecture, it is easy for the student to learn the teacher's feature map. The distinct decoder architecture employed by SeparableFlow, GMA, and GMFlow models makes it difficult to transfer knowledge to FlowNAS through distillation.

### 4.4 Model Parameter and Inference Time

FlowNAS improves the performance of the baseline model and reduces the number of parameters, inference latency, and GFLOPs. We measure the inference time of existing flow networks on the same machine with one RTX 8000 GPU. We set the input size as 384×1280 to measure GFLOPs and inference time. The iterations of RAFT (Teed and Deng, 2020), GMA (Jiang et al., 2021a), SperarableFlow (Zhang et al., 2021), and FlowNAS-RAFT-K is set to 24. As shown in Table 11 and Fig. 1, FlowNAS achieves the best accuracy-efficiency trade-offs. In addition, FlowNAS achieves comparable results as SperarableFlow while reducing 13% parameters and 26% FLOPs.

Since RAFT decoder has an iterative module, we jointly search the encoder architecture and iteration number of the decoder to obtain the Pareto front of accuracy and GFLOPs of FlowNAS, as shown in Fig. 5. We adjust the iteration number of RAFT to obtain its Pareto fronts. As a result, we can find that FlowNAS achieves a better Pareto front of accuracy and GFLOPs than RAFT (Teed and Deng, 2020).

### 4.5 Cross-Model Generalization

FlowNAS can find a more effective encoder with fewer parameters and GFLOPs for existing methods for optical flow estimation. Table 12 shows the results when applying FlowNAS to existing top-performing flow estimators. By replacing with the encoder from FlowNAS, the performance of these methods on the clean pass of the Sintel and KITII test set is improved.

### 4.6 NAS for Decoder

Decoder design plays an essential role in deep models for optical flow estimation. However, the decoders are tightly coupled with the pipeline of flow estimation in the existing

**Table 12** Cross-model generalization of FlowNAS

| Training Data | Encoder | Decoder | Sintel (test) Clean | Sintel (test) Final | KITTI-15 (test) F1-all | Params | GFLOPs |
|---|---|---|---|---|---|---|---|
| C+T+S+K+H | PWC-Net+ (Sun et al., 2018a) | PWC-Net+ (Sun et al., 2018a) | 3.45 | 4.60 | 7.72 | 9.4M | 90.8 |
| | FlowNAS | | **3.23** | **4.41** | **7.02** | **9.3M** | **78.2** |
| | RAFT (Teed and Deng, 2020) | RAFT (Teed and Deng, 2020) | 1.94 | **3.18** | 5.10 | 5.3M | 388 |
| | FlowNAS | | **1.88** | **3.18** | **4.67** | **5.2M** | **368** |
| | SeparableFlow (Zhang et al., 2021) | SeparableFlow (Zhang et al., 2021) | 1.70† | 2.79† | 4.93† | **6.0M** | 495 |
| | FlowNAS | | **1.63** | **2.76** | **4.72** | **6.0M** | **482** |
| | GMA (Jiang et al., 2021a) | GMA (Jiang et al., 2021a) | 1.89† | 2.94† | 5.14† | 5.9M | 435 |
| | FlowNAS | | **1.63** | **2.75** | **5.03** | **5.8M** | **417** |
| | GMFlow (Xu et al., 2022) | GMFlow (Xu et al., 2022) | 1.74 | 2.90 | 9.32 | 4.2M | 226 |
| | FlowNAS | | **1.68** | **2.83** | **8.77** | **4.1M** | **208** |

C+T+S+K+H indicates the results trained on the combination of Sintel, KITTI, and HD1K. † means our reproduced results by consulting the authors

deep models. For example, VCN uses a warp module to fuse features and predicted flows for the feature pyramid, RAFT adopts a motion encoder module for the GRU-based update module, and GMFlow directly enhances the encoded features by a transformer module. Thus, it is challenging to separate decoder operation modules from such a model and construct a search space. To avoid this problem, we propose to search the connections of flow decoders, namely the connection search.

#### 4.6.1 Connection Search

Existing flow decoders are usually constructed in a coarse-to-fine manner. For example, RAFT uses the recurrent decoder to refine the flow predictions. The output of the RAFT decoder is regarded as the input for the next iteration:

$$f_i = \text{Decoder}(f_{i-1}). \tag{7}$$

Thus, the connection of RAFT decoder can be viewed as a plain structure. In the connection search, we add searchable skip connections from each output of previous iterations to the input of the current iteration. The output of $i$-th iteration can be expressed as:

$$f_i = \text{Decoder}(h(f_{i-1}, w_{i,i-2} f_{i-2}, \ldots, w_{i,1} f_1)), \tag{8}$$

where $w \in \{0, 1\}$ is the searchable binary weight, $h(\cdot)$ is the fusion function to aggregate features. The value of $w_{i,j}$ indicates the availability of the skip connection between the output obtained by $j$-th iteration and the input in $i$-th iteration. During architecture sampling, the value of $w_{i,j}$ is chosen form $\{0, 1\}$.

The number of iterations and searchable binary weights $w$ are fixed after initialization. Thus, we cannot directly increase the number of iterations during inference. Otherwise, we will introduce extra binary weights $w$ and skip connections for additional iterations. To address this issue, we introduce the inner and outer loop by repeating the searched connections for additional iterations. The inner loop is the iteration process mentioned in Equation (8), and the number of iterations in the inner loop is fixed after initialization. The outer loop consists of several inner loops, i.e., the output of one inner loop is the input of the next inner loop. Thus, assuming the number of iterations after initialization in the inner loop is $n$, we can obtain the results of $(n \times k + i)$-th iteration (the $i$-th iteration in the $(k + 1)$-th inner loop) by:

$$f_{n\times k+i} = \text{Decoder}(h\ (f_{n\times k+i-1}, w_{i,i-2} f_{n\times k+i-2}, \ldots, w_{i,1} f_{n\times k+1})). \tag{9}$$

**Table 13** Ablation experiments for the fusion function of decoder search

| Training Data | Fusion Function | Sintel (val) Clean | Sintel (val) Final | KITTI-15 (val) F1-epe | KITTI-15 (val) F1-all |
|---|---|---|---|---|---|
| C+T+S (train)+K (train)+H | None | **1.52** | 3.59 | 1.31 | 3.63 |
| | Average | 1.78 | 4.51 | 1.63 | 4.10 |
| | Summation | 1.61 | 3.89 | 1.40 | 3.78 |
| | Dynamic Summation | 1.54 | **3.57** | **1.29** | **3.50** |

"None" denotes the original RAFT decoder without NAS. Dynamic summation achieves the best results among the three fusion functions

**Table 14** Ablation experiments of iteration number and decoupled search for decoder search

| Training Data | Number of Iterations in One Inner Loop | Decoupled Search | Sintel (val) | | KITTI-15 (val) | |
|---|---|---|---|---|---|---|
| | | | Clean | Final | F1-epe | F1-all |
| C+T+S (train)+K (train)+H | 1 | – | **1.52** | 3.59 | 1.31 | 3.63 |
| | 4 | × | 1.54 | **3.57** | **1.29** | **3.50** |
| | | ✓ | 1.54 | **3.57** | **1.29** | **3.50** |
| | 6 | × | 1.57 | 3.62 | 1.33 | 3.78 |
| | | ✓ | 1.56 | 3.62 | 1.30 | 3.79 |
| | 12 | × | 1.60 | 3.78 | 1.35 | 4.00 |
| | | ✓ | 1.57 | 3.71 | 1.32 | 3.89 |

Decouple search brings a better sub-network architecture. The best result is achieved when the number of iterations in one inner loop is 4

**Table 15** Connection search for the flow decoder

| Training Data | Encoder | Decoder | Sintel (test) | | KITTI-15 (test) |
|---|---|---|---|---|---|
| | | | Clean | Final | F1-all |
| C+T+S+K+H | RAFT (Teed and Deng, 2020) | RAFT (Teed and Deng, 2020) | 1.94/1.61* | 3.18/2.86* | 5.10 |
| | | Connection Search | 1.92/1.64* | **3.10**/3.00* | 4.99 |
| | FlowNAS | RAFT (Teed and Deng, 2020) | **1.88/1.60*** | 3.18/2.98* | **4.67** |
| | | Connection Search | 1.95/1.63* | 3.20/**2.82*** | 4.73 |

*Results are evaluated with the "warm-start" strategy mentioned in RAFT (Teed and Deng, 2020). NAS has limited benefits for the flow decoder

#### 4.6.2 Fusion Function

To minimize additional parameters and computation of fusion function $h(\cdot)$, we design three simple fusion functions: average, summation, and dynamic summation. The average or summation fusion function fuses features from previous iterations by averaging or summing all features in the channel dimension. Dynamic summation fusion function first uses a $1 \times 1$ convolution layer followed by batch normalization and a ReLU layer to re-weight the value of each channel for input features. It then sums up the new features in the channel dimension. We can formulate the Dynamic summation fusion function as:

$$h(f_{i-1}, w_{i,i-2} f_{i-2}, ..., w_{i,1} f_1) = f_{i-1} + \sum_{j=1}^{i-2} w_{i,j} ReLU(BN(W f_j^T)). \quad (10)$$

The results using three fusion functions are shown in Table 13. Overall, the method using the dynamic summation function performs best.

#### 4.6.3 Number of Iterations and Decoupled Search

We study the effect of iteration number in one inner loop for the searchable flow decoder. We set the total number of iterations during training to be 12 following RAFT. Experiment results show that the configuration of the connections during the architecture search is more difficult to converge with a larger number of iterations in one inner loop. In contrast, the encoder architecture can easily converge in a few epochs. To alleviate this issue, we propose decoupled evolutionary search algorithm. Specifically, we first use the proposed resource-constrained search in Sect. 3.3 to search the encoder and decoder architecture configuration together for T epochs. The encoder architecture converges (i.e., the encoder architectures are the same for the top 5 sub-network architecture) during the first search stage. Then, in the second search stage, we fix the encoder architecture and only search decoder connections for other T epochs.

Table 14 shows that, with the proposed decoupled search, we can obtain better sub-networks for the different number of iterations. However, the results worsen when the number of iterations in one inner loop is large. This can be attributed to a decoder with complicated connections likely resulting in unstable training.

#### 4.6.4 Connection Search Results

To evaluate the effect of the proposed connection search, we equip it with two flow encoders, i.e., RAFT, and FlowNAS, and test the networks on Sintel and KITTI benchmark datasets. The results are shown in Table 15. For both encoders, the decoder with connection search has marginal improvement on Sintel. However, on KITTI, the decoder with connection search decreases the performance from 4.67

**Table 16** GRU search for the flow decoder

| Training Data | Decoder | Sintel (val) | | KITTI-15 (val) | |
|---|---|---|---|---|---|
| | | Clean | Final | F1-epe | F1-all |
| C+T+S (train)+K (train)+H | RAFT | **1.52** | 3.59 | 1.31 | 3.63 |
| | Connection Search | 1.54 | 3.57 | **1.29** | **3.50** |
| | GRU Search | **1.52** | **3.56** | 1.33 | 3.61 |

GRU search brings marginal improvements on Sintel

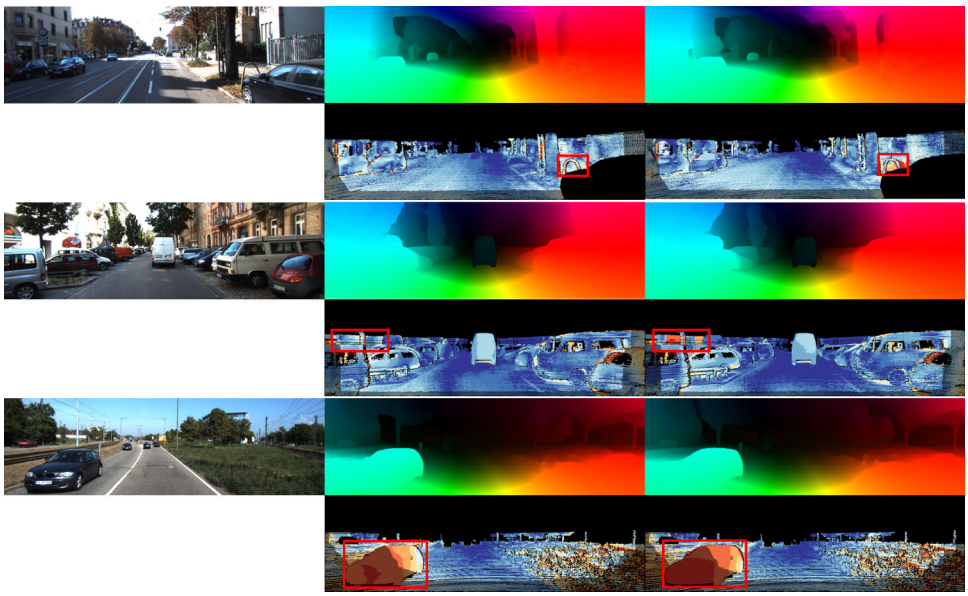

**Fig. 6 Visualization results on the KITTI 2015 test-dev.** The left column is the input image. The middle column shows the estimated flow and error maps of FlowNAS. The right column presents the estimated flow and error maps of RAFT baseline. From blue to red, the error of the estimated flow increases in the error map (Color figure online)

to 4.73 for FlowNAS. These results indicate that NAS has limited benefits for the flow decoder. The reason is that the current flow decoder design involves much human prior knowledge. The designed decoder architecture is rather delicate and reaches a bottleneck. Thus, the modification from NAS for the decoder architecture may bring negative effects. In contrast, with the proposed FlowNAS, which designs a more suitable encoder for optical flow estimation, we can significantly boost the performance of any existing flow estimators (Table 12).

#### 4.6.5 GRU Search

One possible reason for the connection search not performing well for the RAFT decoder is that the GRU may already model the connection between past iterations. Thus, we redesign the search space of the decoder architecture, i.e., only searching the GRU architecture, including the kernel size and channels of the convolution in GRU, and maintaining the overall structure of the decoder unchanged. It is worth noting that DARTS (Liu et al., 2019b) and ENAS (Pham et al., 2018) undertake architecture explorations on recurrent cells. However, these approaches apply NAS on recurrent cells only for 1D text data, while we mainly focus on NAS for 2D images. In addition, the constructed search space in DARTS (Liu et al., 2019b) and ENAS (Pham et al., 2018) for recurrent cells is simple. Within the search space, non-parametric activation functions and node connections are searchable operations, whereas the type of convolutional operations in recurrent cells remains fixed. In contrast, the RAFT decoder adopts a complex recurrent cell structure (i.e., GRU), consisting of a MotionEncoder and a SepConvGRU module with horizontal and vertical convolution layers for feature updates. This complex cell structure allows us to construct a more complicated search space, which DARTS and ENAS may not handle.

Table 16 shows that only searching the GRU architecture achieves slightly better results on Sintel while performing worse on KITTI than the connection search. The reason is that GRU search introduces more trainable parameters in the search space than the connection search (4 M *vs.* 0.3M). GRU Search is prone to overfit during the training process on KITTI, which only contains 160 image pairs on our split training set. In contrast, the split training set of Sintel has 1600 image pairs, which alleviates the overfitting problem and allows the network to find a better GRU architecture. Overall, the strategies for a better decoder achieve marginal improvements to the encoder search. As a result, we only keep the NAS method for the flow encoder design.

### 4.7 Visualization

We present the flow results of FlowNAS and RAFT on the KITTI test-dev in Fig. 6. The figures are input images, flow visualization, and error maps of FlowNAS and RAFT from left to right. With a better encoder, FlowNAS improves the flow details of the background. Furthermore, for a fast-moving object (the car in the last row of Fig. 6), FlowNAS can capture more accurate movement than the original RAFT encoder.

## 5 Conclusions

In this work, we address the problem of designing optical flow estimators automatically. We propose the FlowNAS model to find the optimal encoder structure specifically for optical flow. We study different search spaces and construct the super-network on the best search space for neural architecture search. To improve the accuracy of the super-network and remove the retraining stage of sub-networks, we propose the Feature Alignment Distillation module, which guides the training of all sub-networks of the super-network. The proposed FlowNAS model can be easily incorporated with existing methods to find a better encoder. In addition, we discuss the effect of NAS on the flow decoder search and show that NAS plays a lesser role in the flow decoder design. Extensive experimental results show that FlowNAS achieves state-of-the-art performance with the trade-off between accuracy and efficiency on the Sintel and KITTI benchmark datasets.

**Acknowledgements** This work was supported in part by National Natural Science Foundation of China under Grant 62176007. This work was also a research outcome of Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology).

## References

Bailer, C., Taetz, B., Stricker, D. (2015). Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In IEEE International Conference on Computer Vision (ICCV), pp 4015–4023

Bender, G., Kindermans, P., Zoph, B., Vasudevan, V., Le, Q. V. (2018). Understanding and simplifying one-shot architecture search. In International Conference on Machine Learning (ICML)

Biswas, B., Kr Ghosh, S., Hore, M., & Ghosh, A. (2022). Sift-based visual tracking using optical flow and belief propagation algorithm. *The Computer Journal, 65*(1), 1–17.

Brock, A., Lim, T., Ritchie, J. M., Weston, N. (2018). SMASH: one-shot model architecture search through hypernetworks. In International Conference on Learning Representations (ICLR)

Butler, D. J., Wulff, J., Stanley, G. B., Black, M. J. (2012) A naturalistic open source movie for optical flow evaluation. In European Conference on Computer Vision (ECCV), pp 611–625

Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S. (2020). Once-for-all: Train one network and specialize it for efficient deployment. In International Conference on Learning Representations (ICLR)

Cai, H., Zhu, L., Han, S. (2019). Proxylessnas: Direct neural architecture search on target task and hardware. In International Conference on Learning Representations (ICLR)

Chen, Y., Guo, Y., Chen, Q., Li, M., Zeng, W., Wang, Y., Tan, M. (2021). Contrastive neural architecture search with neural architecture comparators. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Cheng, X., Zhong, Y., Harandi, M., Dai, Y., Chang, X., Li, H., Drummond, T., Ge, Z. (2020). Hierarchical neural architecture search for deep stereo matching. In Neural Information Processing Systems (NeurIPS)

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1800–1807

Chu, X., Zhang, B., Xu, R., Li, J. (2021). Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. IEEE International Conference on Computer Vision (ICCV)

Chu, X., Zhou, T., Zhang, B., Li, J. (2020). Fair DARTS: eliminating unfair advantages in differentiable architecture search. In European Conference on Computer Vision (ECCV)

de Jong, D., Paredes-Vallés, F., de Croon, G. (5555). How do neural networks estimate optical flow a neuropsychology-inspired study. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI) pp 1–1

Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T. (2015). Flownet: Learning optical flow with convolutional networks. In IEEE International Conference on Computer Vision (ICCV), pp 2758–2766

Fortun, D., Bouthemy, P., & Kervrann, C. (2015). Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding (CVIU), 134*, 1–21.

Gao, S., Huang, F., Cai, W., Huang, H. (2021). Network pruning via performance maximization. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research, 32*(11), 1231–1237.

Gou, J., Yu, B., Maybank, S. J., Tao, D. (2021). Knowledge distillation: A survey. International Journal on Computer Vision (IJCV)

Guo, Y., Zheng, Y., Tan, M., Chen, Q., Li, Z., Chen, J., Zhao, P., Huang, J. (2022). Towards accurate and compact architectures via neural architecture transformer. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI)

Guo, Z., Zhang, X., Mu, H., Heng, W., Liu, Z., Wei, Y., Sun, J. (2020). Single path one-shot neural architecture search with uniform sampling. In: European Conference on Computer Vision (ECCV)

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 770–778

Hui, T. W., Tang, X., Loy, C. C. (2019). A lightweight optical flow cnn–revisiting data fidelity and regularization. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI)

Hur, J., Roth, S. (2019). Iterative residual refinement for joint optical flow and occlusion estimation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 5754–5763

Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 2462–2470

Jiang, H., Learned-Miller, E. G. (2021). Dcvnet: Dilated cost volume networks for fast optical flow. arXiv:2103.17271

Jiang, S., Campbell, D., Lu, Y., Li, H., Hartley, R. (2021a). Learning to estimate hidden motions with global motion aggregation. In

IEEE International Conference on Computer Vision (ICCV), pp 9772–9781

Jiang, S., Lu, Y., Li, H., Hartley, R. (2021b). Learning optical flow from a few matches. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 16592–16600

Kondermann, D., Nair, R., Honauer, K., Krispin, K., Andrulis, J., Brock, A., Gussefeld, B., Rahimimoghaddam, M., Hofmann, S., Brenner, C., et al. (2016). The hci benchmark suite: Stereo and flow ground truth with uncertainties for urban autonomous driving. In IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp 19–28

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images

Li, C., Peng, J., Yuan, L., Wang, G., Liang, X., Lin, L., Chang, X. (2020a). Block-wisely supervised neural architecture search with knowledge distillation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Li, R., Tan, R. T., Cheong, L. (2020b). All in one bad weather removal using architectural search. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3172–3182

Liang, T., Wang, Y., Tang, Z., Hu, G., Ling, H. (2021). OPANAS: one-shot path aggregation network architecture search for object detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 10195–10203

Liu, C., Chen, L., Schroff, F., Adam, H., Hua, W., Yuille, A. L., Fei-Fei, L. (2019a). Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 82–92

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L., Fei-Fei, L., Yuille, A. L., Huang, J., & Murphy, K. (2018). Progressive neural architecture search. *European Conference on Computer Vision (ECCV), 11205*, 19–35.

Liu, H., Simonyan, K., Vinyals, O., Fernando, C., Kavukcuoglu, K. (2018b). Hierarchical representations for efficient architecture search. In International Conference on Learning Representations (ICLR)

Liu ,H., Simonyan, K., Yang, Y. (2019b). DARTS: differentiable architecture search. In International Conference on Learning Representations (ICLR)

Liu, J., Zhuang, B., Zhuang, Z., Guo, Y., Huang, J., Zhu, J., Tan, M. (2022). Discrimination-aware network pruning for deep model compression. IEEE Transactions on Pattern Recognition and Machine Intelligence (PAMI)

Liu, R., Ma, L., Zhang, J., Fan, X., Luo, Z. (2021). Retinex-inspired unrolling with cooperative prior architecture search for low-light image enhancement. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 10561–10570

Luo, A., Yang, F., Luo, K., Li, X., Fan, H., Liu, S. (2022), Learning optical flow with adaptive graph reasoning. In Association for the Advancement of Artificial Intelligence (AAAI)

Mayer, N., Ilg, E., Hausser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4040–4048

Menze, M., Geiger, A. (2015). Object scene flow for autonomous vehicles. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3061–3070

Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J. (2018). Efficient neural architecture search via parameters sharing. In International Conference on Machine Learning (ICML)

Ranjan, A., Black, M. J. (2017). Optical flow estimation using a spatial pyramid network. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4161–4170

Real, E., Aggarwal, A., Huang, Y., Le, Q. V. (2019). Regularized evolution for image classifier architecture search. In Association for the Advancement of Artificial Intelligence (AAAI), pp 4780–4789

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In International Conference on Learning Representations (ICLR)

Saikia, T., Marrakchi, Y., Zela, A., Hutter, F., Brox, T. (2019). Autodispnet: Improving disparity estimation with automl. In IEEE International Conference on Computer Vision (ICCV)

Schuster, R., Bailer, C., Wasenmüller, O., Stricker, D. (2018). Flowfields++: Accurate optical flow correspondences meet robust interpolation. In IEEE International Conference on Image Processing (ICIP), pp 1463–1467

Sun, D., Yang, X., Liu, MY., Kautz, J. (2018a). Models matter, so does training: An empirical study of cnns for optical flow estimation. arXiv preprint arXiv:1809.05571

Sun, D., Yang, X., Liu, M. Y., Kautz, J. (2018b). Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 8934–8943

Sun, S., Kuang, Z., Sheng, L., Ouyang, W., Zhang, W. (2018c). Optical flow guided feature: A fast and robust motion representation for video action recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Tan, C., Li, C., He, D., Song, H. (2022). Towards real-time tracking and counting of seedlings with a one-stage detector and optical flow. Computers and Electronics in Agriculture p 106683

Tan, M., Pang, R., Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 10778–10787

Teed, Z., Deng, J. (2020). RAFT: recurrent all-pairs field transforms for optical flow. In Vedaldi A, Bischof H, Brox T, Frahm J (eds) European Conference on Computer Vision (ECCV), pp 402–419

Wang, D., Li, M., Gong, C., Chandra, V. (2021). Attentivenas: Improving neural architecture search via attentive sampling. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6418–6427

Wang, X., Girshick, R. B., Gupta, He, K. (2018). Non-local neural networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 7794–7803

Wulff, J., Sevilla-Lara, L., Black, M. J. (2017). Optical flow in mostly rigid scenes. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 4671–4680

Xiao, T., Yuan, J., Sun, D., Wang, Q., Zhang, X. Y., Xu, K., Yang, M. H. (2020). Learnable cost volume using the cayley representation. In European Conference on Computer Vision (ECCV), pp 483–499

Xie, S., et al. RBG (2017). Aggregated residual transformations for deep neural networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Xu, H., Zhang, J., Cai, J., Rezatofighi, H., Tao, D. (2022). Gmflow: Learning optical flow via global matching. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 8121–8130

Xu J, Ranftl, R., Koltun, V. (2017). Accurate optical flow via direct cost volume processing. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 1289–1297

Xu, Y., Wang, Y., Han, K., Tang, Y., Jui, S., Xu, C., Xu, C. (2021). Renas: Relativistic evaluation of neural architecture search. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Yang, G., Ramanan, D. (2019). Volumetric correspondence networks for optical flow. In Neural Information Processing Systems (NeurIPS), pp 793–803

Yang, Z., Li, Z., Shao, M., Shi, D., Yuan, Z., Yuan, C. (2022). Masked generative distillation. In European Conference on Computer Vision (ECCV)

Yin, Z., Darrell, T., Yu, F. (2019). Hierarchical discrete distribution decomposition for match density estimation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6044–6053

Yu, J., Jin, P., Liu, H., Bender, G., Kindermans, P., Tan, M., Huang, T. S., Song, X., Pang, R., & Le, Q. (2020). Bignas: Scaling up neural architecture search with big single-stage models. *European Conference on Computer Vision (ECCV), 12352*, 702–717.

Yu, J., Yang, L., Xu, N., Yang, J., Huang, T. S. (2019). Slimmable neural networks. In International Conference on Learning Representations (ICLR)

Yuan, F., Shou, L., Pei, J., Lin, W., Gong, M., Fu, Y., Jiang, D. (2021). Reinforced multi-teacher selection for knowledge distillation. In Association for the Advancement of Artificial Intelligence (AAAI)

Zagoruyko, S., Komodakis, N. (2017). Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In International Conference on Learning Representations (ICLR)

Zhang, F., Woodford, O. J., Prisacariu, V. A., Torr, P. H. (2021). Separable flow: Learning motion cost volumes for optical flow estimation. In IEEE International Conference on Computer Vision (ICCV), pp 10807–10817

Zhang, H., Li, Y., Chen, H., Shen, C. (2020). Memory-efficient hierarchical neural architecture search for image denoising. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 3654–3663

Zhang, X., Zhou, X., Lin, M., Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6848–6856

Zhang, Y., Qiu, Z., Liu, J., Yao, T., Liu, D., Mei, T. (2019). Customizable architecture search for semantic segmentation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 11641–11650

Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J. (2017). Pyramid scene parsing network. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6230–6239

Zhao, S., Sheng, Y., Dong, Y., Chang, E. I., Xu, Y., et al. (2020). Maskflownet: Asymmetric feature matching with learnable occlusion mask. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp 6278–6287

Zoph, B., Le, Q. V. (2017). Neural architecture search with reinforcement learning. In International Conference on Learning Representations (ICLR)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.