Real-Time Optical Flow Estimation

Alex Zuzow
University of Texas at Austin
azuzow@utexas.edu

Charles Nimo
University of Texas at Austin
nimo@utexas.edu

Abstract

Major progress has been made for estimating optical flow using deep neural networks in recent years. However, optical flow estimation is an extremely computationally expensive task as it requires computing estimates of the motion between two time steps of a video or a sequence of images. We build upon the previous work RAFT which is the current state-of-the-art approach for optical flow estimation. In this work, by applying techniques such as pruning, quantization, and distillation we aim to improve the inference time of RAFT. Our experiments show that our model achieves similar performance while also being faster and light weight in comparison to original RAFT architecture.

1. Introduction

Optical Flow is a well known problem in computer vision that seeks to estimate the motion of objects between consecutive frames of a sequence caused by the relative movement between the object and the camera. Optical Flow estimation is common to many computer vision applications, considering the resulting flow field is very valuable for autonomous driving, object detection and tracking, ego-motion estimation, and collision avoidance. These issues often surface in the context of real applications such as robotics. A task as simple as computing real-time optical flow requires more processing power than is practical for mobile robotics. Generally, real time is defined as being fast enough to be calculated online and be usable in some reactive systems. In the past, this has proven to be very difficult as the calculation of optical flow is traditionally computationally intensive, very sensitive to noise, and often times inaccurate.

The current trend in optical flow research is to emphasize accuracy over computational resource limitations and robustness to noise, both together are essential for real-time robotics. While the accuracy of these very large, over-parameterized models has significantly increased, the growth in size means that it is not possible to deploy them for most resource-constrained applications. This results in the problem of utilizing real-time optical flow which requires real-time inference, with low energy consumption and high accuracy, in resource-constrained environments. If optical flow is not robust it cannot perform well in varied circumstances and in unpredictable environments and becomes useless in applications such as robotic vision regardless of how fast it runs or how well it can perform in certain conditions. In order to utilize real-time robotic vision, it is necessary to find a method of calculating optical flow that is less sensitive to noise and is computationally efficient. There have been recent breakthroughs in optical flow estimation using deep learning approaches that have shown promising potential. One such approach, RAFT (Recurrent All-Pairs Field transforms) [42] has become the new state-of-the-art approach for optical flows.

Performing an efficient, real-time optical flow with optical accuracy requires rethinking the design, training, and deployment of the neural network architecture. In general, there is a large body of literature that has focused on addressing these issues in neural network models by making the models more efficient in terms of latency, memory footprint, and energy consumption, while still providing optimal accuracy/generalization trade-offs. We evaluate a few of these approaches in our research.

In the original work, RAFT aims to solve the task of optical flow. The authors define this problem as estimating per-pixel motion between between video frames. They show impressive results as such as a 16% error reduction from the previous best published results. Although RAFT claims to have efficient inference time, it is still not applicable to applications that have real-time requirements. To achieve our goal of real time optical flow estimation we aim to improve RAFT to leverage knowledge distillation, and pruning by considering each

method's characteristics. In phase 1, we begin by creating a student model which will be taught by the pretrained RAFT model, this student model will be proportionally smaller compared to the teacher model. In phase 2, we prune and train the model using structured pruning.

2. Related Work

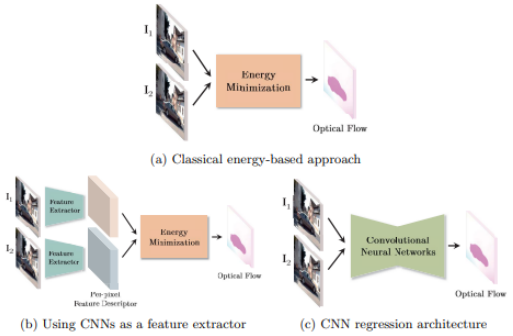
While earlier top performing techniques were mostly based on energy minimization methods, the current state of the art is dominated by deep learning approaches. The recent advances in deep learning have fueled a transition from classical-based energy-based formulations to end-to-end trained models. We first review how this transition proceeded by recapitulating early work that started to utilize neural network approaches, then summarize a few successful end-to-end approaches that have adopted CNNs for optical flow estimation and have highly influenced the direction of our research. Then we summarize the approach that is used in this research to estimate optical flow, including a review of its advantages over earlier architectures.

2.1. Traditional Optical Flow Strategies

Traditionally, optical flow has been studied as an energy minimization problem which sets a trade-off between a data term and a regularization term. For over three decades, research on optical flow estimation has been heavily influenced by the variational framework of Horn and Schnunk [19]. They formulated optical flow as a continuous optimization problem and were able to estimate a dense flow field by performing gradient steps. Originating from this basic formulation, several works of research have been focused on designing better energy models that more accurately represent the flow estimation problem [8][43].

2.2. Modern Optical Flow Strategies

More recently, the relative success of applying Convolutional Neural Networks (CNNs) with backpropagation on large-scale image classification tasks has paved the way on applying CNNs on other computer vision tasks such as optical flow [26]. Original work that applied CNNs to optical flow utilized them as a feature extractor [1] [1] [11] [13]. The main idea behind this use is to replace the energy term in classical energy-based formulations with a CNN-based feature matcher term. CNNs enable learning feature extractors such that each pixel can be represented as a high-dimensional feature vector that consolidates a suitable amount of distinctiveness and invariance. The feature distance gives



(a) Classical energy-based approach

(b) Using CNNs as a feature extractor

(c) CNN regression architecture

Figure 1: The progression of optical flow from traditional approaches (a) classical energy-based methods to (b) CNN-based methods that use CNNs as a feature extractor to (c) end-to-end trainable CNN architectures

the assumed similarity between regions. In these earlier optimization strategies using CNNs, the remaining workflow remained the same. Several of these methods demonstrated an accuracy benefit from CNN-based feature extractors.

At the same time, another line of active research investigates the development of end-to-end CNN architectures for optical flow estimation based on regression. In contrast to other methods, such as the ones mentioned previously, that only utilize CNNs for feature extraction, these regression approaches take advantage of CNNs for their entire workflow and directly output optical flow from a pair of images. By avoiding energy minimization, these approaches combine the advantages of end-to-end trainability and runtime efficiency.

Dosevitsky et al. proposed FlowNet [7], the first end-to-end CNN based architecture for estimating optical flow, which has two main architectural versions, FlowNetC and FlowNetS. Both models consist of an encoder and decoder components and differ only in the encoder part. With FlowNetS, a pair of input images are concatenated and then fed as input to a network that directly outputs optical flow. On the other hand, FlowNetC, uses a shared encoder for both images which outputs a feature map for each image, and then a cost volume is calculated by measuring patch-level similarity between the two feature maps with a correlation operation. To overcome the lack of suitable training data to train the models in a supervised way, Dosevitsky et al [7] created a synthetic dataset called FlyingChairs by layering natural images with rendered CAD models of chairs. However due to the intrinsic differences between real and synthetic data, FlowNet, when trained on syn-

thetic data only, did not generalize well to real images as compared to the state-of-the-art energy-based methods at the time [39] [45]. Despite this, however, FlowNet demonstrated the importance and possibility of employing an end-to-end regression architecture for optical flow estimation and established several standard practices for training optical flow models such as learning rate schedules, basic network architectures, and data augmentation schemes, to name a few.

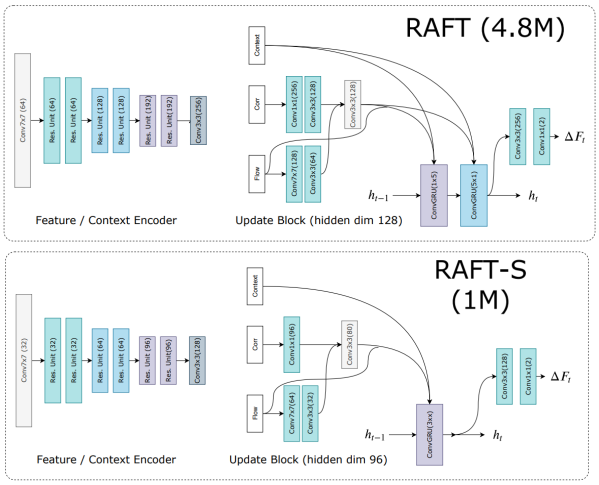


Figure 2: The network structure for the full 5.3 M parameter model and the smaller 1.0 M parameter model. Sharing common ground, the context and feature encoders of both architectures are the same. However, the only difference here is that the feature encoder uses instance normalization while the context encoder uses batch normalization. The residual units are replaced with the bottleneck residual units in RAFT-S. As input to the update block, it takes in context features, correlation features, and flow features to update the latent hidden state. The updated hidden status is then used to predict the flow update. The full model uses two convolutional GRU update blocks with 1x5 and 5x1 filters respectively, while the smaller model, RAFT-S uses a single GRU with 3x3 filters.

Later, Hur and Roth [22] proposed an iterative estimation scheme with weight sharing known as iterative residual refinement (IRR) that can be applied to several architectures to improve their accuracy further. Several recent works have used IRR to improve results for optical flow. The main idea behind IRR is to take the output from a previous pass through the network as input and iteratively refine it by using a single network block with shared weights such that it allows the network to residually refine the previous estimate. The IRR scheme can be used on top of various optical flow architectures such as FlowNet. The biggest difference between this approach and RAFT, which will be covered in more detail later in this study, is that with RAFT, weights are not shared between iterations. RAFT uses a much simpler refinement module that can be applied to 100+ iterations during inference without divergence. Moreover, Hur and Roth [22] also proposed an extension to joint occlusion and flow estimation that leads to further flow accuracy improvements while reducing the number of parameters in the model.

A recent breakthrough in deep optical flow estimation has been achieved by RAFT [42], which successfully attains state-of–of–the–the-art results. This work provides an effective backbone for unsupervised approaches in which flow is learned by optimizing photometric loss from synthesis views. RAFT can be viewed as a learning to optimize approach. The RAFT model uses a large number of update blocks to mimic the behavior of a first-order optimization algorithm. However, unlike earlier works, RAFT does not define a gradient with respect to some optimization objective. Instead, it retrieves features from correlation volumes to propose the descent direction. Raft constructs an all-pairs correlation volume where the displacement range contains the entire feature map to manage large displacements accurately. As of today, RAFT represents the state of the art in the field and therefore is the preferred choice to be enhanced by our research to optimize inference for optical flow estimation. Particularly, RAFT is preferred because of 1) its capability to compute matching scores between all pairs of pixels in the two input images, 2.) it possesses a much faster convergence time and 3.) its superior generalization capability. Figure 2 illustrates the architectural design of RAFT.

The latest research on optical flow is focused almost exclusively on pursuing higher accuracy which has resulted in methods that are computationally extensive and are hardly deployed to energy-constrained applications such as unmanned aerial vehicles, mobile phones, or embedded devices. In order to speed up accurate optical flow estimation and facilitate its use for practical applications, we propose using various network optimization techniques. The RAFT model outperforms other approaches by calculating all-pairs similarity and then performing iterations at a high resolution. This leads to a significant cost of high computational burden. Our aim in this research is to develop a smaller model while retaining good optical flow accuracy.

2.3. Knowledge Transfer

Knowledge distillation, which was popularized by Hinton et al. [18] is an approach that leverages information from a teacher network to train a smaller one. Following this, many algorithms have been proposed to improve upon knowledge distillation. The main idea behind knowledge distillation is that the student model mimics the teacher model in order to obtain a similar or even superior performance. The key problem of knowledge distillation is to determine how to transfer the knowledge from a large teacher model to a small student model. Essentially, knowledge distillation is composed of three parts - knowledge, distillation algorithm, and teacher-student architecture. In this paper, we apply knowledge distillation to achieve faster inference speed, along with other optimization techniques.

The different types of knowledge are categorized into three classes: Response-Based knowledge, Feature-Based Knowledge, and Relation-Based knowledge. Of the three classes mentioned,we perform knowledge distillation of the feature-based type. Feature-based knowledge captures knowledge of the data in its intermediate layers. Neural networks do well at learning multiple levels of feature representations with increasing abstraction, also known as representation learning [2]. Hence, both the output of the last layer and the output of the intermediate layers or feature maps, can be used as the knowledge to supervise the training of the student model. First introduced in Fitnets [40], where intermediate representations provide hints to improve the training of the student model. The main idea behind this is to directly match the teacher model's feature activations with that of the student's. Motivated by this, a variety of of other methods have been proposed to match features indirectly: Attention Maps [50]; Paraphraser [24]; Activation Boundaries [17]; Attention-based layer projection [38]; Feature Maps [4]; and Feature representation [44]

[50] explored deriving an attention map from the origin feature maps to express knowledge. The attention map was generalized using neuron selectivity transfer [20]. Later, transferred knowledge through matching the probability distribution in the feature space [37]. Kim et. al [24] made it easier to transfer the transfer knowledge when they introduced "factors" as a more understandable form of intermediate representations. Jin et. al [23] later introduced route constrained hint learning which supervises the student model with the outputs of hint layers from the teacher model, aiming to reduce the performance gap between teacher and student. Most Recently, [17] explored using the activation boundary of hidden neurons for knowledge transfer. Chen et al. explored using cross-layer knowledge distillation, which robustly assigns proper teacher layers for each student layer through attention allocation to match the semantics between teacher and student [4]. Shen et al. propose knowledge amalgamation, a model-reusing task where the goal is to learn a lightweight student model capable of handling the comprehensive classification given multiple trained teacher networks, each of which specializes in a different classification problem [41]. Chen et. al propose Semantic Calibration for Cross-layer Knowledge Distillation (SEMCKD) that automatically assigns target layers of the teacher model for each student layer with an attention mechanism for association weight learning, based on which knowledge could be transferred in a matched semantic space [3]. Liu et al, propose a knowledge representing (KR) framework that focuses on modeling the parameters distribution as prior knowledge using a knowledge aggregation scheme to represent the parameters knowledge from the teacher network into a more abstract level to alleviate the phenomenon of residual accumulation in the deeper layers [31]. Zhou et. al [51] propose a universal framework that exploits a booster network to help train the lightweight network for prediction in order to get neural networks of better performance, where the booster network is used to guide the learning of our lighter network throughout the whole training process. Guan et al. [12] explore using a Differentiable Feature Aggregation (DFA) search method that is motivated by DARTS in neural architecture search to find the aggregations efficiently.

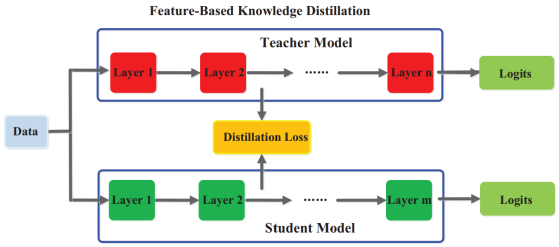


Figure 3: The architecture of the a generic feature-based knowledge distillation

Moreover, the distillation loss for feature-based knowledge transfer can be formulated as

$$L_{FeaD}(f_t(x), f_s(x)) = \mathcal{L}_F(\Phi_t(f_t(x)), \Phi_s(f_s(x))) \quad (1)$$

where $f_t(x)$ and $f_s(x)$ represent the feature maps of the intermediate layers of both the teacher and student models, respectively. Meanwhile, the transformation

functions, $\Phi_t(f_t(x))$ and $\Phi_s(f_s(x))$, are applied when the feature maps of both the teacher and student networks are not of the same shape. Figure 3 shows a generic model of a feature-based knowledge distillation architecture. Although feature-based knowledge transfer presents the advantage of providing information for the learning of the student model, the question of how to effectively choose the hint layers from the teacher model and the guided layers from the student model actually remains to be further investigated [40]. As a consequence of significant differences between sizes of guided and hint layers, how to match the feature representations of teach and student, properly, also needs to be investigated further.

Although knowledge distillation provides reasonable performance improvements, directly transferring the teacher's output overlooks the fundamental differences between the teacher network and the student network like the number of channels, network structure, and initial conditions. That is, when teaching a child, the teacher should not use their own terms because the child would have difficulty understanding it. Instead, if the teacher translates their terms into simpler ones, the child will have a much easier time with learning. In the same respect, the teacher network needs to deliver more understandable information to the student network such that the student understands the information easily. Kim et al [24] proposes a knowledge transferring method titled Factor Transfer that uses convolutional operations to paraphrase the teacher's knowledge and to translate it for the student model. This is accomplished by using two convolutional modules - a paraphrase and a translator. The paraphraser trains in an unsupervised manner to extract paraphrased information from the teacher network called teacher factors. The translator generates student factors which are trained to replicate the 'teacher factors'. The authors demonstrate that this approach effectively enhances the performance of the student network.

2.4. Pruning

Another technique that allows for the creation of a smaller model from an existing larger and over-parameterized model is known as pruning. Network pruning is a special optimization technique that can decrease both the parameter count , storage requirements, and improve the computational performance of inference without compromising accuracy . This allows neural networks to be deployed in constrained environments such as mobile devices and embedded systems. It is believed widespread that smaller networks pruned from over-parameterized networks achieve greater performance than those trained from scratch. A logical explanation behind this phenomenon is the lottery ticket hypothesis which describes that large networks may contain many optimal sub-networks or winning tickets. The aim of pruning is to remove redundant parameters or neurons that do not have significant contribution to the accuracy of results. This is realized when the weight coefficients are zero, close to zero, or are replicated. Consequently, it also reduces the computational complexity. The latest research in pruning can be divided into two components 1.) identifying the most promising neurons to be pruned and 2.) training and fine-tuning the pruned model to recover the base model's prediction performance.

Pruning can be categorized into two classes: structured pruning and unstructured pruning. Unstructured pruning, results in sparse weight matrices which does not directly speed up the inference efficiency unless specialized hardware/libraries are used [5] [27] [36] [47]. On the other hand, structured pruning aims to remove the redundant weights at the level of filters/channels/layers directly resulting in speeding up the network inference [16] [30] [15] [21] [49]. Pruning approaches are distinguishable by the amount of the network to prune at each step. Pruning can be accomplished using many types of methods, with each method chosen based on their desired output. Some methods prune a network with the desired compression ration and retain it for only one time, which is referred to as one-shot pruning [32]. On the other hand, some other methods repeat the process of pruning the network, and retrain it until the desired pruning rate or target size is achieved; this is known as iterative pruning [14]. Iterative pruning is widely used in machine learning and has shown that it could lead to a higher compression ratio compared to one-shot compression approaches. The authors of the lottery ticket hypothesis study, Frankle et al.[9] suggest that iteratively pruned networks at a small network size, learn faster and achieve higher test accuracy. Additionally, usually parameters are scored based on either their absolute values, trained importance coefficients, or contributions to network activations or gradients. For some pruning methods, scores are compared locally such that a fraction of the parameters with the lowest scores within each layer of the network are pruned. While other methods compare scores globally, comparing scores to one another regardless of the part of the network in which the parameter resides [29] [10].

When applied on a network, pruning can achieve many different goals, whether it be reducing the storage

footprint of the neural network, the computational cost of inference, the energy requirements of inference , etc. For each of these goals, different evaluation metrics are required. For example, when reducing the the storage footprint, one should evaluate the overall compression ratio achieved by pruning. Irrespective of the goal, pruning presents a tradeoff between model efficiency and quality. To quantify efficiency, most research studies take advantage of at least two metrics. The first is the number of multiply-adds, often referred to as FLOPs, required to perform inference on the pruned network. The second metric is the fraction of parameters pruned. As noted in several papers, these metrics are far from perfect [28] [6] [33] [48] [14] [25] [46] [34] [15]. Parameter counts are a loose representation for real-world latency, throughput, power consumption, and memory usage. However, because several research studies focus on these metrics, our analysis necessarily does so as well.

3. Method

In this work, we apply knowledge transfer to compress the RAFT-S network by using a method known as factor transfer (paraphraser). We then apply structural pruning to further compress the network. We choose to use this method because it is a feature-based knowledge distillation which focuses on extracting information from intermediate feature maps as opposed to response-based knowledge distillation which utilizing the resulting logits of a classification problem. Factor transfer is broken into two sections, a paraphraser and a translator.

3.1. Paraphraser

The paraphraser uses several convolutional layers to produce the teacher factor FT, which is further processed by a number of transposed convolution layers in the training phase. Most of the convolutional autoencoders are designed to down sample the spatial dimension to increase the receptive field. On the contrary, the paraphraser maintains the spatial dimension while adjusting the number of factor channels because it uses the feature maps of the last group which has a sufficiently reduced spatial dimension. If the teacher network produces m feature maps, we resize the number of factor channels as m × k. We refer to the hyperparameter k as a paraphrase rate. To extract the teacher factors, an adequately trained paraphraser is needed. The reconstruction loss function used for training the paraphraser is quite simple as $\ell_{rec} = \|x - P(x)\|^2$, where the paraphraser takes input feature maps x as an input and outputs P(x) feature maps. After training the paraphraser, it can extract the task specific features (teacher factors).

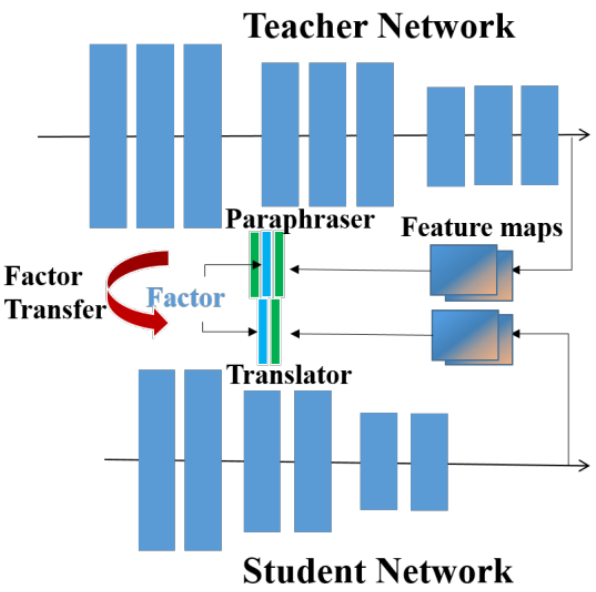


Figure 4: Here is a depiction of the Factor Transfor method. Unlike knowledge distillation, Factor Transfer does not compare the logits of the student and teacher networks.

3.2. Translator

The translator has has the job of converting the paraphrased information of the teacher network into terms the student network can understand. We add the translator to the last layers of the encoder section of the student network. The translator is trained jointly with the student network to relieve the student of the burden of directly learning the output of the teacher network by rephrasing the feature map of the student network. The student network and translator loss functions are defined as follows where ℓ_{raft} is the l_1 distance between the predicted and ground truth flow over the full sequence of predictions, $\{f_1, ..., f_N\}$:

$$\ell_{student} = \ell_{raft} + \beta \ell_{FT} \tag{2}$$

$$\ell_{raft} = \sum_{i=1}^{N} \|f_{gt} - f_i\|_1 \tag{3}$$

$$\ell_{FT} = \| \frac{F_T}{\|F_{T\|_2}} - \frac{F_s}{\|F_S\|_2} \|_{p.} \tag{4}$$

Here, F_T and F_S denote the teacher and student factors, respectively.

3.3. Iterative Pruning

Once our student model is fully trained, we then apply iterative pruning to further reduce the size of the network. We use structured pruning to remove a percentage of channels in the convolutional layers based on which have the lowest L_2 norm. After each pruning iteration, we retrain the student network and evaluate the results. We remove feature maps at rates of 10%, 30%, and 50%.

4. Experiments

We evaluate our student network using a validation set consisting of four optical flow datasets: FlyingChairs3D [7], Monkaa, Kitti, and Sintel. Our results show that we are not able to fully recover the accuracy as the original teacher network, although the model performance is not hindered greatly. This could be a result of us not having access to the same dataset the original authors of RAFT used for training as the torrent for said dataset has no seeders. We begin by pretraining our network on FlyingChairs [7], following RAFT [42] we apply photometric, spatial, and occlusion augmentations to our training set following . Photometric augmentations include randomly perturbing brightness, contrast, saturation, and hue. Spatial augmentations include randomly re-scaling and stretching of images. Lastly, to simulate occlusion we randomly erase rectangular regions. We begin by pretraining our student network with FlyingChairs3D and Monkaa [35] followed by finetuning using FlyingChairs3D, Monkaa, Sintel, and Kitti datsets.

4.1. Metrics

We use a single metric to evaluate our model performance, the main metric is end point error (epe) which is defined as the average of Euclidean distance between the predicted flow and the ground truth flow. we then compare the proportion of pixels with epe < 1 and epe < 3.

4.2. Results

In this section, we evaluate our student network using factor transfer for model compression on several datasets. We find that by utilizing factor transfer we are not able to fully maintain model performance however the loss in accuracy is minimal.

We then compare our student network to the reported results of RAFT and RAFT-S. In our experiments, we reduced our student network to approximately 500k feature maps before pruning, this is a reduction of 90% in comparison to RAFT and a 50% reduction in comparison to RAFT-S. In Table 1 we observe in neither the pretraining category or finetuning were we able to fully maintain epe accuracy, however, after finetuning our model we capture similar results to the finetuning step of RAFT-S. This could be largely due to the change in data used for pretraining and finetuning as we were unable to acquire the same datasets used in the original RAFT paper. To demonstrate we compare the results of the predicted flow from our model vs. RAFT-S, we do not show a direct comparison to the original RAFT due the difference in upsampling having a large effect on the visualization of results. Our results show extremely sim-

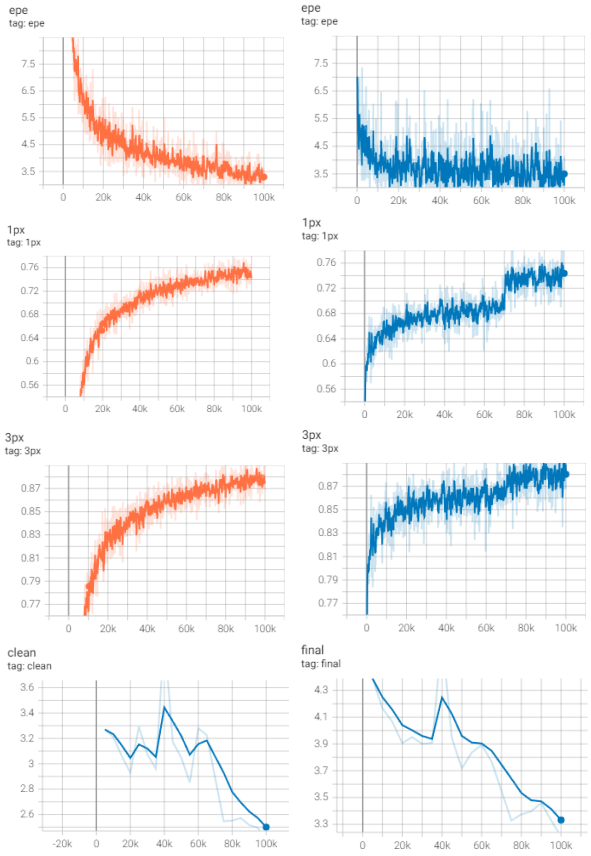


Figure 5: Here are the results of pretraining our student network in orange and finetuning in blue before pruning. The figures labeled clean and final are depicting the progression of epe with respect to iterations for the validation sets included with sintel. It should be noted that the clean and final figures start at 5k iterations due to us not recording validation information from the start. We pretrained for 100k iterations and finetuned on another 100k iterations. Every 5k iterations we evaluated the performance on our validation set.

Training Data	Method	Sintel		
		Clean	Final	Inference Time
C+M	RAFT	1.43	2.71	.1
	RAFT-S	2.21	3.35	.074
	Ours	3.33	4.68	.06
C+M+S/K	RAFT	.77	1.20	.1
	RAFT-S	-	-	-
	Ours	2.39	3.20	.06

Table 1: Here we compare the epe between RAFT, RAFT-S, and our student network on the sintel clean and final validation set. We then compare the average inference time of each network in seconds.

% Feature maps pruned	Sintel-Clean	Sintel-Final
0	2.39	3.20
10	2.84	3.60
30	5.16	6.12
50	10.56	11.23

Table 2: We show in this table the results of pruning our student network after finetuning, after each iteration of pruning we then re-train the network for 5000 iterations. We believe it is highly likely that accuracy could be improved upon if re-trained for a larger amount of iterations

ilar results to that of RAFT-S however, we do notice that the edges of objects seems to be interpreted differently frequently. The overall structure of optical flow is very similar in both models, the difference comes down to the fine details in most cases. We then apply iterative structural pruning to our student network, the results of which are shown in Table 2. Our network is able to produce intermediate flow at a rate which is 40% faster than the original RAFT and 18.91% faster than RAFT-S. We reduced the model memory requirements drastically while maintaining a noteworthy amount of accuracy.

5. Conclusion

In this paper, we have proposed applying knowledge distillation via Factor Transfer and pruning on the current state-of-the-art architecture, RAFT, with the aim of making optical flow suitable to real-time resource constrained applications. Our method consists of applying knowledge transfer to compress the smaller RAFT-S network and then apply structural pruning to further compress the network. We have demonstrated through our results, that our network is able to produce intermediate flow at a rate that is faster than both the original RAFT and RAFT-S architectures. We have also shown that our model maintains a great amount of accuracy considering that the model memory requirements were reduced drastically in comparison to the original networks.

(a) RGB (b) Ours (c) RAFT-S

Figure 6: Visual examples of predicted optical flow from three different models, the images come from our own personal dataset

References

[1] Min Bai, Wenjie Luo, Kaustav Kundu, and Raquel Urtasun. Exploiting semantic information and deep matching for optical flow, 2016.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives, 2014.

[3] Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration, 2021.

[4] Hanting Chen, Yunhe Wang, Chang Xu, Chao Xu, and Dacheng Tao. Learning student networks via feature embedding, 2018.

[5] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon, 2017.

[6] Michael Figurnov, Aijan Ibraimova, Dmitry Vetrov, and Pushmeet Kohli. Perforatedcnns: Acceleration through elimination of redundant convolutions, 2016.

[7] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks, 2015.

[8] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: A survey. *Computer Vision and Image Understanding*, 134:1–21, 2015. Image Understanding for Real-world Distributed Video Networks.

[9] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[10] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Stabilizing the lottery ticket hypothesis, 2020.

[11] David Gadot and Lior Wolf. Patchbatch: a batch augmented loss for optical flow, 2016.

[12] Yushuo Guan, Pengyu Zhao, Bingxuan Wang, Yuanxing Zhang, Cong Yao, Kaigui Bian, and Jian Tang. Differentiable feature aggregation search for knowledge distillation, 2020.

[13] Fatma Guney and Andreas Geiger. Deep discrete flow. volume 10114, pages 207–224, 03 2017.

[14] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015.

[15] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices, 2019.

[16] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks, 2017.

[17] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons, 2018.

[18] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[19] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185–203, 1981.

[20] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer, 2017.

[21] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks, 2018.

[22] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation, 2019.

[23] Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization, 2019.

[24] Jangho Kim, SeongUk Park, and Nojun Kwak. Paraphrasing complex network: Network compression via factor transfer, 2020.

[25] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications, 2016.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[27] César Laurent, Camille Ballas, Thomas George, Nicolas Ballas, and Pascal Vincent. Revisiting loss modelling for unstructured pruning, 2020.

[28] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky.

Speeding-up convolutional neural networks using fine-tuned cp-decomposition, 2015.

[29] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Snip: Single-shot network pruning based on connection sensitivity, 2019.

[30] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017.

[31] Junjie Liu, Dongchao Wen, Hongxing Gao, Wei Tao, Tse-Wei Chen, Kinya Osa, and Masami Kato. Knowledge representing: Efficient, sparse representation of prior knowledge for knowledge distillation, 2019.

[32] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning, 2019.

[33] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning, 2017.

[34] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression, 2017.

[35] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[36] Sejun Park, Jaeho Lee, Sangwoo Mo, and Jinwoo Shin. Lookahead: A far-sighted alternative of magnitude-based pruning, 2020.

[37] Nikolaos Passalis and Anastasios Tefas. Learning deep representations with probabilistic knowledge transfer, 2019.

[38] Peyman Passban, Yimeng Wu, Mehdi Rezagholizadeh, and Qun Liu. Alp-kd: Attention-based layer projection for knowledge distillation, 2020.

[39] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow, 2015.

[40] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015.

[41] Chengchao Shen, Xinchao Wang, Jie Song, Li Sun, and Mingli Song. Amalgamating knowledge towards comprehensive classification, 2018.

[42] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow, 2020.

[43] Zhigang Tu, Wei Xie, Dejun Zhang, Ronald Poppe, Remco C. Veltkamp, Baoxin Li, and Junsong Yuan. A survey of variational and cnn-based optical flow techniques. *Signal Processing: Image Communication*, 72:9–24, 2019.

[44] Xiaobo Wang, Tianyu Fu, Shengcai Liao, Shuo Wang, Zhen Lei, and Tao Mei. *Exclusivity-Consistency Regularized Knowledge Distillation for Face Recognition*, pages 325–342. 11 2020.

[45] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *2013 IEEE International Conference on Computer Vision*, pages 1385–1392, 2013.

[46] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks, 2016.

[47] XIA XIAO, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[48] Tien-Ju Yang, Yu-Hsin Chen, and Vivienne Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning, 2017.

[49] Shixing Yu, Zhewei Yao, Amir Gholami, Zhen Dong, Sehoon Kim, Michael W Mahoney, and Kurt Keutzer. Hessian-aware pruning and optimal neural implant, 2021.

[50] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, 2017.

[51] Guorui Zhou, Ying Fan, Runpeng Cui, Weijie Bian, Xiaoqiang Zhu, and Kun Gai. Rocket launching: A universal and efficient framework for training well-performing light net, 2018.